Performance Modeling, Analysis, and Tuning of Blockchain Networks

by

Yahya SHAHSAVARI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, DECEMBER 8, 2022

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

M. Kaiwen Zhang, Thesis supervisor
Département de génie logiciel et des technologies de l'information, École de technologie supérieure

M. Chamseddine Talhi, Thesis Co-Supervisor
Département de génie logiciel et des technologies de l'information, École de technologie supérieure

M. Georges Kaddoum, Chair, Board of Examiners
Département de génie électrique, École de technologie supérieure

M. Mohamed Faten Zhani, Member of the Jury
Département de génie logiciel et des technologies de l'information, École de technologie supérieure

M. Martin Maier, External Independent Examiner
Institut national de la recherche scientifique (INRS)

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON "NOVEMBER 22, 2022"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

There are no proper words to convey my deep respect and gratitude to my thesis and research advisor, Professor Kaiwen Zhang for his endless supports during the years of my study. I would like to appreciate him for the useful guidance and excellent company and the freedom that he gave me to pursue independent works in this way. As well, I would like to appreciate Professor Chamseddine Talhi for his helpful advice during the course of this research.

Further, yet importantly, my sense of respect and appreciation goes to entire my extended family, especially my parents for their endless love during my life. Without their support I could never have made it here. As well, I would like to express my warmest thanks to my love, Soodabeh. Her kind company and support during this study, before and after that will be always remembered.

I was fortunate to be a part of the FUSÉE laboratory. I would like to thank all my colleagues and friends in this lab for making an exciting and motivating environment.

Finally, I would like to express my thanks to staff members of École de Technologie Supérieure for their direct and indirect helps during the past years.

Modélisation de la performance, analyse et mise au point de réseaux de chaînes de blocs

Yahya SHAHSAVARI

RÉSUMÉ

La modélisation des performances de la chîne de blocs peut être utilisée pour nous fournir une compréhension plus approfondie du comportement et de la dynamique au sein des réseaux pair-à-pair chaîne de blocs. Un modèle théorique aidera les concepteurs de chaîne de blocs à mieux comprendre la dynamique et les caractéristiques sous-jacentes de la chaîne de blocs qui ont un impact sur les performances du réseau de chaîne de blocs. Un modèle théorique peut également accélérer le développement d'un système chaîne de blocs en découvrant rapidement une conception initiale théoriquement optimale, ce qui est préférable à une conception incrémentale via un benchmarking itératif.

Dans cette recherche, nous développons des modèles théoriques pour les chaîne de blocs publiques et privées. Nous utilisons ces modèles pour étudier l'impact de la topologie du réseau sur trois aspects différents des réseaux chaîne de blocs en termes de performances, de sécurité (forks) et de décentralisation.

Étant donné que le mécanisme de consensus dans la majorité des chaîne de blocs publiques est la preuve de travail (PoW), nous développons d'abord un modèle de performance pour le réseau de chaîne de blocs Bitcoin original et bien établi. Le modèle proposé peut être facilement adapté à d'autres chaîne de blocs bien connues basées sur PoW telles qu'Ethereum. Les réseaux de chaînes de blocs qui utilisent PoW dans leur mécanisme de consensus peuvent être confrontés à des incohérences sous la forme de fourches. Dans cette recherche, nous étudions la cause et la longueur des fourches pour le réseau Bitcoin. De plus, nous présentons une méthodologie pour quantifier le degré de décentralisation d'un réseau chaîne de blocs. Pour atteindre ces objectifs, nous utilisons deux modèles de graphes bien connus d'Erdös-Rény et Barabási–Albert afin d'étudier la topologie du réseau chaîne de blocs. Nous adaptons également notre modèle pour étudier l'impact du déploiement un réseau de relais et étudier l'effet de la taille du réseau de relais sur les performances du réseau et la décentralisation des chaînes de blocs basées sur PoW.Pour vérifier et valider les modèles développés, nous utilisons des données historiques extraites du réseau Bitcoin ainsi que des résultats obtenus à partir de la simulation à l'aide d'OMNet++ .

D'autre part, les protocoles byzantins tolérants aux pannes (BFT) sont des algorithmes classiques qui offrent un mécanisme de consensus plus rapide et plus économe en énergie par rapport au PoW. Hotstuff est un protocole de réplication de machine d'état BFT (SMR) partiellement synchrone qui vise à résoudre les problèmes de performances et d'évolutivité couramment rencontrés dans PoW. Nous proposons un cadre pour développer un modèle de performance pour les chaîne de blocs basées sur BFT et nous nous concentrons particulièrement sur HotStuff. Étant donné que HotStuff et d'autres chaîne de blocs basées sur BFT ont un mécanisme presque similaire, ce modèle peut également être adapté pour d'autres variantes du consensus BFT.

Mots-clés: Chaînes de blocs, Modélisation des performances, Crypto-monnaie, Bitcoin, Réseau pair-à-pair, Technologies de grand livre distribué, Réseaux de relais, Fourche chaîne de blocs, Modélisation théorique, Graphe Erdös-Rény, Graphe Barabási-Albert, Décentralisation, Hotstuff, Tolérance aux pannes byzantines a, Débit de transaction

Performance Modeling, Analysis, and Tuning of Blockchain Networks

Yahya SHAHSAVARI

ABSTRACT

Blockchain performance modeling can be used to provide us with a deeper understanding of the behaviour and dynamics within blockchain peer-to-peer networks. A theoretical model will help the blockchain designers obtain a better understanding of the underlying blockchain dynamics and characteristics which impact the performance of the blockchain network. A theoretical model can also accelerate the development of a blockchain system by quickly discovering a theoretically optimal initial design, which is preferable to an incremental design through iterative benchmarking.

In this research, we develop theoretical models for both public and private blockchains. We use these models to investigate the impact of the network topology on three different aspects of blockchain networks in terms of performance, security (forks) and decentralization.

Since the consensus mechanism in the majority of public blockchains is proof-of-work (PoW), we first develop a performance model for the original and well-established Bitcoin blockchain network. The proposed model can be easily adapted for other well-known PoW-based blockchains such as Ethereum. Blockchain networks that employ PoW in their consensus mechanism may face inconsistencies in the form of forks. In this research, we investigate the cause and length of forks for the Bitcoin network. In addition, we present a methodology for quantifying the decentralization degree of a blockchain network. To accomplish these objectives, we use two well-known graph models of Erdös-Rény and Barabási–Albert in order to study the blockchain network topology. We also adapt our model to study the impact of deploying a relay network and investigate the effect of the relay network size on the network performance and decentralization of PoW-based blockchains. For verifying and validating the developed models, we use historical data mined from the Bitcoin network as well as results obtained from simulation using OMNet++.

On the other hand, Byzantine Fault-Tolerant (BFT) protocols are classical algorithms that offer a faster and more energy-efficient consensus mechanism compared to PoW. Hotstuff is a partially synchronous BFT State Machine Replication (SMR) protocol that aims to address the performance and scalability issues commonly found in PoW. We propose a framework for developing a performance model for BFT-based blockchains and particularly focus on HotStuff. Since HotStuff and other BFT-based blockchains have an almost similar mechanism, this model can be also adapted for other variants of BFT consensus.

Keywords: Blockchain, Performance modeling, Cryptocurrency, Bitcoin, Peer-to-peer network,Distributed Ledger Technologies, Relay Networks, Blockchain fork, Theoretical modeling, Erdös-Rény graph, Barabási–Albert graph, Decentralization, Hotstuff, Byzantine Fault Tolerance, Transaction throughput

TABLE OF CONTENTS

Page

INTRODUCTION 1

CHAPTER 1 LITERATURE REVIEW 7
1.1 Blockchain networks 7
1.1.1 Public blockchains 7
1.1.2 Private blockchains 8
1.2 Performance modeling and analysis of public blockchains 8
1.2.1 Performance modeling and analysis of the Bitcoin network 8
1.2.2 Performance modeling of P2P networks 12
1.2.3 Analysis of forks in Bitcoin 12
1.3 Decentralization in blockchain networks 14
1.4 Performance modeling and analysis of private blockchains 15
1.4.1 Performance analysis of Hotstuff 15

CHAPTER 2 BACKGROUND 19
2.1 Public blockchains: background on Bitcoin 19
2.1.1 Cryptocurrency transactions 21
2.1.2 Blocks 22
2.1.3 Node 23
2.1.4 Mempool 24
2.1.5 Overlay network 24
2.1.6 Relay networks 26
2.1.7 Information dissemination 27
2.1.8 Forks 30
2.1.9 Traffic handling overhead 31
2.1.10 Erdös-Rény model 32
2.1.11 Barabási–Albert model 33
2.2 Private blockchains: background on Hotstaff and BFT 34

CHAPTER 3 A THEORETICAL MODEL FOR BLOCK PROPAGATION ANALYSIS IN BITCOIN NETWORK 37
3.1 Introduction 37
3.2 Analytical Model 39
3.2.1 Random graph construction 39
3.2.2 Achieving the connectedness properties 40
3.2.3 Legacy protocol block dissemination analysis 42
3.2.4 Performance metrics analysis of the legacy protocol 46
3.2.5 Compact block protocol and block propagation analysis 48
3.2.6 Performance metrics analysis of BIP152 52
3.2.7 Assumptions 53

3.3 Evaluation 53
3.3.1 Settings and implementation 54
3.3.2 Performance analysis of the Bitcoin network 56
3.3.3 Sensitivity analysis of the relay network 60
3.4 Managerial Implications 64
3.5 Conclusion 64

CHAPTER 4 A THEORETICAL MODEL FOR FORK ANALYSIS IN THE BITCOIN NETWORK 67
4.1 Introduction 67
4.2 Analytical Model 69
4.2.1 Fork model 69
4.2.2 Fork probability analysis 70
4.2.3 Block propagation model 71
4.3 Results and Analysis 77
4.3.1 Settings and implementation 77
4.4 Conclusion 80

CHAPTER 5 TOWARD QUANTIFYING DECENTRALIZATION OF BLOCKCHAIN NETWORKS WITH RELAY NODES 81
5.1 Introduction 81
5.2 System model and analysis 84
5.2.1 Performance analysis 86
5.3 Verification and validation 87
5.3.1 Simulation Model and Verification 87
5.3.2 Model validation 89
5.4 Results and analysis 91
5.4.1 Peer selection strategy 92
5.4.2 Shortest Path Analysis 94
5.4.3 Traffic Overhead Analysis 96
5.4.4 Relay Network Size 96
5.5 Conclusion and Future works 97

CHAPTER 6 PERFORMANCE MODELING AND ANALYSIS OF HOTSTUFF FOR BLOCKCHAIN CONSENSUS 99
6.1 Introduction 99
6.2 Proposed Analytical Model 101
6.2.1 System model 102
6.2.2 Analytical model 104
6.3 Model Validation and Analysis 110
6.4 Conclusion 112

CONCLUSION AND RECOMMENDATIONS 115
7.1 Summary 115

7.2 Future works 117

BIBLIOGRAPHY 119

LIST OF TABLES

Page

Table 5.1 Simulation results for Bitcoin. Targeted M was 32. 91

LIST OF FIGURES

Page

Figure 2.1 Overview of the Bitcoin network 19

Figure 2.2 Bitcoin blockchain data structure 20

Figure 2.3 Structure of a transaction 21

Figure 2.4 Internal structure of a block 22

Figure 2.5 A typical blockchain network with and without relay networks 26

Figure 2.6 Message structures of the inventory protocol 27

Figure 2.7 Block propagation protocols in Bitcoin 29

Figure 2.8 Fork with two competing branches N_1 and N_2 30

Figure 2.9 ER and BA graphs 33

Figure 2.10 Overview of the permissioned blockchain network 36

Figure 3.1 Minimum number of connections for connectedness 41

Figure 3.2 Block dissemination waves 43

Figure 3.3 Block propagation time in different modes 49

Figure 3.4 Long waves and short waves in block propagation 50

Figure 3.5 90% Block propagation delay vs. block size 57

Figure 3.6 Network traffic overhead (90%) 58

Figure 3.7 Block outreach vs. number of waves (90%) 59

Figure 3.8 90% block propagation delay vs. network size 59

Figure 3.9 Probability density function for waves 61

Figure 3.10 Probability density function for waves (1% relay network) 62

Figure 3.11 Probability density function for waves(2.5% relay network) 63

Figure 3.12 Probability density function for waves(5% relay network) 65

Figure 4.1 Network diagram when a fork occurs 72
Figure 4.2 Fork probability analysis based on network parameters 77
Figure 4.3 Branch weight analysis based on t' 78
Figure 5.1 Importance of the average shortest path 85
Figure 5.2 Degree distribution of nodes for BA network 86
Figure 5.3 Degree distribution of nodes for BA network 87
Figure 5.4 Schematic of a generated network for Bitcoin 91
Figure 5.5 Concept of dissemination waves in the Bitcoin network 92
Figure 5.6 Local and global clustering coefficient of BA network 93
Figure 5.7 Local and global clustering coefficient of ER network 94
Figure 5.8 Average shortest path for ER and BA networks for N = 10000 95
Figure 5.9 Total number of links (traffic overhead) for N=10000 95
Figure 5.10 Local and global clustering coefficient 96
Figure 6.1 Bernoulli loss model 103
Figure 6.2 Overview of the Hotstuff protocol 107
Figure 6.3 Theoretical vs. simulation results 111
Figure 6.4 Theoretical vs. simulation results 112

LIST OF ALGORITHMS

Page

Algorithm 3.1 Calculating the number of waves 48

Algorithm 3.2 Calculating the number of short waves 54

Algorithm 4.1 Calculating number of waves K and K' 76

Algorithm 5.1 Erdös-Rény model generation algorithm 88

Algorithm 5.2 Barabási–Albert model generation algorithm 90

LIST OF ABBREVIATIONS

API	Application Program Interface
BA	Barabási–Albert
BIND	Berkeley Internet Name Daemon
BFT	Byzantine Fault Tolerance
BIP	Bitcoin Improvement Proposal
DAG	Directed Acyclic Graph
DoS	denial of Service
DLT	Distributed Ledger Technology
DPoS	Delegated Proof-of-Stake
DNS	Domain Name Service
ER	Erdös-Rényi
HBM	High Bandwidth Mode
IoT	Internet of Things
LBM	Low Bandwidth Mode
P2P	Peer-to-Peer
PoS	Proof-of-Stake
PBFT	Practical Byzantine Fault Tolerance
PDF	probability density function
PoW	Proof-of work

PoS	Proof-of-Stake
SMR	State Machine Replication
SPV	Simple Payment Verification
TPS	Transaction per Second
TTP	trusted third party

INTRODUCTION

As a newly emerging and disruptive technology, blockchain-based systems have now applications far beyond cryptocurrencies (Narayanan, Bonneau, Felten, Miller & Goldfeder, 2016) in variety of domains: decentralized applications such as cloud storage (Wilkinson *et al.*, 2016), health care (Gordon & Catalini, 2018; Rabah, 2017), logistic and supply chain (Perboli, Musso & Rosano, 2018), energy sector (Andoni *et al.*, 2019), agriculture and food production (Ge *et al.*, 2017) legal and law enforcement (Sajjad *et al.*, 2017; Levy, 2017; Christopher, 2016) as well as pervasive financial services (Fanning & Centers, 2016) and Internet of Things (IoT) (Dai, Zheng & Zhang, 2019) use cases. The common goal of using blockchain technology in all aforementioned applications is to provide a live, immutable, safe and secure operation platform.

In order to properly operate, blockchain-based systems need a set of protocols and infrastructures referred to as Distributed Ledger Technology (DLT) (Rauchs *et al.*, 2018). Depending on the architecture, consensus mechanisms and data structure, blockchain-based systems may utilize different DLTs. Blockchain-based systems may have a permissioned or permission-less architecture (Xu *et al.*, 2017a). Moreover, consensus mechanism is another crucial part of these systems. Most prominent consensus mechanisms in blockchain-based systems are Proof-of work (PoW) (Gervais *et al.*, 2016), Proof-of-Stake (PoS) (Nguyen *et al.*, 2019), Delegated Proof-of-Stake (DPoS) (Yang *et al.*, 2019a), and Practical Byzantine Fault Tolerance (PBFT) (Castro, Liskov *et al.*, 1999). Furthermore, data in different DLTs can be stored in different data structures i.e. blockchain or Directed Acyclic Graph (DAG) (Benčić & Žarko, 2018).

Depending on the use case and operational goals, each of the mentioned systems will have different architectural design requirements. As well, each of them will also have a variety of configurable parameters, such as block size, block time, or the number of peers. The joint impact of various architectures and various configurations may lead to different performance and operational characteristics such as different amounts of block processing and propagation time,

throughput (i.e. number of processed transactions per time unit), degree of decentralization, or consumed energy.

In light of the above, we argue that having an appropriate model of the proposed blockchain systems seems to be necessary for blockchain developers as well as management operators in order to choose the right architecture and properly set the configuration metrics or tune them to maximize the performance of the system.

In addition, an analytical model will help the blockchain designers in obtaining a better understanding of the underlying blockchain dynamics and characteristics, which impact the performance of the blockchain network. A theoretical model can also accelerate the development of a blockchain system by quickly discovering a theoretically optimal initial design, which is preferable to an incremental design through iterative benchmarking. As well, analytical models can be useful for gaining a deeper insight into accidental disorders or intentional manipulations in the systems such as forks in blockchain-based systems.

In order to accomplish the objectives of this research, we first classify blockchain networks into two groups public (permissionless) and private (permissioned) networks.

Thesis contributions

In this context, we aim to propose a methodology for developing analytical performance models for public and private blockchain-based systems. We use these models to investigate the impact of the network topology on three different aspects of blockchain networks in terms of performance, security (forks), and decentralization. The contributions of this research are as follows.

- We develop an analytical performance model for Bitcoin as the representative of public blockchains. This model investigated the impact of network topology and configuration parameters in terms of block size, number of participating nodes, number of connections per node, and relay network size on the overall performance of the Network.

- We extend the developed model in the previous step to investigate the impact of the aforementioned configuration parameters on the probability of a fork in the network. It is important to study forks in blockchain-based systems since forks create inconsistencies across the local copies of the ledger,
- We present a methodology for quantifying the decentralization degree of a blockchain network. This model studies the impact of relay networks on the decentralization of the blockchain network. It is important to study decentralization degree in blockchain network since a poorly decentralized blockchain network can encounter issues such as governance issues, being a single point of failure, or being vulnerable to DoS attacks.
- We develop an analytical performance model for Hotstuff as the representative of consensus mechanism in private blockchains. This model investigated the impact of network size, packet loss, and transaction processing time on the performance of the network.

Summary of publication

This section briefly presents the articles published in the course of this research. Publications listed based on the aforementioned list of contributions, respectively.

A Theoretical Model for Block Propagation Analysis in Bitcoin Network: This article is published in IEEE transactions on engineering management (IEEE TEM), 2020 (Shahsavari, Zhang & Talhi, 2020). This article is the extension of a conference paper published in the IEEE International Conference on Decentralized Applications and Infrastructures (IEEE DAPPCON), 2019 (Shahsavari, Zhang & Talhi, 2019b). This paper won the best paper award of the conference. In this article, the theoretical model already presented in the conference version is revised and considerably extended. The focus of this article is on the design space surrounding the original and most well-known blockchain system, i.e., Bitcoin. A random graph model is used for performance modeling and analysis of the inventory-based protocol for block dissemination. This model addresses the impact of crucial blockchain parameters on the overall performance

of Bitcoin. Furthermore, the former model is adapted to study the impact of deploying a relay network and investigates the effect of the relay network size on network performance and decentralization. The results show the trade-off between the default number of connections per node, network bandwidth, and block size in order to compute the optimal block propagation delay over the network. Another significant finding presented in this article is that a bigger relay network can jeopardize the decentralization of the Bitcoin network with a higher probability.

A Theoretical Model for Fork Analysis in the Bitcoin Network: This paper is published in IEEE International Conference on Blockchain (Blockchain), 2019 (Shahsavari, Zhang & Talhi, 2019a). This paper presents a theoretical model for modeling forks in public blockchain networks such as Bitcoin. Like the former papers mentioned above, this model is based on a random graph for modeling the Bitcoin overlay network. This research studies the effect of key blockchain parameters on the fork occurrence probability, such as block propagation delay, network bandwidth, and block size. As well, it shows that Bitcoin will not benefit from increasing the number of connections per node. In addition, this model can be leveraged to estimate the weight of fork branches.

Toward Quantifying Decentralization of Blockchain Networks With Relay Nodes: This article is published in Frontiers in Blockchain, 2022 (Shahsavari, Zhang & Talhi, 2022a). The aim of this research is to present a methodology for quantifying the decentralization degree of a blockchain network. To accomplish this goal, this paper leverages two well-known graph models of Erdös-Rény and Barabási–Albert to model blockchains network topology. It then quantifies the decentralization degree using the clustering coefficient of our network models. The solution presented in this paper exposes the trade-off between the average shortest path and the decentralization degree. As well, it shows the impact of the average shortest path on the network speed and traffic overhead. As another result, this paper demonstrates that the

presence of hub-like nodes such as relay gateways negatively impacts the decentralization degree of blockchain networks.

Performance Modeling and Analysis of Hotstuff for Blockchain Consensus: This paper is published in IEEE International Conference on Blockchain Computing and Applications (BCCA), 2022 (Shahsavari, Zhang & Talhi, 2022b). In this paper, a theoretical approach for performance modeling and analysis of Hotstuff consensus mechanism is presented. This model can accurately predict blockchain-related metrics such as the transaction throughput and expected confirmation time using important networking parameters such as the number of replicas, link latency, and packet loss. A crucial finding of this paper is that Hotstuff performance degrades drastically when the number of replicas increases. As well, it has shown that packet loss ratio and transaction processing time are two other factors that significantly affect the performance of Hotstuff.

The rest of this thesis is organized as follows. In Chapter 1, we present the related works to our objectives. Chapter 2 briefly presents the background knowledge required for understanding this thesis. Chapter 3 presents an Erdös-Rényi (ER) random graph (Erdős, Rényi *et al.*, 1960) model for modeling the Bitcoin network. An extension of this model is used in Chapter 4 for modeling and analysis of forks in the Bitcoin network. In Chapter 5, a methodology for quantifying decentralization in blockchain networks is presented. To accomplish this, we use both ER and Barabási–Albert (BA) random graph (Albert & Barabási, 2002) models for modeling the topology of blockchain networks. Finally, an analytical performance model for practical Byzantine fault-tolerant (PBFT) blockchains is presented in Chapter 6,

CHAPTER 1

LITERATURE REVIEW

This chapter presents some of the existing research works that are close to our subject. The related works we discuss in this chapter are organized in line with the list of the objectives of this research as well as published papers and follow the steps of our methodology.

1.1 Blockchain networks

To accomplish the goals of this chapter, we divided blockchain-based systems into two groups public and private systems. We study the related works of each group separately.

1.1.1 Public blockchains

Ethereum and Bitcoin are two well-known blockchains that fall into the category of public blockchains. The consensus mechanism in these blockchains is proof of work (PoW) that involves a gossip protocol for block propagation over the network. Public blockchain networks can consist of a relay network or operate without it. As well, public blockchains with a PoW consensus may encounter inconsistencies referred to as forks. We develop a theoretical model for the Bitcoin overlay network using an Erdös-Rény model (Erdős & Rényi, 1959) to generate connected random graphs. Bitcoin and Ethereum have almost the same block propagation mechanisms and this model can easily be adapted for Ethereum. As well, we use Barabási–Albert (BA) (Barabási & Albert, 1999) model for modeling blockchains whne we consider them with relay networks. As explained in Chapter 5, decentralization rather matters in public blockchains. Hence, we dedicate our study of decentralization in blockchain networks to only public blockchains.

1.1.2 Private blockchains

Currently, the majority of operational private blockchains are based on one of the Byzantine fault tolerance (BFT) algorithms. BFT consensus is a voting-based mechanism in which participants vote for a proposal suggested by the leader validator. Hence, forks do not usually matter in private blockchains. We develop a BFT performance model based on recursive equations.

1.2 Performance modeling and analysis of public blockchains

Existing research works related to performance modeling and analysis of public blockchain networks are presented in this section.

1.2.1 Performance modeling and analysis of the Bitcoin network

To the best of our knowledge, there exist few works on performance modeling and analysis of blockchain networks, especially from a theoretical perspective. The majority of existing works rely on analysis based on simulation or experimental data gathered from blockchain networks.

In (Neudecker, Andelfinger & Hartenstein, 2016), a theoretical model for the propagation delay based on the path length is presented. However, this model is not validated by realistic data. Moreover, this model does not describe the impact of different configuration parameters such as block size, the average number of connections per node, network bandwidth, and network size (i.e total number of nodes) on the performance of the network.

In (Misic, Misic, Chang, Motlagh & Ali, 2019), an analytical model for the Bitcoin network based on the Jacksonian queuing network model is presented. In this work, the data (i.e. transactions and blocks) forwarding is modeled by branching processes in the network with a random distribution of node connectivity, and arrival of blocks and transactions is modeled as a non-homogeneous Poisson process. However, this work suffers in some aspects. For example, it does not validate or verify the proposed model. Moreover, the evaluation is done for a network size of 2500-5000 nodes which is not a realistic assumption regarding the current size

of the Bitcoin network. Like the previous work, this paper does not address the effects of some important configuration parameters on the overall performance of the network.

(Nagayama, Shudo & Banno, 2019) analyses the block propagation delay in the Bitcoin network using a network simulator. The paper conducts experiments comparing the compact block propagation protocol against the legacy block propagation protocol. According to the results, the block propagation delay is reduced by 90.1% and 87.6% for the 50th and 90th percentile respectively. However, this work uses unrealistic values for some of the parameters. In this paper, we consider realistic values for the parameters.

(Donet, Pérez-Sola & Herrera-Joancomartí, 2014) presents an analysis of collected data mined from Bitcoin. This work analyzes the network in terms of node geographic distribution, network stability, transaction propagation time, and block propagation time. According to this work, the Bitcoin network is homogeneously distributed all over the world, except in countries with very low population count. The work claims that although the delay in the Bitcoin network can be acceptable for regular nodes, nevertheless it is still too high for some miner nodes, which causes them to continue working on a block already found.

(Decker & Wattenhofer, 2013) presents an analysis of Bitcoin from a networking perspective. The main observation from this paper is that network delay is the main cause of forks in Bitcoin. This work studies the delay cost based on parameters such as block size, but does not provide an analytical approach: its results are gathered empirically from the Bitcoin network.

(Yasaweerasinghelage, Staples & Weber, 2017) demonstrates the feasibility of using architectural performance modeling and simulation tools to predict the latency of blockchain-based systems.

VIBES is a visual simulation tool for blockchain networks Stoykov, Zhang & Jacobsen (2017). This application can estimate performance measures with configurable blockchain parameters.

In (Fadhil, Owenson & Adda, 2016), an event-based simulation model for the Bitcoin overlay network is presented. To parameterize this model, a large-scale measurement of the real network

of Bitcoin is performed. However, this work does not provide any theoretical framework for modeling the Bitcoin network.

(Nasir, Qasse, Abu Talib & Nassif, 2018) performs a performance evaluation of two versions of Hyperledger Fabric (v0.6 and v1.0), measuring latency, execution time, and throughput, with various workloads. The paper also studies the scalability of Hyperledger Fabric using Hyperledger Caliper which is a benchmarking tool. Our work differs from this one as we develop an analytical model rather than present empirical results through benchmarking.

(Thakkar, Nathan & Vishwanathan, 2018) presents an empirical study on the performance of Hyperledger Fabric with the goal of identifying potential performance bottlenecks. This work aims to identify the impact of different configuration parameters (e.g., block size, endorsement policy, etc.) on transaction throughput and delay. It puts the emphasis on optimizing Hyperledger Fabric v1.0 in light of its results.

The majority of research works already done in the context of performance analysis of blockchain systems are in agreement that the current Bitcoin network suffers from poor performance and thus improvement schemes are necessary in order to scale up.

Since information propagation delay is one of the most important performance measures, different proposals have been presented to improve it. (Decker & Wattenhofer, 2013) and (Stathakopoulou, Decker & Wattenhofer, 2015) propose some optimizations to Bitcoin such as pipelining, increasing locality and using content distribution networks (CDNs) in order to speed up the network.

In (Hao *et al.*, 2019), an optimization mechanism based on geographical proximity sensing clustering for fast information broadcasting is proposed. In this approach, the participating nodes are grouped into clusters based on geographical proximity. Then, strong connectivity and minimum network diameter are ensured using node attribute classification. Finally, the parallel spanning tree broadcast algorithm is used to broadcast the data among the participating nodes.

The results of this paper demonstrate the lower delay of the solution compared to the current networks of Bitcoin and Ethereum.

A similar work about forming geographical clusters is presented in (Owenson, Adda *et al.*, 2017). The difference between this work and the work mentioned above is the use of ping packets in order to detect proximity with considerably high P2P link bandwidth. However, none of these works present an analytical model which can describe the relationship between configuration parameters (e.g., distance and bandwidth) and the information propagation delay.

Another approach to reducing the information propagation delay in blockchain networks is to deploy a central backbone of high-speed servers with a high degree called a relay network.

The Bitcoin Relay Network (Corallo, 2016a) was the first well-known and operational relay network that achieved a lower propagation delay by avoiding re-transmission of known transactions as well as full block verification. This system was eventually upgraded as the Bitcoin FIBRE (Corallo, 2019) which exploits cut-through routing with compact blocks and forward error correction (Watson, Begen & Roca, 2011) over UDP.

The Falcon Relay Network (Basu, Eyal & Sirer, 2019) is another relay network that uses cut-through routing in order to increase the block propagation speed. This relay network is directly connected to 36.4% of the total hash power in Bitcoin (Gencer, Basu, Eyal, van Renesse & Sirer, 2018).

(Klarman, Basu, Kuzmanovic & Sirer, 2018) presents a blockchain distribution network (BDN) that increases throughput to thousands of transactions per second (TPS) while reducing block propagation overhead without requiring any change to the blockchain protocol. The authors claim that BDN enables faster and Gigabyte-size block propagation, decreases inter-block times without increasing the risk of forks, reduces wasted miner effort, and improves fairness and decentralization compared to the aforementioned relay networks.

None of the aforementioned works provide a theoretical approach to performance modeling based on important configuration parameters. In contrast, we model the Bitcoin P2P network

using a random graph model in order to derive closed-form equations for two performance metrics: block propagation delay and network overhead. Moreover, we assess the impact of deploying relay networks on the performance of the system. To the best of our knowledge, this is the first work in which such a contribution is presented.

1.2.2 Performance modeling of P2P networks

Beyond blockchains, peer-to-peer networks and random graphs have been studied to some extent. (Kumar & Ross, 2006) studies the minimum achievable file distribution time in terms of basic network parameters such as file size, number of servers, number of receiving nodes and the upload and download capacities of participating nodes. In (Mundinger, Weber & Weiss, 2006), the authors calculate the minimal time to fully disseminate a file of M parts from a server to N end users in a centralized architecture. However, none of these two above works address the problem of average delay of file dissemination in a peer-to-peer network in decentralized scenario.

One of the most relevant papers to this thesis is (Oikonomou & Stavrakakis, 2007). This work investigates probabilistic flooding (randomly choosing the next neighbor node) when the underlying network is a random graph. The distribution of diameters in Erdös-Rényi graphs is studied in (Hartmann & Mézard, 2018). The diameter is useful to calculate the global outreach time in a decentralized peer-to-peer network. However, this work does not give any explicit equation for calculating these parameters.

1.2.3 Analysis of forks in Bitcoin

There exist many works in the literature which address different kinds of forks in Bitcoin. But to the best of our knowledge, there is no comprehensive theoretical model for analyzing forks based on their input parameters and network configurations. In addition, there exist several works which analyze forks using empirical data.

(Decker & Wattenhofer, 2013) analyses Bitcoin from a networking perspective. In this work, it is shown that the main cause of forks is the block propagation delay. According to the model presented in this network, the probability of forks is 1.78%, and an observed probability of 1.69% using their own empirical data. Our paper provides more detailed equations, which consider block size and network conditions, to accurately calculate fork probability.

(Biais, Bisiere, Bouvard & Casamatta, 2018) presents a taxonomy of blockchain forks. Furthermore, this work studies the generation of forks based on information delays and mining software upgrades. These two factors are orthogonal to our work.

In (Klarman *et al.*, 2018), an equation for estimating the fork occurrence probability based on 90% block propagation delay is presented. However, this work does not provide any theoretical proof or calculation approach for the mentioned equation.

(Nguyen & Kim, 2018) presents a comparison between Proof-of-Work (PoW), Proof-of-Stake (PoS), and other hybrid forms. According to this work, forks are more probable in PoW-based blockchains while in PoS-based systems, the chance is minimal. This work is orthogonal to our own as we focus on PoW-based systems, which are most susceptible to forks.

Although the Bitcoin white paper (Nakamoto, 2009) suggests modeling the block arrivals as a homogeneous Poisson process, authors in (Bowden, Keeler, Krzesinski & Taylor, 2018) have claimed that the block arrival times in the Bitcoin network follow instead a non-homogeneous Poisson process. Nevertheless, in our work, we build our model based on a homogeneous Poisson process to simplify our equations since it still provides a good approximation over a long period of time.

In summary, none of the works above provide a comprehensive and well-detailed theoretical model for the analysis of forks in the Bitcoin network, which is one of the main contributions of this thesis.

1.3 Decentralization in blockchain networks

While the quantification of the decentralization degree at the application layer of blockchains has been widely studied in the literature, to the best of our knowledge, there have been little or no work on quantifying and measuring the decentralization of the network layer of blockchain-based systems. Most of the existing works such as (Atzori, 2015; Puthal, Malik, Mohanty, Kougianos & Yang, 2018; Swan, 2015; Cai *et al.*, 2018) are geared towards introducing P2P networks as a new architecture to replace traditional server-client centralized networks. Furthermore, some of the research works have reported the tendency of blockchain systems such as Bitcoin towards a centralized architecture due to the existence of mining pools (Beikverdi & Song, 2015; Tschorsch & Scheuermann, 2016).

In (Wu, Peng, Xie & Huang, 2019), an information entropy-based approach for quantifying the degree of decentralization in Bitcoin and Ethereum is proposed. Using empirical data, the work compares the decentralization of mining and wealth in Bitcoin and Ethereum. However, it does not consider the networking aspects and configuration metrics of blockchain networks.

In (Chu & Wang, 2018), another attempt for quantifying the decentralization degree in blockchain systems has been carried out. In this work, decentralization is defined using the fraction of the transactions performed by top nodes. With this definition and related analysis, Chu, et al. have concluded that achieving full decentralization in blockchain networks is very hard due to the skewed mining power of the nodes. As well, they have claimed that full decentralization comes at the expense of limited scalability. Our work in this thesis is complementary as it studies the problem of decentralization from the perspective of the network.

(Li & Palanisamy, 2020) compares the decentralization degree of consensus protocols between Proof-of-Work (PoW) and Delegated Proof-of-Stake (DPoS). To accomplish this, the decentralization degree in Bitcoin and Steem is calculated using the Shannon entropy of the distribution of hash power and distribution of invested stake among stakeholders, respectively. According to this research work, Bitcoin is more decentralized between top miners but is overall less decentralized

than Steem. Unlike our work, network characteristics like topology and the average number of connections per node are not considered.

(Gencer *et al.*, 2018) presents a comparative measurement study on decentralization in Bitcoin and Ethereum. To accomplish this, this work relies on the measurement of network resources and evaluates the impact of a relay network (Falcon network). Authors of this work have reported that Bitcoin has more clustered nodes and that mining processes are fairly centralized in both of them. This work is purely empirical and does not propose an analytical technique unlike our work.

(Kwon, Liu, Kim, Song & Kim, 2019) proves that it is impossible for a permissionless blockchain to be fully decentralized using a concept known as the Sybil cost. This theoretical proof is based on the consensus protocol and does not consider network decentralization, which is the focus of our research.

None of the research works mentioned above have proposed an analytical technique for quantifying and comparing the decentralization degree in the network layer of the blockchain-based systems using the design and configuration metrics (e.g. number of selected peers, network size and etc).

1.4 Performance modeling and analysis of private blockchains

Byzantine Fault-Tolerant (BFT) protocols are classical algorithms that offer a faster and more energy-efficient consensus mechanism compared to PoW. Hotstuff is a partially synchronous BFT State Machine Replication (SMR) protocol that aims to address the performance and scalability issues commonly found in PoW. In this section, we study the related works to Hotstuff, as the representative of private blockchains.

1.4.1 Performance analysis of Hotstuff

BFT is the basis of Hotstuff. The performance of BFT and its different variants are well-studied in literature for several decades, especially from the impact of networking metrics

on the performance view point (Yang, 2018; Bano *et al.*, 2019; Agrawal & Daudjee, 2016; Momose & Ren, 2020; Distler, 2021; Berger & Reiser, 2018; Sukhwani, Martínez, Chang, Trivedi & Rindos, 2017; Hao, Li, Dong, Fang & Chen, 2018; Meshcheryakov, Melman, Evsutin, Morozov & Koucheryavy, 2021; Jalalzai, Busch & Richard, 2019; Li *et al.*, 2020).

Despite BFT, the performance of Hotstuff in terms of throughput and latency is still under-explored. Hotstuff was introduced in (Yin, Malkhi, Reiter, Gueta & Abraham, 2018). An explanation of the basic and pipelined HotStuff protocol and its specifications are presented in this white paper. In addition, a performance evaluation of Hotstuff in terms of throughput and latency is presented and compared to BFT-SMaRt (Bessani, Sousa & Alchieri, 2014). All the benchmarks are carried out versus network size (i.e. number of replicas). According to the results, three-phase and two-phase Hotstuff consistently exhibit better throughput while BFT-SMaRt shows considerably better latency than both Hotstuff variants. Three-phase Hotstuff also exhibits better throughput than the two-phase variant, but no meaningful difference in latency was seen. However, these results are based on experimental data and no theoretical model is presented.

In (Niu, Gai, Jalalzai & Feng, 2021), a multi-metric evaluation framework for performance analysis of pipelined Hotstuff regarding the chain quality, growth rate, and latency is presented. As well, several attack strategies and their impact on the performance of pipelined HotStuff are studied.

Diem, previously known as Libra, was the cryptocurrency introduced by Meta (re-branded name of Facebook) (bau). Diem was operating based on a consensus mechanism, namely Libra-BFT. Libra-BFT was a variant of Hotstuff with several distinguishing features. Contrary to popular cryptocurrencies such as Bitcoin and Ethereum, Diem operated on a permissioned network maintained by independent members of the Diem association. Libra-BFT and its performance were studied to some extent, before shutting down by Meta due to massive regulatory pushback in early 2022.

An experimental study of Diem is presented in (Zhang *et al.*, 2019). The performance and scalability of Diem are evaluated and compared with Hyperledger Fabric. The evaluation metrics are throughput (transactions per second, TPS) and execution time (i.e., the time taken to commit a block of transactions in the ledger). Parameters include the number of peers and the system workload (i.e. total number of transactions). According to the authors, the throughput of the system decreases and execution time increases when the number of peers increases for both platforms. As well, with respect to increasing the number of transactions, the throughput of both systems increases at first and then decreases. A reverse trend is reported for the execution time. Unlike our work, this paper is based only on experimental results and does not study the performance and scalability of the system in the presence of faulty nodes.

A comparison of Bitcoin, Ethereum and Libra is presented in (Li & He, 2020). However, it reports the results of (Zhang *et al.*, 2019) directly and does not provide any independent performance analysis.

In (Fu, Wang & Shi, 2021), a classification of consensus mechanisms in blockchain-based systems is presented. In this paper, HotStuff is studied as the basis of DiemBFT. According to this work, the advantage of Diem is its low confirmation latency. However, Diem suffers from low network size scalability, high communication cost, and weak network synchronization.

To the best of our knowledge, there is no theoretical model for performance modeling and analysis of Hotstuff that explains the effect of network parameters such as network size and the number of faulty replicas on the system throughput.

CHAPTER 2

BACKGROUND

This chapter provides the background knowledge required for understanding this thesis. Main concepts related to public and private blockchains are presented separately.

2.1 Public blockchains: background on Bitcoin

In this subsection, we study the information dissemination in Bitcoin networks, which can be broken down into two types: *(i) transactions dissemination* and *(ii) blocks dissemination*. In this thesis, we develop an analytical model for block propagation in the Bitcoin network and leave transaction propagation as future work. Nevertheless, the propagation mechanism in both systems is almost the same.

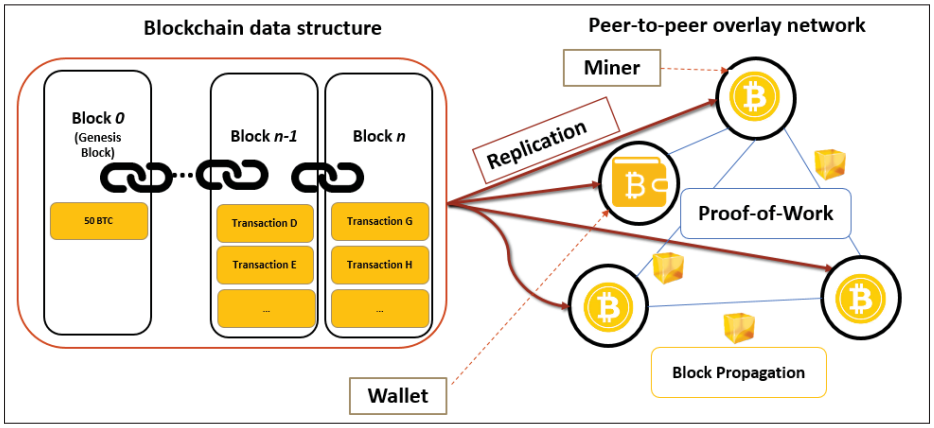


Figure 2.1 Overview of the Bitcoin network

Transactions are the atomic units of information of any blockchain system, which are then grouped into blocks to be entered into the distributed ledger. Figure 2.1 provides an overview of the different concepts related to Bitcoin architecture.

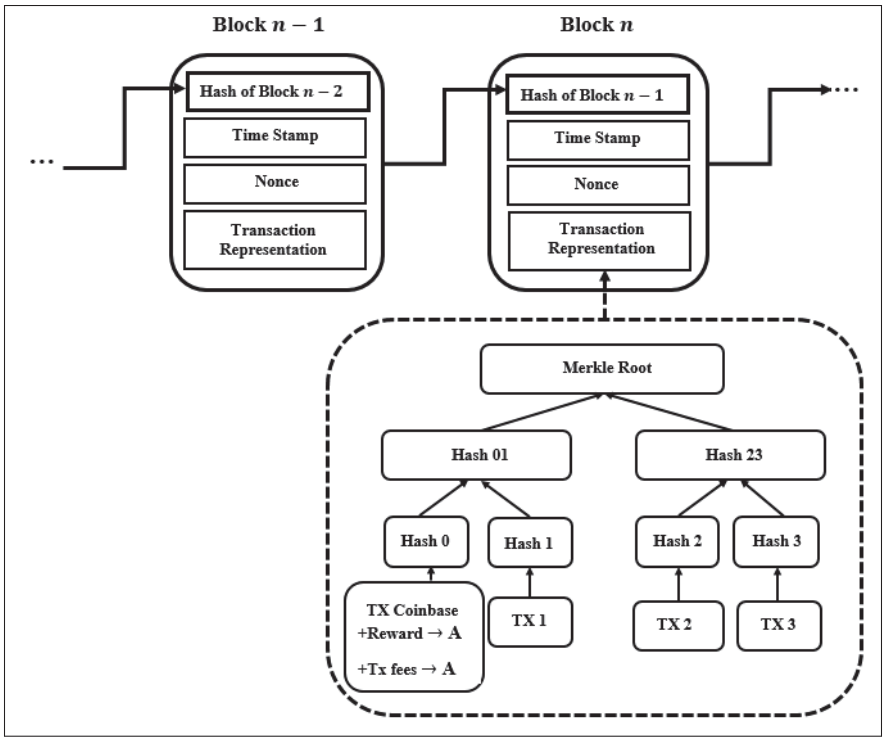


Figure 2.2 Bitcoin blockchain data structure

Note that although the scope of this thesis (and thus this background section) is limited to Bitcoin, our theoretical model can be generalized to other blockchains such as Ethereum (Buterin *et al.*, 2013)

Figure 2.2 illustrates the blockchain data structure. Blocks are linked together using the hash pointer of the previous block, starting from the root block (genesis block). In Bitcoin, the genesis block is hard-coded into the clients. These series of blocks are distributed, replicated, and stored in a ledger which is located in all participating nodes. This ledger is necessary for the verification of transactions embedded in each block. Participating nodes run over a peer-to-peer network and there is no intermediary node between them. Any node with access to this distributed ledger can read it and be notified about new data stored in the nodes.

2.1.1 Cryptocurrency transactions

In Bitcoin, transactions describe the transfer of digital coins between users. Each transaction consists of several input and output records. Each input indicates funds to be spent from previous transactions and output records indicate the amount transferred to the specified addresses (which are generated from the public key of the recipient). To be valid, each transaction should be digitally signed by the private key of the sender.

In each transaction, the sum of inputs should be equal or greater than the sum of outputs. There exists also special transactions (called *coinbase*) which have no input and grant a reward to the miner who successfully added the block.

For example, transaction m in Figure 2.3 takes as input #1 the output #2 from transaction $m - x$ and sends part of the funds to transaction $m + y$.

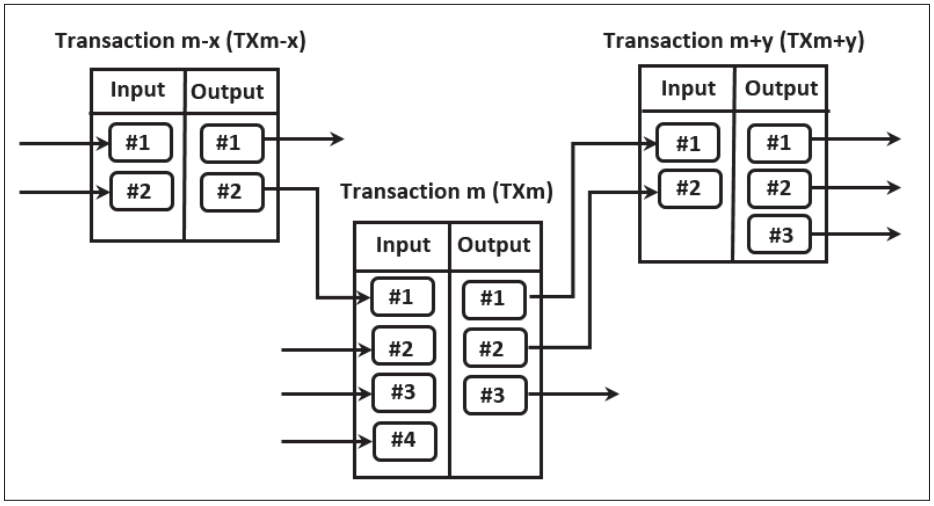


Figure 2.3 Structure of a transaction

2.1.2 Blocks

As depicted in Figure 2.4, each block consists of several components: a hash of the previous block, a timestamp, and transactions arranged to form a Merkle tree Narayanan *et al.* (2016).

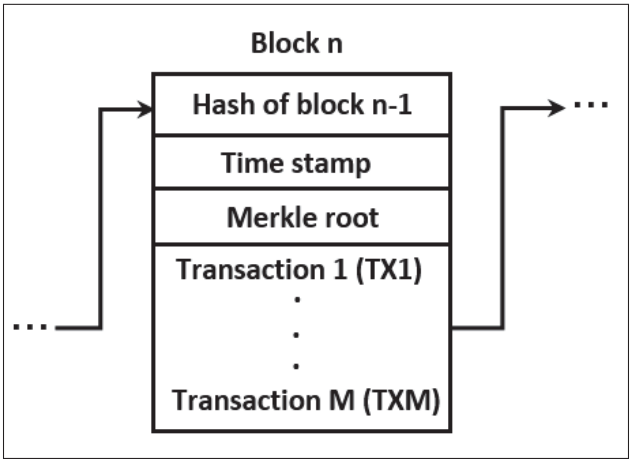


Figure 2.4 Internal structure of a block

To be entered in the distributed ledger, individual transactions have to be embedded in the blocks. A subset of nodes in the Bitcoin network, called miners, gather transactions already propagated by users and group them into blocks to be added to the blockchain. Grouping the transactions in the blocks is an optimization, since a hash chain of blocks is significantly shorter than a hash chain of transactions.

To be eligible to propagate a proposed block as the next block in the chain, each miner has to show a Proof-of-Work (PoW). To accomplish this, miners have to find a number called the nonce. The hash of the nonce, combined with the hash of the previous block, and the Merkle root of the tree containing the transactions proposed by the miner, should be below a certain target threshold. In other words,

$$H(nonce||H(prev\_block)||Root\{TX_1||...||TX_M\}) \tag{2.1}$$

while $H(prev\_block)$ is the hash of the previous block. This means that the result of Equation (2.1) should start with a certain number of leading zeros. The difficulty of the mentioned puzzle is adjusted periodically.

For a given block, the miner who first solves the PoW will broadcast his block over the network. The nodes which receive the block must verify the incoming block and add it to the local copy of the blockchain, thus becoming the new blockchain head. Since Bitcoin uses a very wide peer-to-peer network, inconsistencies in this network are unavoidable. When multiple valid blocks are solved and disseminated at the same time, we say that a fork has occurred in the network, which is too likely to be resolved by the following block mined (Eyal, Gencer, Sirer & Van Renesse, 2016).

In order to reduce the block propagation delay in the Bitcoin network, the compact block proposal was introduced in Bitcoin Improvement Proposal 152 (BIP 152). According to this proposal, compact blocks have the same metadata as legacy blocks. The main difference is that the hash of the block transactions are disseminated instead of a full copy of the transactions.

2.1.3 Node

The term *node* refers to a physical device that acts as a logical entity and can have one or some of the following functions:

Routing: This functionality is responsible for message propagation and maintaining connections to other participating peers in the network. since the network is unstructured, each node can potentially communicate with any other node in the system. The exception is nodes running inside a cooperative mining pool. In this case, the pool server is responsible for routing duties.

Full blockchain storage: The node locally stores a complete and up-to-date version of the blockchain data. Full nodes do not need an external reference for verifying transactions or blocks and perform this duty autonomously.

Lightweight blockchain storage: This type of node only keeps a lightweight version of the blockchain. Lightweight nodes verify transactions using a method called Simple Payment Verification (SPV) (Back *et al.*, 2014).

Mining (Consensus participation): Miners participate in consensus to decide the next block to be added to the Bitcoin blockchain. Miners are typically equipped with specialized hardware in a competition to find the next PoW. The winner collects the coinbase transaction of the created block.

Wallet management: Typically, wallet nodes are transaction issuer nodes and are commonly responsible for transferring digital coins from the sender to the receiver. Most Wallet nodes, particularly those running on resource-constrained devices, are lightweight SPV nodes.

2.1.4 Mempool

The nodes store unconfirmed transactions that are arriving from different links in a local memory pool (mempool). These transactions will remain in the mempool until they are included in the blockchain. Currently, the mempool can contain between 10^4 to 10^5 transactions, which a churn rate of 1300 to 2400 transactions per block (Imtiaz, Starobinski, Trachtenberg & Younis, 2019).

2.1.5 Overlay network

Bitcoin operates on an unstructured peer-to-peer network. When a node joins the network, the Bitcoin protocol allows the node to collaboratively maintain peer-to-peer connections with other nodes to exchange blocks and transactions.

When a node receives a transaction or block, it verifies the transaction or block. If the transaction or block is valid, the node relays it to other nodes. Thus, the verification process is helpful to avoid denial of service (DoS) attacks.

When a new node joins the network, it has no knowledge of the IP addresses of active nodes in the network. To discover this information, the new node queries a number of DNS servers

(or DNS seeds) which are run by volunteer nodes in the Bitcoin community. Whenever a DNS server receives such a query, it responds with a number of DNS records with the IP addresses of bootstrap nodes that may accept the incoming requests. DNS servers can be configured to get the IP addresses of active nodes either automatically or manually.

Once connected, peers can send *addr* (address) messages to other peers. This message consists of the IP address and port number of other nodes existing in the network. Also, newly joined nodes learn about other nodes by asking the neighboring nodes for known addresses or by listening to occasional advertisements of new addresses which are broadcasted in the network. Nodes may leave the network or change their IP address silently. Hence, it is possible that new nodes may have to try several attempts before successfully connecting to a peer. This can impose a considerable delay to a node bootstrapping time. To avoid this problem, Bitcoin can use dynamic DNS seeds in order to get the IP addresses of nodes with a high probability to be available.

Before a node can validate transactions and blocks, it must download and validate all the blocks and transactions of the known longest version of the blockchain ledger. Furthermore, there is no need to download the genesis block which is already hard-coded in the blockchain program. This process is called initial block download (IDB), which is done only once per new node. However, a node that was disconnected from the network for a significant amount of time may wish to run this process again.

Each participating node in the blockchain network maintains a connection pool, with connections to other peers active at all times. The minimum, maximum, and the average number of connections are indicated by N_L, N_U, and M, respectively. If the number of connections is below the predefined amount of N_L, the node will randomly select connections from known neighboring nodes and will attempt to establish new connections. Also, if the node accepts incoming connections from other nodes, it can maintain these connections open. It is reported that nodes running Bitcoin have an average of 32 connections, which far more than the default number of connections in Bitcoin, set to 8 (Decker & Wattenhofer, 2013).

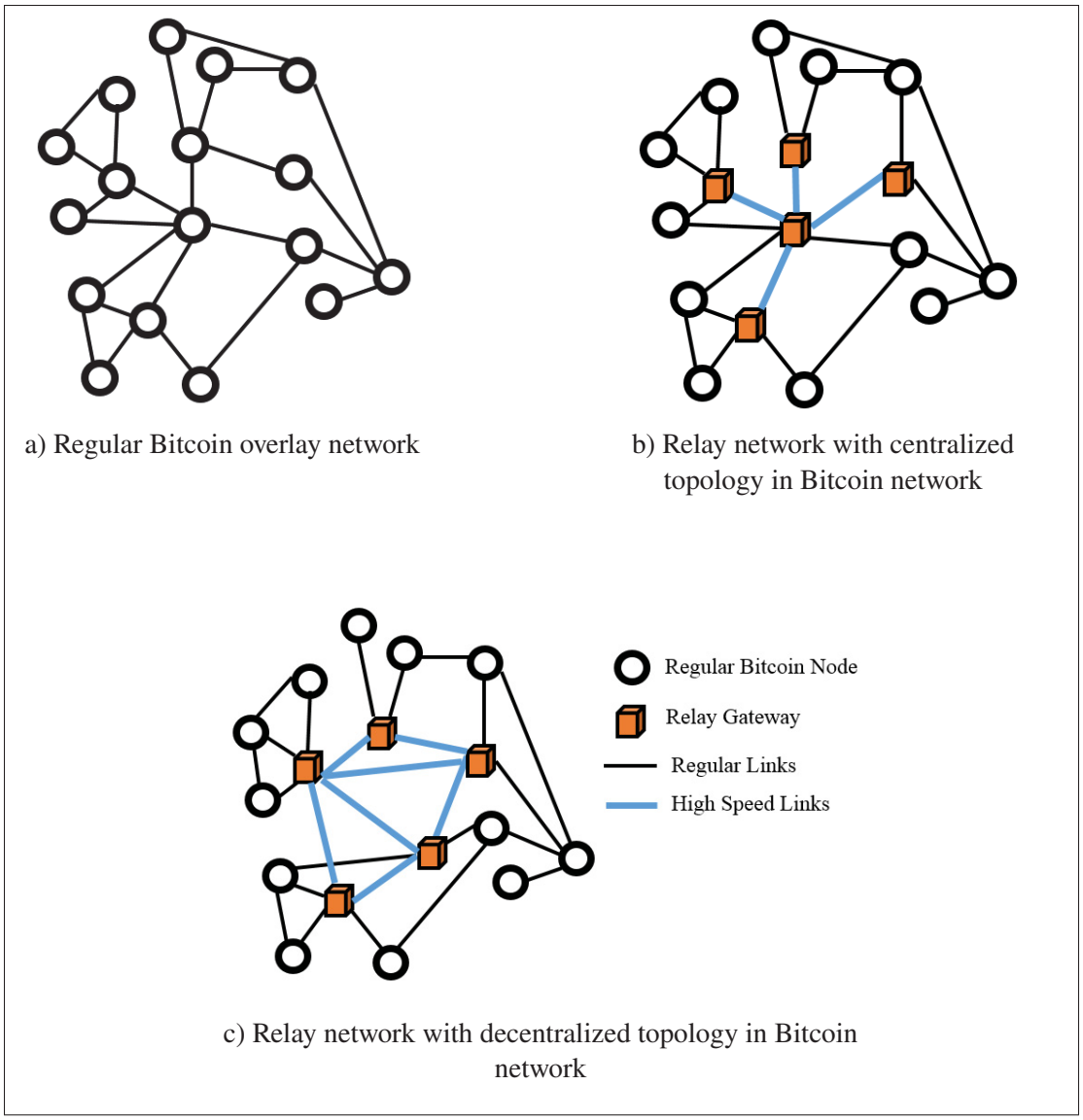


Figure 2.5 A typical blockchain network with and without relay networks

2.1.6 Relay networks

Generally speaking, a relay network is a set of global gateways with high-velocity links which form the backbone of the P2P network and connect to a large proportion of the network. Relay networks can propagate blocks and transactions much faster than the regular P2P network and thus increase the efficiency of information propagation. Relay networks can also be leveraged as a scalability solution for increasing the transaction throughput (Klarman *et al.*, 2018). Analytically,

relay networks can have a centralized or decentralized architecture. In a centralized architecture, the relay gateways interact with each other via a central orchestrator entity. But in decentralized architecture, the gateways are connected to each other in a P2P manner. Figure 2.5 illustrates a typical blockchain network with and without relay networks. However, relay networks are accused of decreasing decentralization since they can censor certain nodes, wallets, and miners or discriminate among different nodes by filtering the information they spread.

2.1.7 Information dissemination

In order to update the distributed ledger, transactions and blocks must be disseminated over the overlay network. To accomplish this, Bitcoin and most other blockchains use a gossip protocol (Lin & Marzullo, 1999). In order to avoid saturating the network with redundant copies of transactions and blocks, Bitcoin employs a push-based approach to sharing data.

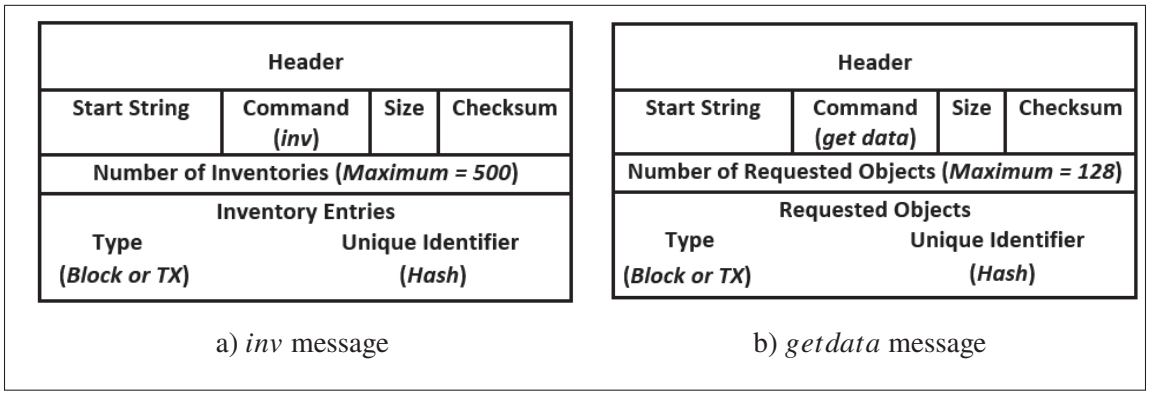


Figure 2.6 Message structures of the inventory protocol

Currently, two kinds of block propagation protocols are being used in the Bitcoin network. The legacy block propagation protocol and compact block propagation protocol which was introduced with BIP-152 (Corallo, 2016b). The compact block protocol is currently being used by more than 98% of nodes in the Bitcoin network[1].

[1] See the presentation by core developer Greg Maxwell: https://people.xiph.org/~{}greg/gmaxwell-sf-prop-2017.pdf

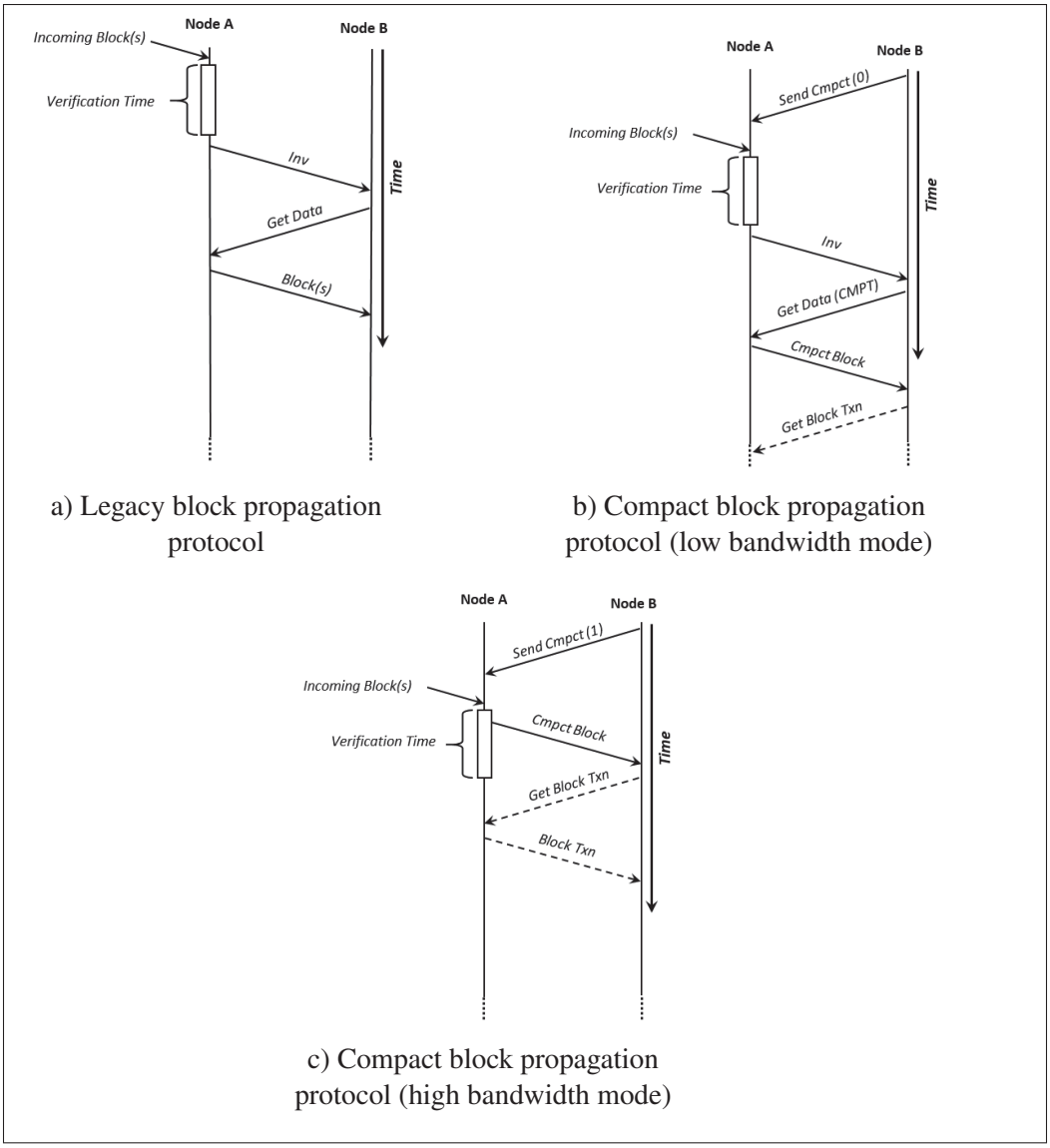
Legacy Block propagation protocol: Sending nodes notify neighboring nodes about the availability of a transaction or block once they are verified. To accomplish this, a node that wants to forward a transaction or block sends an inventory message (*inv*) to all nodes existing in its connection pool. The structure of the *inv* message is depicted in Figure 2.6a. An *inv* message consists of the hash of transactions or blocks that are now available and ready to be sent.

When a node receives an *inv* message for a block or transaction which it does not have yet, it replies with a $getdata$ message. As depicted in Figure 2.6b, this message contains the hash of requested transactions or blocks.

Once the sender node receives the $getdata$ message, it will send the requested block or transaction to the receiver. This process is depicted in Figure 2.7a.

Compact block propagation protocol: Compact block reduces the amount of bandwidth required for disseminating blocks in the Bitcoin network when the nodes are fairly synchronized and have already gathered a considerable amount of similar information (i.e., transactions) in their mempool. The main idea of this protocol is to let peers try to reconstruct entire blocks using their mempool content and a sketch of the blocks already received from the connected peers. In this case, transactions need to be sent over the network once only as they are not repeated during block propagation.

This protocol consists of two modes of operations: low bandwidth mode (LBM) and high bandwidth mode (HBM). In LBM (depicted in Figure 2.7b), node B notifies node A that it intends to minimize bandwidth usage by sending a $sendcmp(0)$ message. When node A receives a new block, it fully validates it. If the block is valid, then it informs node B about the reception of the new block using an *inv* message. If node B has already received this block, it will ignore it. Otherwise, it will respond to the *inv* message using a $getdata$ $(cmpct)$ message. Then node A will send the header of the new block, hash of transactions, and transactions that B is missing (guessed by A). If B receives all of the transactions necessary for reconstructing the new block, the protocol stops. Otherwise, it will ask node A to send transactions that are still missing.



a) Legacy block propagation protocol

b) Compact block propagation protocol (low bandwidth mode)

c) Compact block propagation protocol (high bandwidth mode)

Figure 2.7 Block propagation protocols in Bitcoin

In HBM (depicted in Figure 2.7c), node B notifies node A that it needs to receive blocks as soon as possible by sending a *sendcmp* (1) message. When a new block arrives, node A starts to perform some basic validation (e.g., checking the block header). Then it will send the header of the new block, the hash of transactions, and missing transactions to node B. Then, if the received information is adequate, node B will try to reconstruct the block. Otherwise, it will request information about missing transactions by sending a *getblocktxn* message to node A. Node A will respond to this message with a *blocktxn* message.

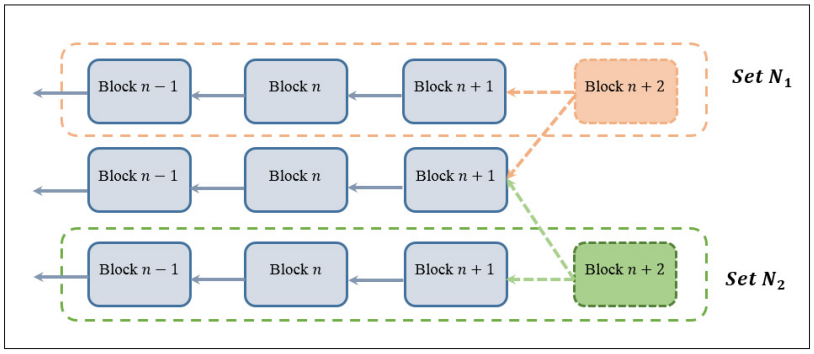


Figure 2.8 Fork with two competing branches N_1 and N_2

In both LBM and HBM modes, node A follows a simple approach to guess missing transactions in node B: It checks the recently arrived block and sees which transactions were in the block but not in its mempool. These transactions are the transactions that will be missing in the neighboring peers with high probability.

2.1.8 Forks

Bitcoin is a blockchain system operating over a vast P2P network around the globe. In such a large-scale network, inconsistencies in blockchain data may occasionally occur. The term *fork* refers to a situation in which different nodes have a different perspective of the blockchain. We can represent a fork as a directed tree of blocks. This concept is depicted in Figure 2.8 where two different sets of nodes have different states for the blockchain. In this case, the directed tree contains two separate branches. Generally, a fork can occur in one of the following cases:

Network isolation: Due to a poor connection between different nodes in the network, the network may become temporarily partitioned. This can occur if there are some very weak links acting as bottlenecks, or if there are not enough network connections in the system. As discussed in our previous work (Shahsavari *et al.*, 2019b), for a network with N participating nodes, to prevent this situation with high probability, it is sufficient:

$$M \geq \lceil \frac{N-1}{N} log(N) \rceil \tag{2.2}$$

Where M is the average number of connections between participating peers. This kind of fork will prolong until the network becomes connected again.

Changes in core components of the blockchain protocol: Any change in core components of the protocol such as the format of a valid block or transaction, difficulty retargeting function, or any upgrade in the mining software can cause a fork in the system. Changes that are incompatible with previous versions cause a hard fork, as opposed to soft fork.

Miners deviation from the standard protocol: The most well-known fork caused by deviating miners is the double-spending attack. Other examples of this kind of attack are temporary block withholding, selfish mining, feather forking attacks (Narayanan *et al.*, 2016), etc.

Block propagation delay: In practice, two or more different miners may find a valid block almost at the same time. If the first miner has not fully disseminated its block due to propagation delay, a fork occurs if a second miner starts gossiping its own proposed block. Increasing the block size to include more transactions has the drawback of raising the propagation delay, which increases the probability of fork occurrence. Therefore, we consider the trade-off between the block size and fork occurrence probability.

According to the Nakamoto consensus, this type of inconsistency is usually solved at the next block if miners select the longest chain (Nakamoto, 2009).

2.1.9 Traffic handling overhead

In blockchain networks, a traffic handling protocol is required in order to efficiently disseminate information (i.e. transactions and blocks) over the network. In most public blockchains such as Bitcoin and Ethereum, an inventory-based protocol is implemented in order to avoid overwhelming the network with redundant messages (Corallo, 2016b; Lange, Ballet & Toulme,

2016). According to these protocols, every node has to notify its neighbors that it has a new block/transaction to forward. The inventory message can be either a special message or hash of the block/transaction. If the targeted node does not have the new item, it will inform the inventory message sender, which will transfer the requested data. Therefore, each link in the network will transfer at least one inventory message to notify its neighbor about its content.

Figure 2.7 depicts the inventory-based gossiping protocol of Bitcoin. The number of links is a good parameter to count the minimum number of inventory messages required for the block/transaction dissemination over the blockchain network. Note that the number of links indicates the lower bound of the number of inventory messages and it can be more than this amount in practice.

2.1.10 Erdös-Rény model

The Erdös-Rény model (ER) is a random graph model for generating random graphs. In this model, in a network of N participants, two arbitrary nodes are connected to each other with the same probability of p in such a way that the average degree of nodes equals M. Hence,

$$p = \frac{M}{N-1} \tag{2.3}$$

In other words, this graph is chosen uniformly at random from the set of all possible graphs with N nodes and M links per node on average. This model can be suitable for modeling blockchain networks with no hubs or relay networks. As well it can be suitable for modeling large-scale public blockchains such as Bitcoin when the relay networks are not taken into account and instead, the average values of the network (e.g. average number of connections per node) are considered (Shahsavari *et al.*, 2019b).

The ER model does not exhibit a power-law degree distribution or preferential attachment. Instead, degree distribution in this model follows a Poisson distribution. A sample network generated with the ER model is depicted in Figure 2.9a.

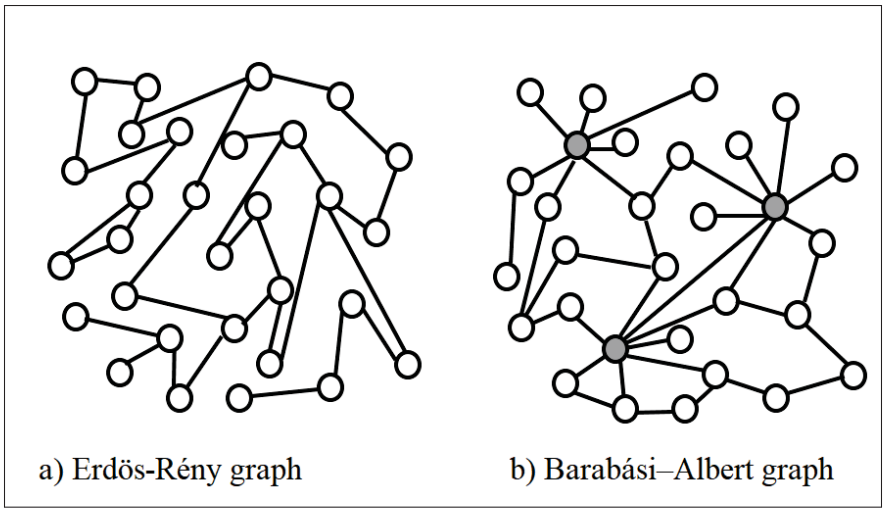


Figure 2.9 ER and BA graphs

2.1.11 Barabási–Albert model

Barabási–Albert (BA) is a random graph model with two defining features: *preferential attachment* and *power law degree distribution*. Networks with power-law degree distribution are also referred to as scale-free networks. Preferential attachment is also called the *rich gets richer*. This is because it is more likely for newly joining nodes to select peers with more connections, thereby increasing their already high number of connections. Hence, this peer selection strategy tends to produce networks where a limited number of hubs with a high degree can be found. A complete description of preferential attachment networks is presented in (Van Der Hofstad, 2016).

As a more formal definition, assume there exist N_0 initial nodes in the gradually growing network. Initially, these nodes are connected together as a complete graph with a degree of M_0. Suppose a new incoming node intends to establish m ($m \leq N_0$) connections to the existing nodes. The probability that this node will select node i as a peer can be estimated as follows:

$$p_i = \frac{m_i}{\sum_j m_j} \tag{2.4}$$

where m_i is the degree of the node i. In this model, nodes with the highest degree are themselves more likely to be connected to each other according to Equation 2.4. We propose BA graphs as a good model for public blockchain networks such as Bitcoin and Ethereum which employ a limited number of relay gateways, which use a high-speed backbone to efficiently transfer new blocks and transactions from one part of the network to another. A sample network generated using the Barabási–Albert model is depicted in Figure 2.9b.

2.2 Private blockchains: background on Hotstaff and BFT

In this section, we briefly introduce basic Hotstuff and present the the required background that is necessary to understand it. We study the Hotstuff protocol in order to develop a theoretical model for the performance analysis of the system.

Assume a network of connected nodes consists of two entities: *Clients* and *Replicas*. Replicas are the set of geographically distributed governor nodes that form the set of validators. Consider replicas are connected together in a fully connected P2P manner (i.e. every node is directly and independently connected to other $N - 1$ nodes). As well assume each client is connected to all of the replicas according to (Yin *et al.*, 2018) and can deliver its request (e.g. transaction) to the network of the replicas. Then, the client will wait for getting a response from at least $f + 1$ replicas. We assume replicas use a shared memory pool and can gossip received transactions together. Such a network is illustrated in Figure 2.10. Validators are responsible for maintaining the distributed ledger of programmable resources. Hence, each validator keeps a full copy of the ledger, while this is optional for clients. In some variants of Hotstuff such as Diem, validators can form a consortium consisting of a set of founding members (e.g. well-reputed central banks and large companies) which back the blockchain resources (e.g. cryptocurrency) with treasuries and cash deposits (bau).

The protocol operates in a succession of steps referred to as *views*. Views are numbered monotonically and increasingly. At the beginning of each view, one of the replicas is designated as the leader. The leader is known to all. Every replica store received requests in its local

memory. Normally, a consensus happens in one view. Each view consists of a sequence of phases: $prepare$, $pre-commit$, $commit$, and $decide$.

Prepare: protocol execution gets started with sending `new-view` message from the replicas together. The selected leader needs to gather $N - f$ `new-view` messages in order to ensure that it has been selected as the leader of this view. This threshold of $N - f$ is referred to as a quorum certificate (QC). Then the newly elected leader embeds its proposal in a `Prepare` message and broadcasts it to the replicas. Once the replicas receive the `Prepare` message, they execute it and prepare a `Prepare` vote and send it to the leader.

Pre-commit: after gathering QC `prepare` votes for the proposal, the leader prepares a `Pre-commit` message and broadcasts it to the replicas. This message also contains a `prepare QC` message and informs the replicas that QC `Prepare` votes are gathered. Replicas that receive this message, sign the digest of the proposal and embed it in a `Pre-commit` vote and send it to the leader.

Commit: similar to the previous phase, the leader needs to receive at least QC `Pre-commit` votes. Then, it prepares a `commit` message and broadcasts it to the replicas. This message contains a `Pre-commit QC` message and lets the replicas know that that QC `Pre-commit` votes are gathered. Once the replicas receive this message, they execute it and reply to the leader with a `Commit` vote.

Decide: this phase is quite similar to the two previous phases. In this phase, the leader waits until receives QC `Commit` votes. Then it generates a `Commit QC` and combines it with a `Decide` message and broadcasts it to the replicas. Once a replica receives this message, it assumes that the proposal is committed and executes the request(s) embedded in it. Then, every replica broadcasts a final `Decide` vote to the clients and increases view number by one.

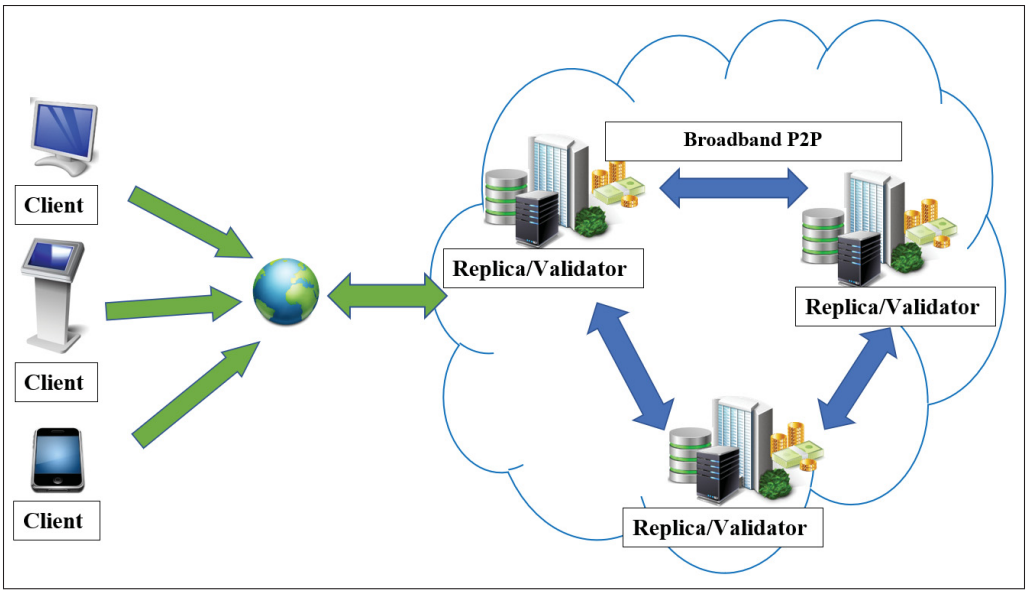


Figure 2.10 Overview of the proposed blockchain network validators are peers belonging to a set of replicas and can form an association. Each validator maintains a full copy of the blockchain distributed ledger

CHAPTER 3

A THEORETICAL MODEL FOR BLOCK PROPAGATION ANALYSIS IN BITCOIN NETWORK

In this chapter, we focus on the original and well-established Bitcoin blockchain network and propose a random graph model for performance modeling and analysis of the inventory-based protocol for block dissemination. This model addresses the impact of key blockchain parameters on the overall performance of Bitcoin. We derive some explicit and closed-form equations for block propagation delay and traffic overhead in the Bitcoin network. We also adapt our model to study the impact of deploying a relay network and investigate the effect of the relay network size on network performance and decentralization. We implement our model using the popular network simulator OMNet++. We validate the accuracy of our theoretical model and its implementation with our dataset mined from the Bitcoin network. Our results show the trade-off between the default number of connections per node, network bandwidth, and block size in order to compute the optimal block propagation delay over the network. As well, we found that bigger relay networks can jeopardize the decentralization of the Bitcoin network.

3.1 Introduction

Bitcoin suffers from a lack of scalability which may endanger the longevity of the cryptocurrency. In particular, the performance of Bitcoin is greatly impacted by P2P network parameters, such as throughput and information propagation delay.

In recent years, there have been many proposals to address the performance inefficiency of blockchain networks. From selecting nodes located in the geographical proximity of the miners as the next logical hop (Hao *et al.*, 2019; Owenson *et al.*, 2017), to implementing global relay networks (Corallo, 2019; Basu *et al.*, 2019; Klarman *et al.*, 2018) as well as efforts for optimizing the block propagation mechanism such as BIP 152 (Corallo, 2016b) and Graphene (Ozisik, Andresen, Bissias, Houmansadr & Levine, 2017), these solutions attempt to reduce information

propagation delay in the Bitcoin network. Therefore, a performance model would allow us to calculate the anticipated benefit of such solutions.

In this chapter, we focus on exploring the design space surrounding the original and most well-known blockchain system, Bitcoin (Nakamoto, 2009). We present a model for estimating compact block propagation delay and message exchanging traffic overhead for the Bitcoin data exchange protocol using inventory vectors. The contributions of this chapter are as follows:

1. We model the Bitcoin overlay network using an Erdös-Rény model (Erdős & Rényi, 1959) to generate connected random graphs. We take into account both legacy block propagation protocol and Bitcoin Improvement Proposal 152 (BIP 152).
2. We derive explicit mathematical equations to estimate important performance metrics, namely, block propagation delay and traffic overhead. Our extended model considers the long tail in the block outreach model.
3. We identify two important Bitcoin configuration parameters which impact performance: the average number of connections per node, and the block size.
4. We implement our theoretical model using the network simulator OMNet++, as a discrete event simulator.
5. We validate our model and our simulation using our dataset mined from the Bitcoin network. Our results show the sensitivity of block propagation delay with various Bitcoin parameters.
6. We estimate the probability density function (PDF) of the rate at which the participating nodes receive the propagating block.
7. We investigate the impact of deploying relay networks on the aforementioned performance metrics.

The rest of this chapter is organized as follows. In Section 3.2, we present our analytical model using a random graph network and capture the Bitcoin network behavior and dynamics. In Section 3.3, we compare the results of our analytical model with simulation results as well as empirical amounts mined from the Bitcoin network in order to validate the accuracy of the model. In Section 3.3.2, we leverage our theoretical model for performance analysis of the

Bitcoin network. In Section 3.4, we discuss the managerial implications of this work. Finally, Section 3.5 concludes this chapter.

3.2 Analytical Model

In this section, we propose a random graph model for performance modeling and analysis of the Bitcoin network. To accomplish this, we use a graph model which was introduced by Erdös and Rényi. This graph has properties suitable for modeling peer-to-peer overlay networks used by blockchain systems.

In this work, we define an overlay network as a graph $G(V, L)$ where V is the set of vertices and L is the set of links between nodes. For example, if there is a link between node i and node j, then $(i, j) \in L$.

Furthermore, we represent a random graph using $G_p(N)$, where N is the total number of nodes and p is the independent probability that there exists a link between any two selected nodes in the peer-to-peer overlay network. In this work, we assume that N is significantly large, as it is in the Bitcoin network ($N \approx 10000$).

3.2.1 Random graph construction

To construct our random graph representation of the overlay network, we first start with a single node, which we call the initiator node, denoted by n_0. Then, a second node n_1 enters the system and establishes a link with n_0 with probability p. Naturally, this means the probability of not having a link is $1 - p$. After this, a third node n_2 enters and can potentially connect n_0 or n_1, both with probability p. This process continues until node n_{N-1} enters the system and potentially establishes links to the $N - 1$ existing nodes under the same probability p.

However, the random graph construction is an artificial mechanism to generate a topology of a fixed size and does not reflect how the P2P system evolves in reality. Another way to interpret the random graph construction is as follows: first generate the total number of nodes (e.g., 10000

nodes) as vertices in the graphs. Then, generate the set of all possible edges between any pair of vertices (called F). Finally, choose a subset of edges to be included in E, such that $\frac{|E|}{|F|} = p$. In this way, we can obtain a random graph with the desired number of nodes and the correct link probability p. In other words, our model only calculates performance values for the steady state (when the graph is fully constructed), and not the transient state of nodes joining.

It is obvious that for $p = 0$, we have an empty graph (all nodes are isolated with no edges) and that for $p = 1$ we have a complete graph with a degree of $N - 1$ for each node. In general, our random graph model has several characteristics: (i) if $p > \frac{1}{N}$ then a giant component exists with high probability (w.h.p.). (ii) for $p \geq \frac{\log(N)}{N}$ all nodes become a part of the giant component and $G_p(N)$ becomes a connected graph.

These connectedness properties are vital for a blockchain network to operate properly. With a sufficient p value, each node has likely more than one path to reach any other node, using different outgoing links. In such a network, removing one link does not cause a network partitioning as the entire system stays connected. Network partitions are important to avoid in a blockchain network, since they guarantee that blockchain forks cannot resolve as long as the partitions stay isolated, thus compromising the integrity of the data on the ledger.

3.2.2 Achieving the connectedness properties

We now show how we can satisfy the above connectedness properties (i.e., obtaining a sufficiently high value of p) by appropriately configuring blockchain parameters.

Consider a connected blockchain network consisting of a set $\mathcal{N}=\{n_0,n_1,...,n_{N-1}\}$ of N participating nodes. For convenience, assume that all links between nodes have the same point-to-point bandwidth of B. Each node maintains on average M open connections to other nodes. Since the network is connected, the average number of links between nodes can be calculated as follows:

$$\overline{L} = \frac{pN(N-1)}{2} \tag{3.1}$$

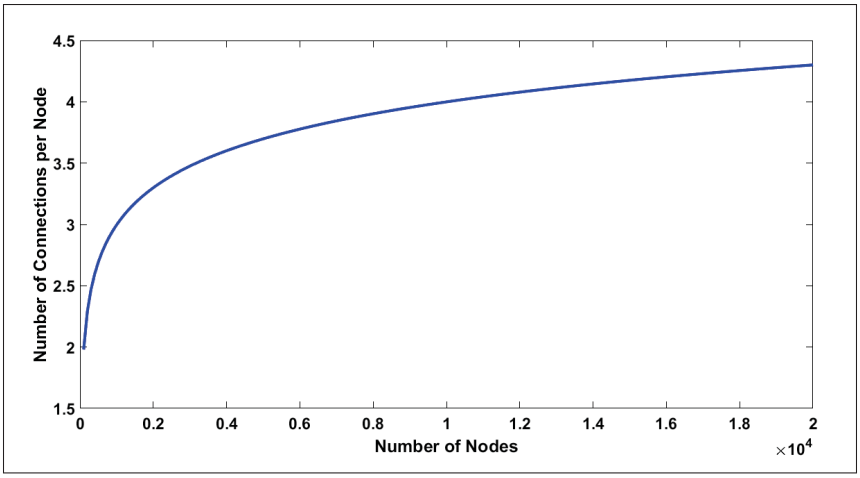


Figure 3.1 Minimum number of connections for connectedness

As well, in a connected random graph, the average degree of each node equals to $p(N-1)$ (i.e., $M = p(N-1)$). Hence,

$$p = \frac{M}{N-1} \tag{3.2}$$

The above equation enables us to approximate p when we have the average number of default connections for each node. As well, according to Equation (4.5), given the total number of participating nodes in a random graph, we can calculate the minimum degree or the number of connections per node in order to have a connected graph with high probability as follows:

$$M > \frac{N-1}{N} log(N) \tag{3.3}$$

In other words, to form a connected graph with high probability, it is sufficient that:

$$M \geq \lceil \frac{N-1}{N} log(N) \rceil \tag{3.4}$$

Figure 3.1 illustrates the relationship between the total number of participating nodes in the graph and the minimum number of connections required to form a connected graph. According to this figure, since the Bitcoin network has approximately 10,000 participating nodes, if each individual node maintains at least 5 connections to other nodes, the network should be connected with high probability.

3.2.3 Legacy protocol block dissemination analysis

In this section, we present a theoretical model for block propagation delay in the legacy propagation protocol. Although this protocol is deprecated for the current Bitcoin network, we include this analysis for the sake of completeness and to demonstrate that this model serves as a foundation that can be generalized for other blockchain protocols.

Assume a P2P overlay network that consists of N participating nodes. Suppose one miner node mines a block at time $t = 0$. We denote this initial node by n_0. According to Section 2.1.7, node n_0 sends an *inv* message to the set $\mathcal{W}_1 = \{n_1^1, n_2^1, ..., n_M^1\}$ of neighboring nodes in its connection pool. We assume the sending node sends the *inv* message to its neighbors in succession with a very small delay ϵ between each message. Since this is the first time that the *inv* message is being sent for this block, none of the $\mathcal{W}_1$ neighboring nodes have the block and will respond the *inv* message with the $getdata$ message. Then, the node n_0 will send the complete block to the requesting neighbors. We call this first step of the block dissemination process $Wave$ 1, as illustrated by Figure 3.2a.

Upon completion of $Wave$ 1, all involved nodes during this initial wave (i.e., nodes which received the block during the first wave) validate the block and send an *inv* message to the neighboring nodes in each of their own connection pool. We call this $Wave$ 2, as illustrated by Figure 3.2b. For convenience, we assume nodes do not know or remember the initial node n_0, but they do remember the node which they received the block from and will not send an *inv* message back to the sender.

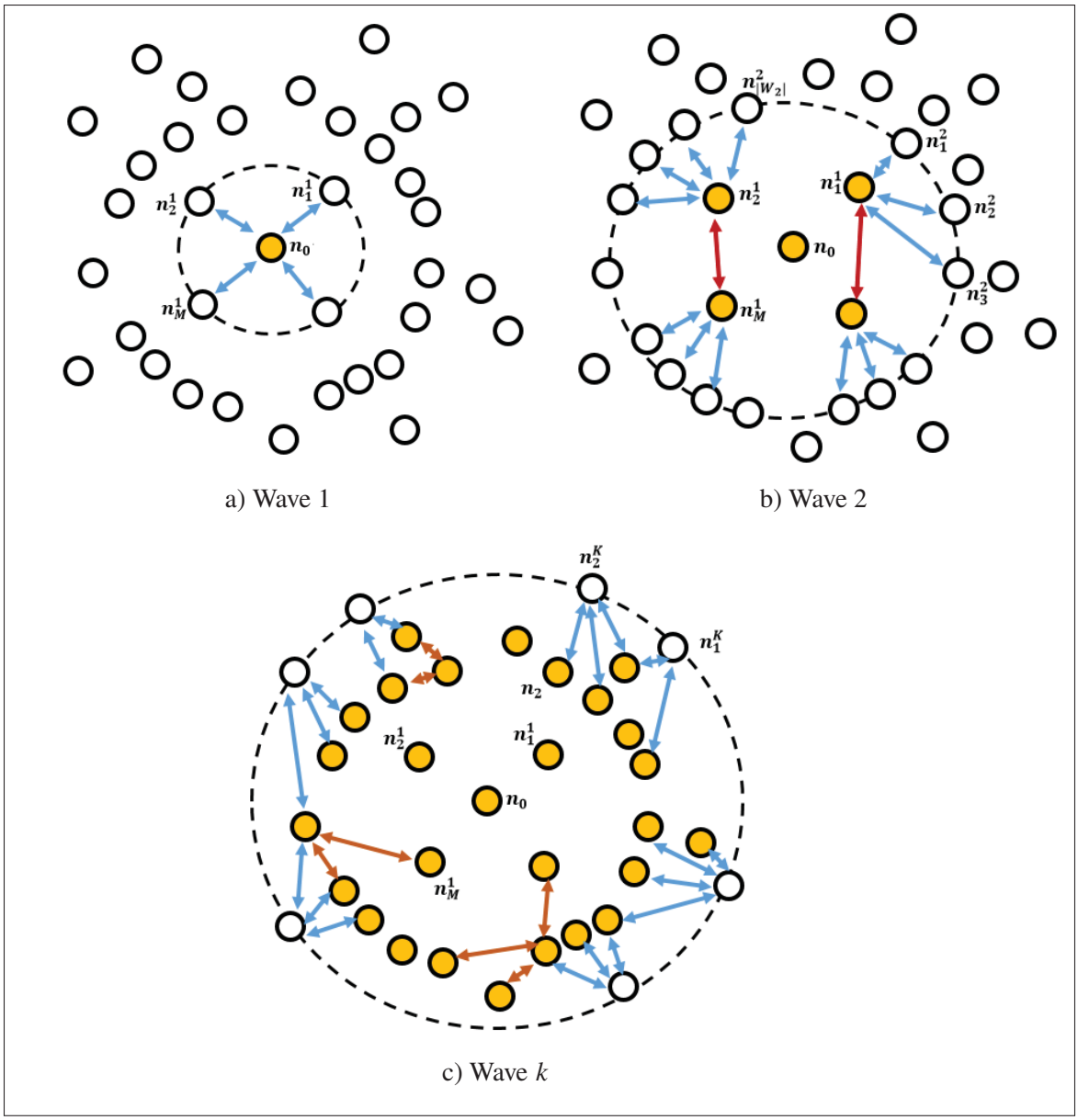


Figure 3.2 Different waves of block dissemination in the proposed blockchain network Colored nodes are the nodes which have already received the block. Blue arrows show successful block transfers. Red arrows show timed out *inv* messages

We observe that some of the nodes receiving an *inv* message during $Wave$ 2 may have already received the block in $Wave$ 1 and will not respond to the *inv* message with a $getdata$ message. Therefore, the forwarding probability, that is the probability that a node replies affirmatively to

the *inv* message with a $getdata$ message immediately after ending the $Wave$ 1, can be calculated as follows:

$$p_{f2} = \alpha \frac{N - 1 - |\mathcal{W}_1|}{N - 1} \tag{3.5}$$

Where $0 \leq \alpha \leq 1$ is the reachability factor of the nodes in the network. We use this factor to take into account the connections lost during the three-way message exchange mentioned in Section 2.1.7 or due to message loss (e.g., congestion, link outage, and etc.). For instance, if $\alpha = 0.95$, this means that 95% of the messages on average will reach the destinations during the three-way inventory message exchange.

Accordingly, the set of receiver nodes which send back a $getdata$ message can be represented as: $\mathcal{W}_2 = \{n_1^2, n_2^2, ..., n_{|\mathcal{W}_2|}^2\}$, where $|\mathcal{W}_2| = \lceil p_{f1} p_{f2} M^2 \rceil$.

Each subsequent wave will follow the same pattern as $Wave$ 2. Figure 3.2c shows an example for some $Wave\ k$, where only the nodes who received the block during the previous $Wave\ k-1$ will send *inv* messages during the current wave.

In general, for each wave, we define forwarding probability p_{fi} as follows:

$$p_{fi} = \alpha \frac{N - 1 - \sum_{j=0}^{i-1} |\mathcal{W}_j|}{N - 1} \ (i \geq 1 \ and \ |\mathcal{W}_0| = 0) \tag{3.6}$$

where $|\mathcal{W}_j| = \lceil M^j \prod_{k=1}^{j} p_{fk} \rceil$. The numerator of the equation above gives the number of nodes which have not received the block at the beginning of $Wave\ i$. This means during $Wave\ i$, the nodes which receive the *inv* message will respond to it with a $getdata$ message with probability p_{fi}. Accordingly, the *inv* message will be timed out with probability $1 - p_{f_i}$. Note that $p_{f1} = 1$ since all the nodes which are connected to the initial node n_0 will receive the block during the first wave.

We define parameter C_i as the partial coverage which means the coverage of nodes during the $Wave\ i$ (i.e., the number of nodes which receive the block during $Wave\ i$) as

$$C_i = |\mathcal{W}_i| = \lceil M^i \prod_{j=1}^{i} p_{fj} \rceil \tag{3.7}$$

As well, we define cumulative coverage as follows:

$$C_i^T = \sum_{j=1}^{i} |\mathcal{W}_j| = \sum_{j=1}^{i} \lceil M^j \prod_{j=1}^{i} p_{fj} \rceil \tag{3.8}$$

Where C_i^T is the total number of nodes that have received the block at the end of $Wave\ i$.

The set of receiver nodes which send back $getdata$ message after receiving inv message during $Wave\ i$ can be represented as $\mathcal{W}_i = \{n_1^i, n_2^i, ..., n_{|\mathcal{W}_i|}^i\}$. According to Equation (4.7), p_{fi} can be calculated as follows:

$$p_{fi} = \alpha \frac{(N-1) - \sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{fk}}{N-1} \ (1 < i \leq K) \tag{3.9}$$

Where K is the total number of waves needed for a block to be distributed among all nodes in a connected peer-to-peer overlay network. Note that the above equation is valid as long as $C_i^T \leq N-1$. For the case where $C_i^T > N-1$, we need to adapt and reform Equation (4.8) as follows:

$$p_{fi} = \beta\alpha \frac{(N-1) - \sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{fk}}{N-1} \ (1 < i \leq K) \tag{3.10}$$

where β is the adaptation factor. In fact, Equation (3.10) can be used for calculating all forwarding probabilities where β is expressed as follows:

$$\beta = \begin{cases} 1 & C_i^T \leq N-1 \\ \frac{N-1-C_{i-1}^T}{C_i} & C_i^T > N-1 \end{cases} \tag{3.11}$$

The Equation (3.10) is recursive and enables us to calculate the number of waves required for global outreach of a block (x% block propagation):

$$\sum_{i=1}^{K} M^i \prod_{j=1}^{i-1} p_{fj} = x(N-1) \tag{3.12}$$

After the calculated number of waves have occurred, if no conflicting blocks were found during the entire dissemination process of the current block, we can guarantee that forks resulting from network delays or partitions cannot occur for this block. Note that malicious attacks (such as 51% attacks) to intentionally introduce forks in the blockchain are out of the scope of this model.

Algorithm 5.1 shows how we can obtain the number of required waves for fully disseminating a block, as well as the forwarding probability in each wave (Equation (3.10)). This algorithm computes forwarding probabilities in a recursive manner.

Consequently, key features and performance measures can be derived in terms of block dissemination delay and traffic overhead. We calculate block dissemination delay as follows:

$$D = K(D_v + X_I + Y_I + D_g + X_G + Y_G + D_b + X_B + Y_B) \tag{3.13}$$

3.2.4 Performance metrics analysis of the legacy protocol

where D_v is the block validation time, X_I is the transmission delay, and Y_I is the signal propagation delay of *inv* messages. D_g is the time interval taken by a node to process an *inv* message (i.e., lookup if the content of the message are present locally or not) before replying to the *inv* message. As well, X_G and Y_G are the transmission delay and propagation delay of the

getdata message. Similarly, D_b is the time interval taken to process a *getdata* message before replying with the requested block. Similar to previous ones, X_B and Y_B are the transmission delay and average propagation delay of the sent block, respectively. For convenience, we assume that Y_I=Y_G=Y_B.

Given the bandwidth B for each link of the random graph, we can rewrite Equation (4.10) as follows:

$$D = K(D_v + \frac{S_i}{B} + Y_I + D_g + \frac{S_g}{B} + Y_G + D_b + \frac{S_b}{B} + Y_B) \tag{3.14}$$

where S_i, S_g, and S_b are the sizes of *inv* message, *getdata* message, and the transmitted block, respectively.

Moreover, the network traffic overhead for *Wave i* can be calculated as follows:

$$H_i = \frac{(1 - p_{fi})M^i \prod_{j=1}^{i-1} p_{fj}}{N - 1} \ (1 \leq i \leq K) \tag{3.15}$$

The overall packet exchanging traffic overhead is the sum of the traffic overheads in each wave.

$$\overline{H} = \frac{1}{N - 1} \sum_{i=1}^{K} [(1 - p_{fi})M^i \prod_{j=1}^{i-1} p_{fj}] \tag{3.16}$$

Note that the packet exchanging traffic overhead reveals the burden of sending redundant *inv* messages in later waves to nodes that have already previously received the block. Since Bitcoin is an unstructured and decentralized P2P network that relies on gossiping, there is no coordination that allows nodes to efficiently decide which neighbors are likely to have received the block already without at least contacting them with an *inv* message.

Also note that for this model, we focus principally on the dissemination of a single block at a time. An *inv* message can also be used to exchange multiple blocks at the same time. However, we argue that the single block use case, as discussed here, is much more common in Bitcoin due

to the long block time of 10 minutes, which diminishes the chance of requiring multiple blocks to be disseminated in the same *inv* message.

Algorithm 3.1 Calculating the number of waves

Input : The values of N, M
Output K and Matrix $\mathcal{P}[p_{f1}...p_{f_K}]$
:

$\mathcal{P}[1] \leftarrow 1$
$K \leftarrow 2$; /* We assume M<N-1 and hence K≥ 2 ∗ /
$C[1] \leftarrow M$
$Ctrl \leftarrow 1$; /* Controller */
while $Ctlr = 1$ **do**
  for $i = 2$ *to* K **do**
    $\mathcal{P}[i]=\frac{(N-1)-\sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} \mathcal{P}[k]}{N-1}$
    $C[i]= M^i \prod_{j=1}^{i} \mathcal{P}[j]$
    if $\sum_{j=1}^{i} C[j] \leq N-1$ **then**
      $K = K + 1$
    else
      $\mathcal{P}[i]=\frac{N-1-\sum_{j=1}^{i-1} C[j]}{C[j]} \times \frac{(N-1)-\sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} \mathcal{P}[k]}{N-1}$
      if $\mathcal{P}[i] \geq 0$ **then**
        $K = K + 1$
        $C[i]= M^i \prod_{j=1}^{i} \mathcal{P}[j]$
      else
        $Ctrl \leftarrow 0$
      end if
    end if
  end for
end while
return K *and* $\mathcal{P}$

3.2.5 Compact block protocol and block propagation analysis

In this section, we present an analytical model for block propagation delay using the compact block propagation protocol of Bitcoin. As already shown in Figure 3.3, the compact block protocol operates in two modes of operation: low bandwidth (LBM) and high bandwidth (HBM).

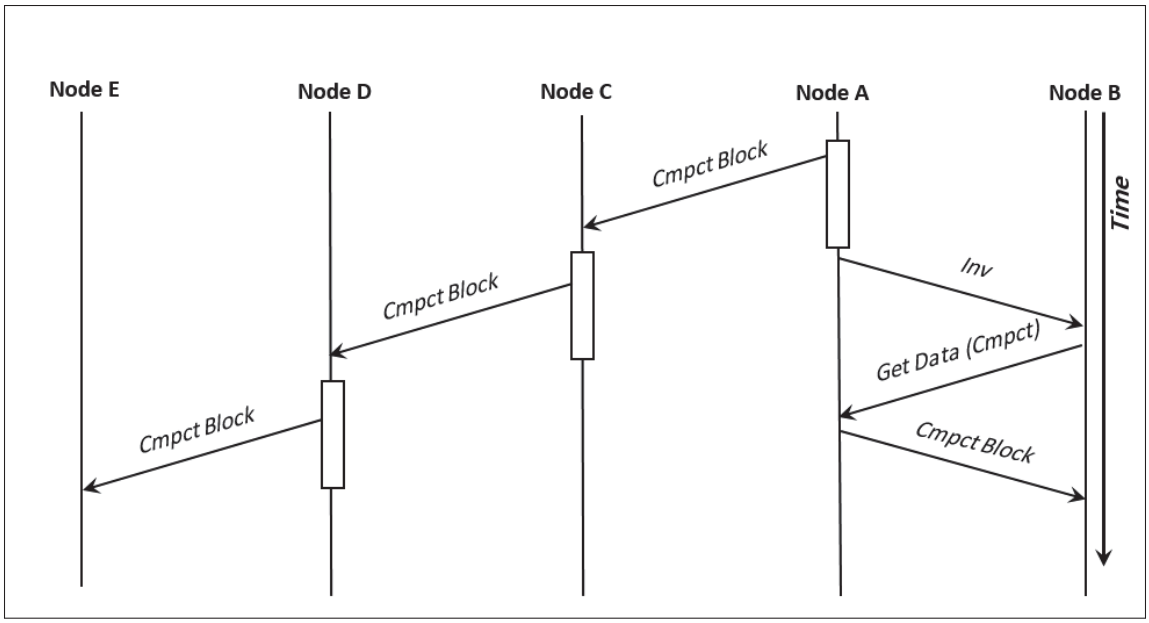


Figure 3.3 An example to show the block propagation time in different modes of operation Node A has established LBM connection to node B as well as an HBM connection to node C. In the same way, node C has established an HBM connection to node D, and node D has established an HBM connection to node E. Nodes B and E receive the block at almost the same time

Each node in the network establishes regular connections to some peers in its connection pool using LBM and tells a subset of the remaining nodes to push newly arrived blocks (i.e., by sending *inv* message). Thus, LBM operates similarly to the legacy protocol. The main differences are the size of full blocks compared to the compact ones and the two additional messages exchanged in case of block reconstruction failure (*getblocktxn* and *blocktxn*). In this section, we assume the block reconstruction failure rate is very low and negligible[1].

Given a P2P overlay network with N participating nodes, suppose each node has established on average M LBM connections and m HBM connections (i.e., m peer nodes will push compact blocks using a *sendcmp* message). We assume that all links between nodes have the same P2P bandwidth of B. Since block dissemination using LBM is similar to block dissemination in the legacy protocol, we can use the concept of waves for LBM as we did in Section 3.2.3.

[1] See the presentation by core developer Greg Maxwell: https://people.xiph.org/~{}greg/gmaxwell-sf-prop-2017.pdf

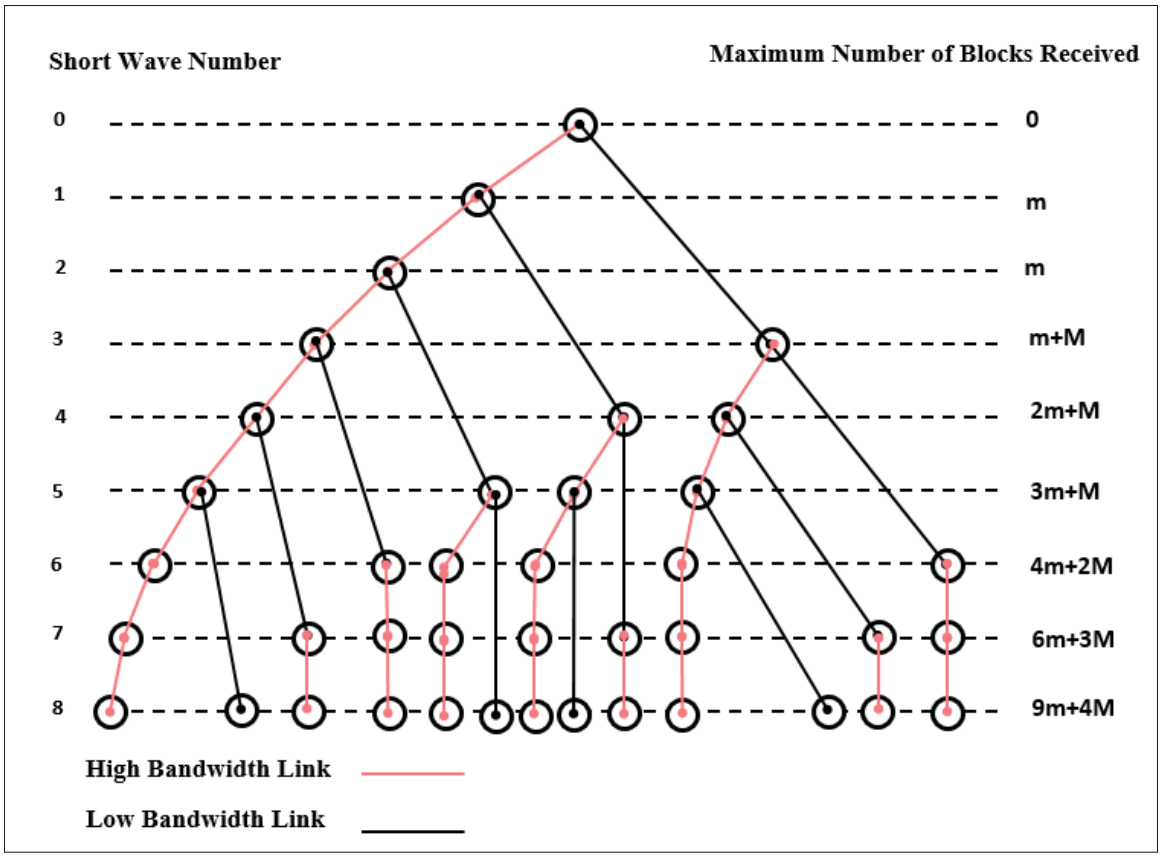


Figure 3.4 Long waves and short waves in block propagation
Each long wave equivalents three short waves. Red links are the links operated in HBM and black ones are the links operated in LBM

But for HBM, we adapt our model by introducing the concepts of long waves and short waves. Each long wave involves three message transfer as required to deliver the compact block as shown in the Figure 3.3. As well, each short wave involves just a compact block transmission. Therefore, each long wave time duration is essentially three times the duration of a short wave as depicted in Figure 3.3. In other words, one block transfer in LBM equals three block transfers in HBM, since the size of the transferred packets is comparatively small and has a negligible transmission delay compared to the signal propagation delay. Since we have assumed an equal signal propagation delay for all links, this assumption is reasonable.

Suppose n_0 is the initiator node that intends to disseminate a newly mined block. It will send the compact block to M peers in LBM and to m nodes in HBM. At the end of the first short

wave, m nodes will receive the compact block and immediately will start sending the compact block to m of their already selected peers in HBM and to M of the remaining peers in LBM. According to Figure 3.4, at the end of the second short wave, at most $2m$ nodes will have the new block. We call the number of nodes that receive a block during a short wave i as the *wave coverage* and denote it by C_i. Also, we use the term *total coverage* for the total number of nodes that have received the block at the end of short wave i and denote it by N_i. At the end of the third short wave, the first set of the nodes which are connected to the initiator node in LBM will receive the compact block (M nodes). As well, m more nodes operating in HBM will receive the block at the end of the third short wave (See Figure 3.4). Therefore, the coverage of the third wave and total coverage of the third wave will be at most $m + M$ and $3m + M$ nodes, respectively. According to the above, the coverage of each wave is calculated as follows:

$$C_i = X_i m + Y_i M \tag{3.17}$$

X_i and Y_i are non-negative integers and can be calculated as follows:

$$\begin{aligned} X_1 &= 1 \\ X_i &= X_{i-1} + Y_{i-1} \; i \geq 2 \end{aligned} \tag{3.18}$$

and

$$Y_i = \begin{cases} 0 & 1 \leq i \leq 2 \\ 1 & i = 3 \\ X_{i-3} + Y_{i-3} & i \geq 4 \end{cases} \tag{3.19}$$

According to the above, $C_1 = N_1 = m$. Thus, we can estimate the total coverage at the end of the second short wave as follows:

$$N_2 = N_1 + (1 - \frac{N_1}{N-1}\alpha)m \tag{3.20}$$

where $\frac{N_1}{N-1}$ is the fraction of the nodes that have received the block so far. Accordingly, $1 - \frac{N_1}{N-1}$ yields the probability that receiving nodes have not already received the block and therefore will accept it. Consequently,

$$N_k = N_{k-1} + [(1 - \frac{N_{k-1}}{N-1}\alpha\beta)(X_k m + Y_k M)] \tag{3.21}$$

where N_k is the total coverage at the end of the short wave k. Finding the maximum amount of N_k that satisfies the condition $N_k \leq N - 1$ will give an estimation of the block propagation delay based on the number of short waves.

$$\beta = \begin{cases} 1 & N_k \leq N-1 \\ \frac{N-1-N_{k-1}}{N_k} & N_k > N-1 \end{cases} \tag{3.22}$$

The number of short waves required for global outreach of the compact block can be estimated as follows

$$K = \{min\ k | N_k \geq N - 1\} \tag{3.23}$$

3.2.6 Performance metrics analysis of BIP152

The compact block dissemination delay can be estimated as follows:

$$D = \lfloor \frac{K}{3} \rfloor (D_p + X_I + Y_I + X_G + Y_G + X_B + Y_B) + (K - 3\lfloor \frac{K}{3} \rfloor)(X_I + Y_I) \tag{3.24}$$

Where D_p is the processing delay of the compact block and consists of block validation time and internal processing delay in nodes.

3.2.7 Assumptions

Our analytical solution models block propagation in several waves. This means all nodes receive and release blocks from/to their neighbors at the same time. As well, it assumes the latency numbers for each pair of nodes. In reality, the latency between peers depends on different factors such as geographical distance between peers and the bandwidth of the links. Moreover, block validation time varies in different nodes depending on their computational power. Although the network is overall asynchronous, we argue that the propagation of a single new block can be approximated as a synchronous process. Furthermore, Bitcoin has only been proven to be correct in a partially asynchronous system (Pass, Seeman & Shelat, 2017), which supports our assumption. Note that in a real system, different configuration parameters are tightly tangled together such that a theoretical model of the system becomes intractable. Hence, our model aims to simplify to a certain degree while still providing approximate, yet accurate results. Thus, we mostly rely on the average values of the metrics. Despite of these assumptions, the results of our model closely match the results of the data mined from the real network of Bitcoin.

3.3 Evaluation

In this section, we first validate our theoretical model by comparing the theoretical results with simulation results as well as the values of the empirical data mined from the real Bitcoin network.

We then conduct a performance analysis to assess the impact of various parameters on the performance of the Bitcoin network. We study the impact of the average number of connections per node, the network size and the network bandwidth, and the reachability factor on the performance of the Bitcoin network. We also measure the network traffic overhead for different number of connections per node. Finally, we investigate the impact of deploying a relay network

on the system performance and study the overall performance of the network when modifying other parameters.

Algorithm 3.2 Calculating the number of short waves

Input : The values of N, M, m
Output K and Matrix $\mathcal{N}[N_1,..., N_K]$
:

$\mathcal{N}[1] \leftarrow m$
$\mathcal{X}[1] \leftarrow 1$
$\mathcal{Y}[1] \leftarrow 0, \mathcal{Y}[2] \leftarrow 0, \mathcal{Y}[3] \leftarrow 1$
$C[1] \leftarrow m$; `/* We define this matrix for the coverage of each short wave */`
while $\mathcal{N}[i] \leq 0.9N$ **do**
    $\mathcal{X}$[i+1]= $\mathcal{X}$[i]+$\mathcal{Y}$[i]
    $\mathcal{Y}$[i+3]= $\mathcal{X}$[i]+$\mathcal{Y}$[i]
    C[i+1]= $m\mathcal{X}$[i+1]+$M\mathcal{Y}$[i+1]
    $\mathcal{N}$[i+1]= $\mathcal{N}$[i]+ C[i+1]$(\frac{\mathcal{Y}[i+1]}{N-1})$
    $i = K$
end while
return K *and* $\mathcal{N}$

3.3.1 Settings and implementation

Implementation: We implemented a discrete event-based simulation using $Omnet++$ (OmNet). We developed a *C#* code to automatically get the data from the provided application program interface (API). We also used $Matlab$ for theoretical analysis.

Dataset: We compare numerical values of our theoretical model and simulation results to real empirical measurements provided by (bit, 2018). In this data set, 90% denotes 90% of inv messages for a block were observed within the given time from the first 1000 nodes. We extracted the data of almost 15000 Bitcoin blocks from block number 507016 through 522429 in the main chain. These block numbers correspond to blocks generated from February 1, 2018, until June 12, 2018. These blocks comprise different number of transactions embedded in them and hence have different sizes.

Parameters: The block processing time (i.e, block validation time plus other internal processing delays) depends on the block size. After observing the block propagation delay data reported by (bit, 2018), we infer that the average block propagation delay increases sub-linearly with the increase of block size. Since the impact of increasing the compact block size on the propagation delay (X_B or compact block transmission delay) is negligible, we can conclude that the processing delay increases sub-linearly with increasing the block size. Hence, we estimated an average time of $4ms$ for processing of a block (around $1ms$ for block validation) with a size of 0.1 MB (1800 KB size of the compact block) and extrapolated it for other blocks with different sizes.

Although different nodes in Bitcoin use different links with various bandwidths, it has been shown in (Gencer *et al.*, 2018) that the median provisioned bandwidth of nodes in the Bitcoin network was 33 $Mbps$ and 56 $Mbps$ for early 2016 and early 2017, respectively. The latter is thus 1.7 times the first one. Accordingly, the average bandwidth for nodes that use IPv4, IPv6, and Tor addresses are reported as 73.1 $Mbps$, 86.5 $Mbps$, and 4.7 $Mbps$, respectively. We also observed that almost 83%, 13%, and 4% of nodes in the Bitcoin network during the time interval related to our data set used IPv4, IPv6, and Tor addresses, respectively. Using a weighted mean and multiplying it by 1.7 in order to scale to early 2018, we estimated provisioned bandwidth as 123 $Mbps$ in early 2018. However, due to the small size of compact block, this parameter becomes less important and the compact block transmission delay can be neglected.

In (Decker & Wattenhofer, 2013), it has been stated that a $bitcoind$ node which accepts incoming connections, has an average of 32 connections which is much more than the default number of 8 connections for each node. In this experiment, we thus considered M=32 for theoretical analysis and simulation.

To estimate the average signal propagation delay in the Bitcoin network, we consider blocks with the smallest size in the data set ($S_b < 1kB$) and obtained 20 ms as the average value for signal propagation delay.

We also consider the same size of 37 bytes for inv and $getdata$ messages according to (Biryukov, Khovratovich & Pustogarov, 2014). Also, in the theoretical analysis and simulation, we assume

that there is no packet loss and consequently, no packet re-transmission. In all of the experiments, we consider a network size of 10,000 nodes, which is near the Bitcoin network size.

To generate an Erdös-Renyi random graph with the two parameters of N and p, we used the network description (NED) language in $Omnet++$. To fix the average number of connections per node, we counted the number of inv messages in the simulation log file using awk and divided them by N. This approach is sound because each node sends inv messages to M neighboring nodes in his connection pool.

The results of the simulation and theoretical analysis are depicted in Figure 3.5. In this figure, the horizontal axis indicates the block size where each point on this axis represents a range of block sizes with a length of 0.1 MB. For instance, block size = 0.2 represents the interval between 0.1 MB and 0.2 MB. The box plot shows the propagation delay of blocks in the real Bitcoin network versus block sizes. We removed outliers to clean up the graph. Bitcoin Core with BIP 152 recommends peers establish their last three connections in HBM. Therefore, we did the experiments with this value ($m = 3$). However, we have also tested other values, such as the case with no HBM connections ($m = 0$). For $m = 1, 2$, results fall between the results of the mentioned experiments. As it can be seen, the results of simulation and theoretical values are almost identical with high accuracy with respect to empirical results. By calculating the slope of the graph (Figure 3.5) at different points, the results show that block propagation delay increases sub-linearly with respect to block size. For the smaller blocks, the size of the quartiles are comparatively smaller. The main reason is that in small block sizes, the delay stems from the signal propagation delay. But for big blocks, the delay originates from the block processing time.

3.3.2 Performance analysis of the Bitcoin network

To study the effect of the default number of connections on block propagation delay, we conduct an experiment using our theoretical model for different values of M (M = 8, 16, 32, 64). As well we assume no connection is set in HBM (m=0). $M = 8$ refers to the default number of connections in the original protocol whereas $M = 32$ refers to the average number of connections

per node observed in practice. $M = 64$ and $M = 16$ can be considered proposals for doubling or halving the average number of connections per node in the current Bitcoin network. Like in the previous experiment, we use N=10000. Figure 3.7 depicts the results of this experiment. The network with $M = 8$ is significantly slower than others and takes five waves to fully disseminate the block. While the network with $M = 64$ is considerably faster than networks with $M = 16$ and $M = 8$, there is no significant differences between the networks with $M = 64$ and $M = 32$, since both fully disseminate the block in two waves. This means that increasing the average number of connections in the Bitcoin network from 8 to 32 significantly increases the block propagation speed. In all settings evaluated, the block does not reach a majority of nodes (5100 nodes) before the last wave. Dividing the values obtained in this experiment by the total number of nodes in the network will yield similar curves for the cumulative distribution function (CDF) of the block outreach.

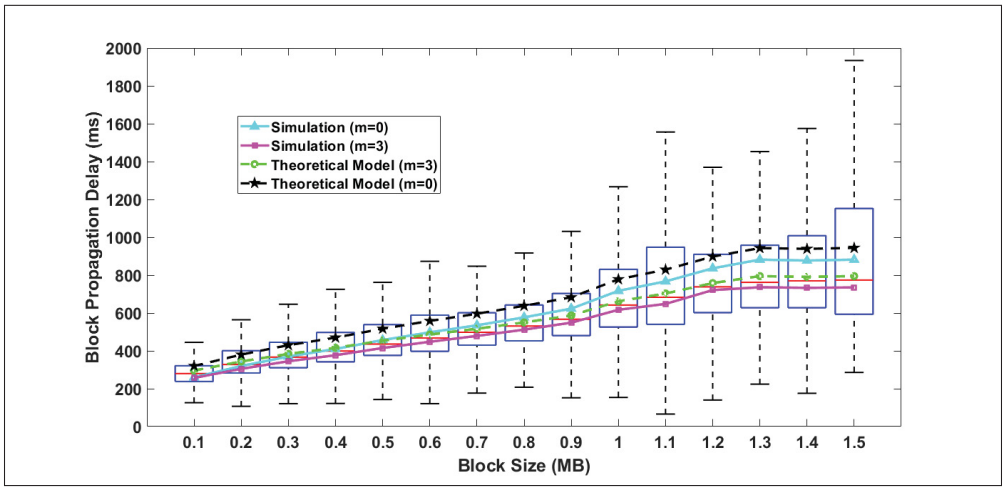


Figure 3.5 90% Block propagation delay vs. block size

To study the impact of the average number of connections per node on the traffic overhead, we carry out another set of experiments for N=7500, 10000, and 12500. The results of this experiment are depicted in Figure 3.6. As can be seen in the figure, the optimal traffic overhead is for M=32. This means that this configuration achieves a minimal number of redundant *inv* messages (*inv* messages which timed out). This can be explained by the fact that in a network

with a lower M, each node sends fewer messages to neighbors and according to Equation (4.8), a bigger forwarding probability is achieved. On the other hand, an increased number of connections creates a faster network with a lower number of waves, which leads to an increase in total traffic overhead. Therefore, there is a trade-off between block propagation delay and network overhead.

According to Figure 3.6, the other choice for low traffic overhead can be $M = 8$. But according to Figure 3.7, this will create a slow network. Thus, we claim that $M = 32$ is the best choice for the current Bitcoin network.

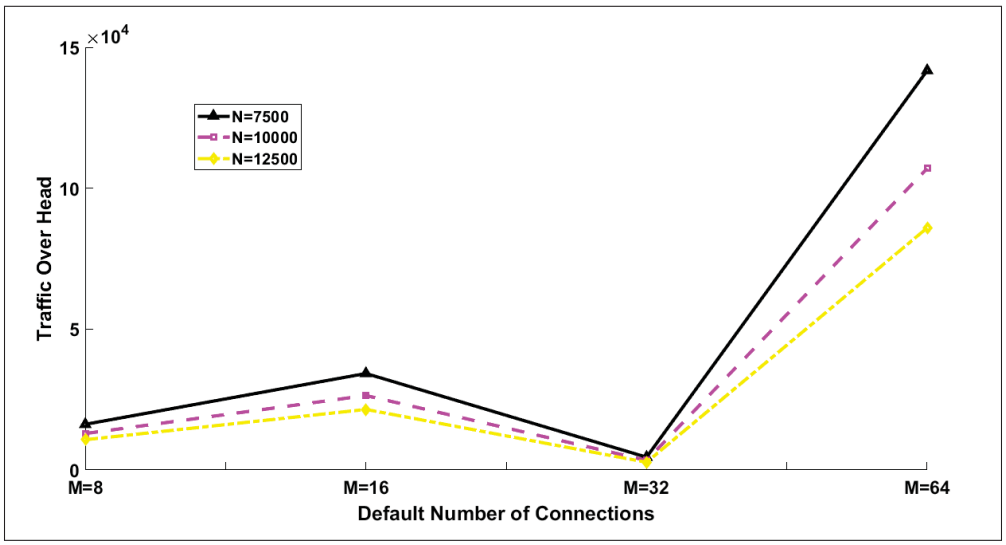


Figure 3.6 Network traffic overhead (90%)

Another point of interest is the impact of network size or the number of participating nodes on the average block propagation delay. To study this, we conduct another experiment and calculate the average block propagation delay (90%) for a different number of nodes (intervals of 1000 nodes) and repeat it for different number of connections. The results are depicted in Figure 3.8. Increasing the number of connections will not always lead to a faster network. For the current size of Bitcoin, we expect roughly the same propagation delay when $M = 16, 32, 64$ but with different costs of traffic overhead. For smaller sizes, $M = 64$ is significantly faster while there is no considerable change for the networks with $m = 16$ and $M = 32$. For sizes bigger than that of the current network, $M = 32$ and $M = 64$ will have the same impact on the block propagation

delay with different amounts of traffic overhead. It is unlikely to have less than 5000 or more than 13000 nodes in the Bitcoin network. Therefore, we limit the evaluated intervals to 4000 – 5000.

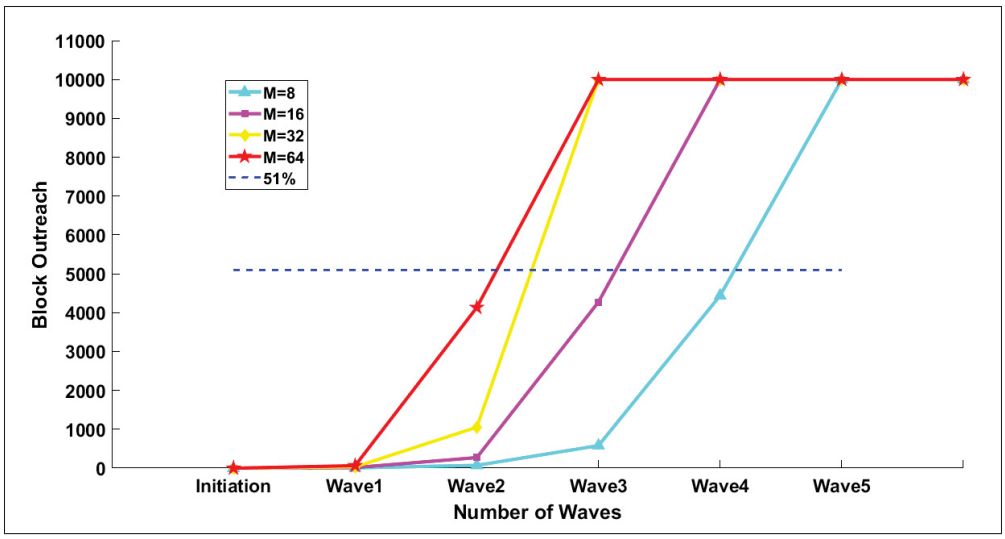


Figure 3.7 Block outreach vs. number of waves (90%)

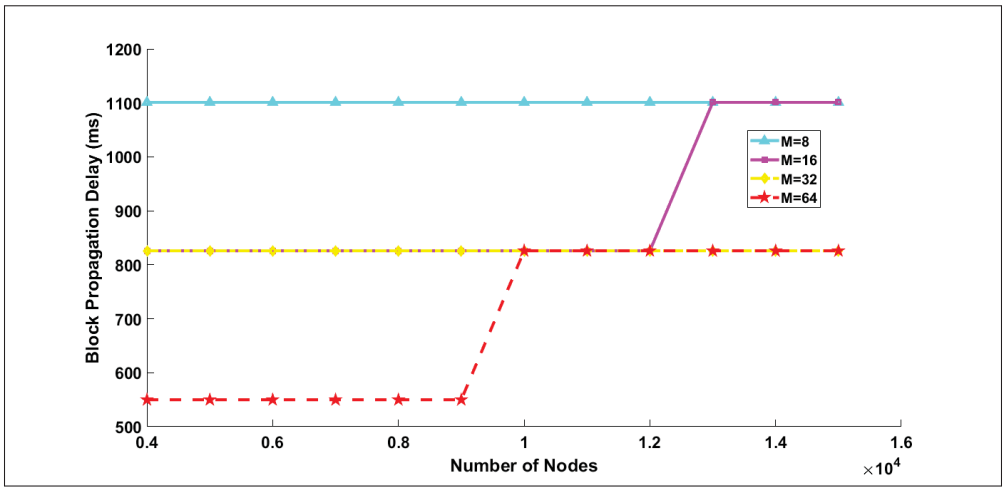


Figure 3.8 90% block propagation delay vs. network size

It is useful to study the Bitcoin network when it operates in non-ideal conditions. To model such a situation, we decrease the reachability factor α which corresponds to a decrease in forwarding probability. As mentioned in Section 3.2.3, a decrease in forwarding probability implies that

some of the blocks have not reached their destinations or are rejected by the candidate receiver nodes.

Dividing the coverage of different block propagation waves (i.e., the number of nodes that receive the block during a wave) by the network size (i.e., the total number of participating nodes) will yield the probability density function (PDF) of the rate at which the nodes receive the block. Figure 3.9a plots this function for the ideal conditions (i.e. $\alpha = 1.0$). We repeated this experiment for different number of connections per node. For all values of M, the block is fully propagated over the network without any long tail in the PDF. If we slightly decrease α, a long tail appears on the PDF curve as depicted in Figure 3.9b. This means that more waves are needed for 100% block propagation. Nevertheless, more than 95% of the nodes still receive the block as they would in an ideal network. Lowering α will lead to longer tails, which means a slower 100% propagation delay (see Figures 3.9c and 3.9d). Moreover, as plotted in Figure 3.9d, when $\alpha = 0.85$, less than 90% of nodes will receive the block during the time required for 100% propagation under ideal conditions.

3.3.3 Sensitivity analysis of the relay network

When a relay network is present, interested miners can maintain a link to one of the gateways and send any mined block directly to the relay network. The relay network will then relay copies of the block to the rest of the network using high-speed links thus increasing the block propagation speed.

Consider an ideal relay network deployed in the network, where gateways are homogeneous with infinite bandwidth (see Figure 2.5) with very low latency. To model this network, we adapt our basic analytical model aforementioned in Section 3.2.3 as follows. Since a node which maintains a high-speed link to the relay network does not need to exchange inventory messages with the relay gateways, it can immediately send the newly mined block to the gateway. As well, since the relay network do not need to validate the blocks multiple times, we assume the internal block propagation between relay gateways is almost instantaneous: the latency

from the moment the miner sends the block to the relay gateway until the time at which the relay network starts to propagate the block over the network is almost zero. We define the parameter *relay network size* as the percentage of the nodes in the network which are directly connected to the relay network and denote it by γ. For instance, if 100 nodes of the Bitcoin network are connected to the relay network, then $\gamma = 1\%$ (for a total size of 10,000 nodes). Consequently, we can consider the relay network as an initiator node with $\gamma(N-1)$ connections to other neighboring nodes. We will treat with all non-gateway nodes as described in Section 3.3.1.

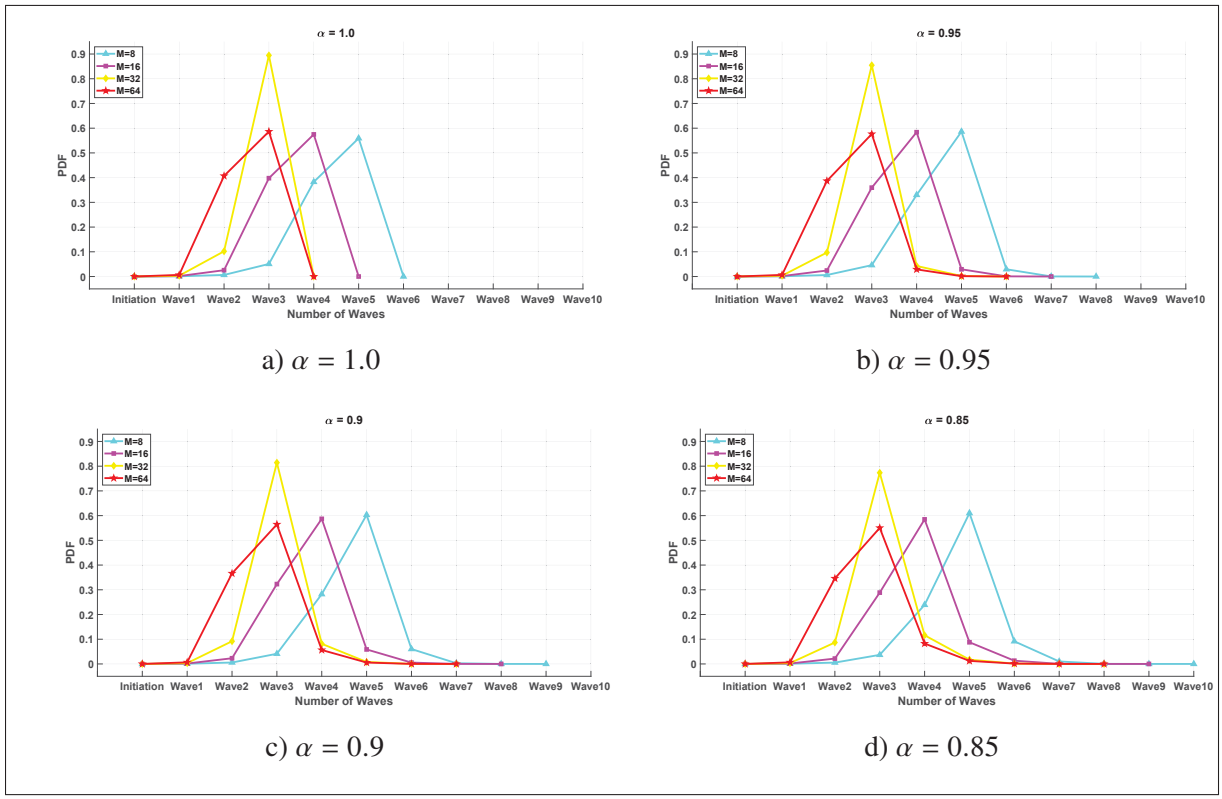
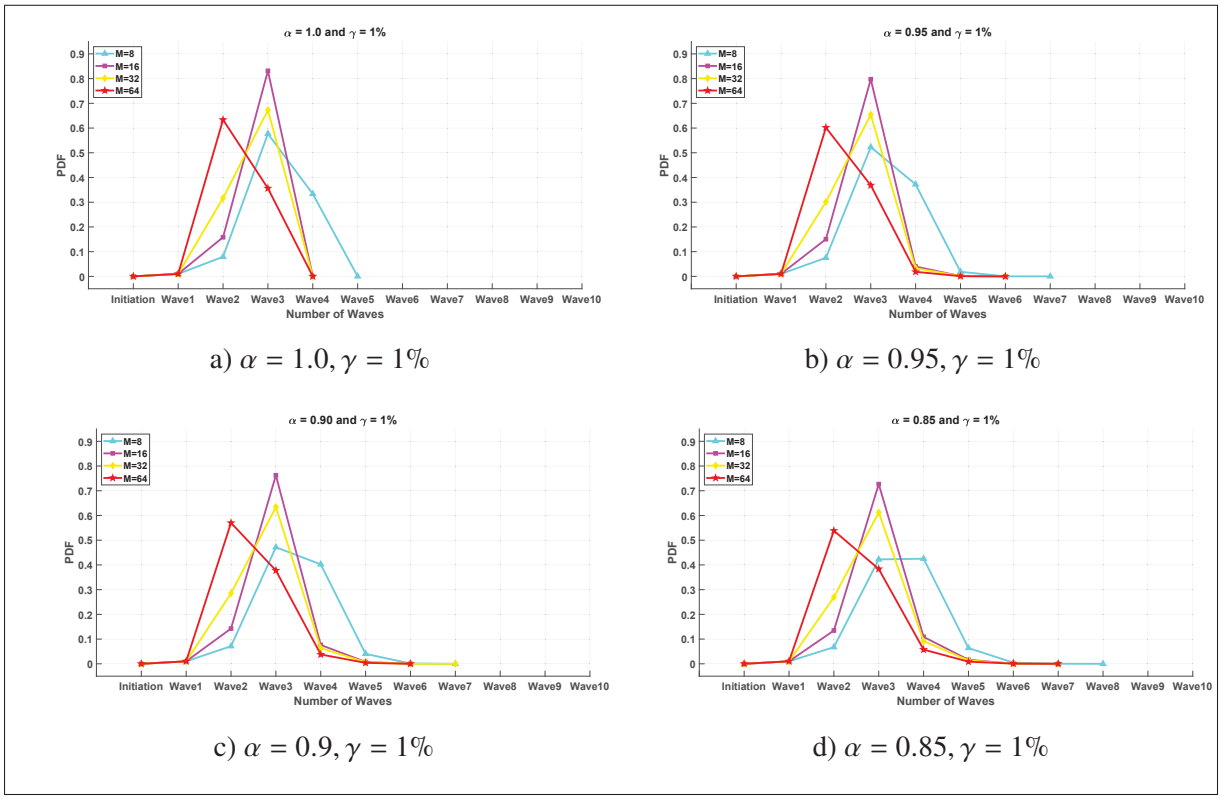


Figure 3.9 Probability density function for waves at which Bitcoin nodes receive the block

We carry out a set of experiments in order to study the impact of the relay network size on the performance of the Bitcoin network with different configurations for the average number of connections per node as well as reachability. In this set of experiments, we employ three values for the relay network size ($\gamma = 1\%,\ 2.5\%\ and\ 5\%$). Simultaneously, we change the value of α

and decrease it gradually to study the effect of message loss on the network performance. We repeat these experiments for different numbers of connections per node. The results are plotted in Figures 3.10, 3.11, and 3.12.



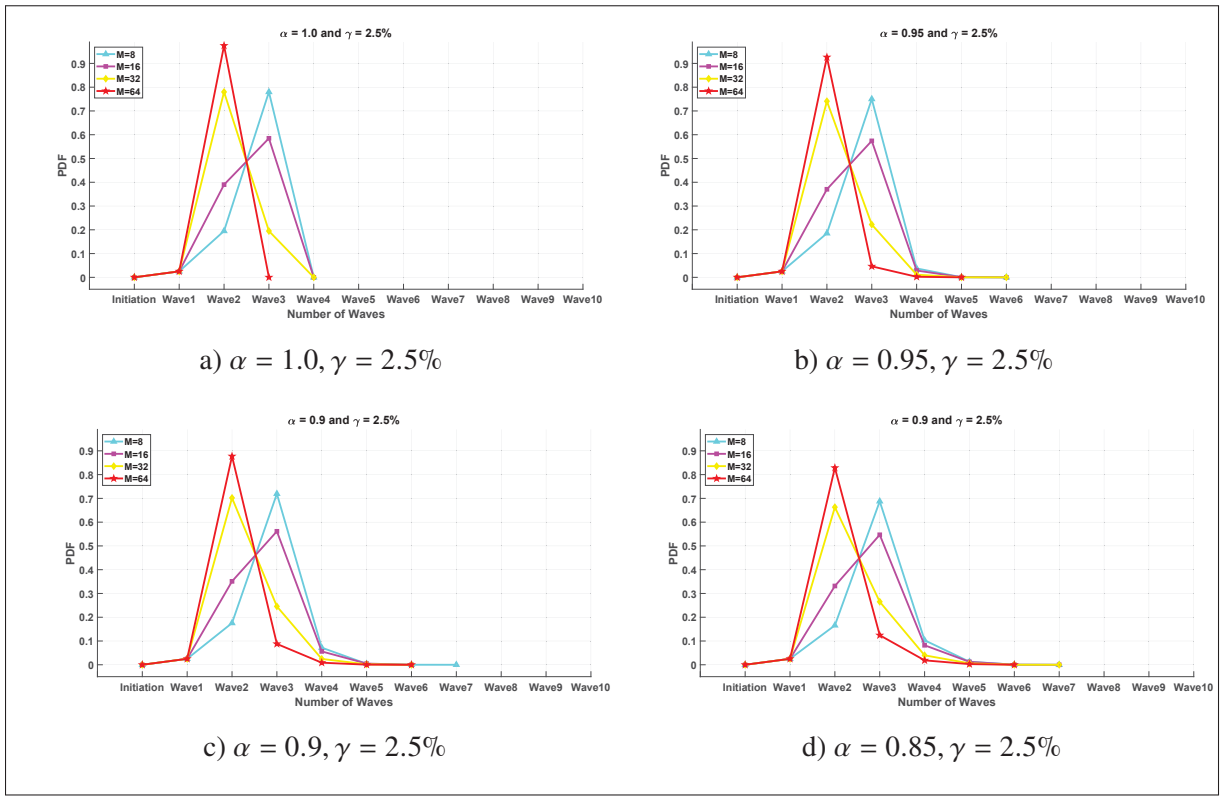
a) $\alpha = 1.0, \gamma = 1\%$ b) $\alpha = 0.95, \gamma = 1\%$

c) $\alpha = 0.9, \gamma = 1\%$ d) $\alpha = 0.85, \gamma = 1\%$

Figure 3.10 Probability density function for waves at which Bitcoin nodes receive the block with a 1% relay network

As depicted in Figure 3.10a, when $\gamma = 1\%$ almost all of the nodes will receive the block in three (when $M = 16,\ 32,\ 64$) or four waves (when $M = 8$). Compared to the configuration when the relay network is not deployed, there is not much difference for 100% block propagation when $M = 64,\ 32$ but for the cases where $M = 16,\ 8$, the relay network accelerates the 100% block propagation for one wave (see Figure 3.9a). According to the results depicted in Figure 3.10b, when reachability is decreased, the long tail of PDF manifests itself even in the presence of the relay network. More reduction in reachability leads to longer tails as shown in Figure 3.10c and Figure 3.10d.

To assess the impact of γ, we conduct another experiment where γ is increased to 2.5%. The results are depicted in Figure 3.11. Figure 3.11a shows the results for the ideal conditions where $\gamma = 2.5\%$. It takes two waves for 100% block propagation when $M = 32$ *and* 64. Also, 100% block propagation needs three propagation waves when $M = 8$ *and* 16. According to these observations, we can claim that the network has a noticeable improvement over the case where $\gamma = 1\%$. Note that although for $M = 16$ the 100% block propagation takes place in three waves, but when $\gamma = 2.5\%$, the block coverage is a bit more than 45% in the second wave when it is around 15% when $\gamma = 1\%$. Also, in Figure 3.11b, Figure 3.11c and Figure 3.11d, we observe that decreasing the reachability causes the long tails to appear on the PDF again. Another effect of decreasing the reachability factor is that the block coverage decreases in all waves.



a) $\alpha = 1.0, \gamma = 2.5\%$ b) $\alpha = 0.95, \gamma = 2.5\%$

c) $\alpha = 0.9, \gamma = 2.5\%$ d) $\alpha = 0.85, \gamma = 2.5\%$

Figure 3.11 Probability density function for waves at which Bitcoin nodes receive the block with a 2.5% relay network

If we keep increasing the relay network size, the network will become faster as expected. For

$\gamma = 5\%$ and $M = 16,\ 32$ and 64, the maximum coverage of the block takes place in the second wave as can be seen in Figure 3.12. The interesting point is that by increasing the relay network size, the behavior of the network with different values of M becomes similar. This implies that with a bigger relay network, the default configuration of the non-relay nodes becomes less important which may endanger the decentralization of the Bitcoin network.

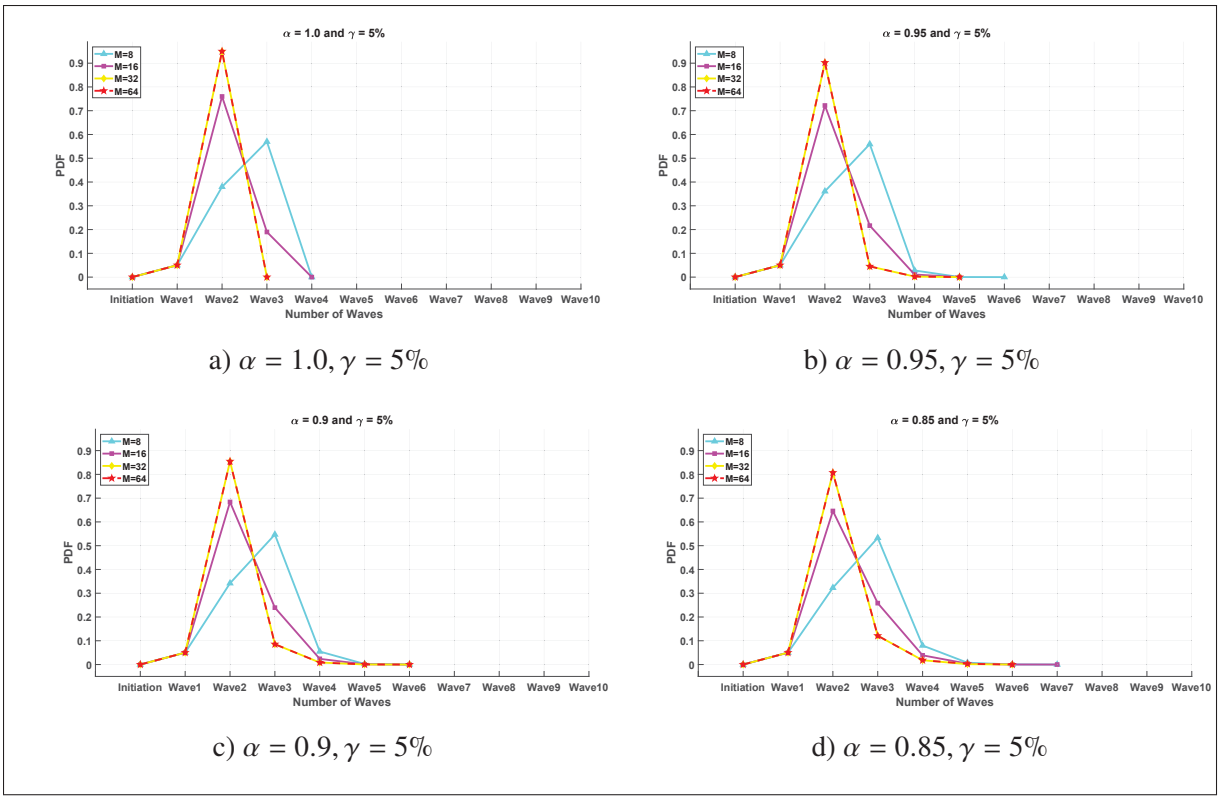
3.4 Managerial Implications

Performance models can provide us with extensive insight into blockchain network dynamics and behaviors. The analytical model proposed in this work contains an approach for predicting and evaluating the performance of Bitcoin-based blockchains (i.e., hard forks of Bitcoin) with controllable input parameters such as block size, number of HBM and LBM connections, network size, and network speed.

From the perspective of a system architect, our performance model can help blockchain developers achieve a good first-cut design that will meet the application requirements, alleviating the need for trial-and-error tuning of parameters, and thus saving on capital expenses. Furthermore, our performance model can be used to conduct *What-if* analyses and anticipate the impact of proposed changes, such as doubling the block size of Bitcoin. The model can therefore inform the manager when taking operational decisions. From the perspective of a cryptocurrency investment manager, our performance model can be used to compare existing Bitcoin-based cryptocurrencies (also called *altcoins*) by quickly calculating important metrics (such as block propagation delay) using collected measurements from the network. These metrics are indicators of the stability and security of the network, which in turn can be used to assess the viability and volatility of a given cryptocurrency.

3.5 Conclusion

In this chapter, we proposed an analytical model for modeling the delay and traffic overhead in a Bitcoin network based on an Erdös-Renyi random graph and derived key features of performance



a) $\alpha = 1.0, \gamma = 5\%$ b) $\alpha = 0.95, \gamma = 5\%$

c) $\alpha = 0.9, \gamma = 5\%$ d) $\alpha = 0.85, \gamma = 5\%$

Figure 3.12 Probability density function for waves at which Bitcoin nodes receive the block with a 5% relay network

measures of the Bitcoin network. We validated our analytical model through simulation and compared to real data measured from the Bitcoin network. We also investigated the effect of the default number of connections on the performance of the Bitcoin network. Although the throughput of Bitcoin can be increased by choosing a bigger size for blocks, this can cause a significant increase in the block propagation time. The delay can be reduced by increasing the average default number of connections per node but this has the drawback of increased traffic overhead in the network.

We used our model to estimate the PDF of the times at which a portion of the Bitcoin nodes receive a propagated block at the end of each wave. We adapted our model to analyze the Bitcoin network in the presence of the relay networks. We observe that bigger relay networks (i.e., relay networks with more miners connected to) can significantly improve block propagation delay,

and make the network less sensitive to the parameters of non-relay nodes, which can endanger the decentralization of the network.

CHAPTER 4

A THEORETICAL MODEL FOR FORK ANALYSIS IN THE BITCOIN NETWORK

Blockchain networks that employ Proof-of-Work in their consensus mechanism may face inconsistencies in the form of forks. These forks are usually resolved through the application of block selection rules (such as the Nakamoto consensus). In this chapter, we investigate the cause and length of forks for the Bitcoin network. We develop theoretical formulas which model the Bitcoin consensus and network protocols, based on an Erdös-Rényi random graph construction of the overlay network of peers. Our theoretical model addresses the effect of key parameters on the fork occurrence probability, such as block propagation delay, network bandwidth, and block size. We also leverage this model to estimate the weight of fork branches. Our model is implemented using the network simulator OMNET++ and validated by historical Bitcoin data. We show that under current conditions, Bitcoin will not benefit from increasing the number of connections per node.

4.1 Introduction

A cryptocurrency network may face some inconsistencies that arise from its decentralized nature. For instance, Bitcoin uses a large and unstructured peer-to-peer (P2P) network, which is susceptible to fork-related inconsistencies when propagating blocks (Narayanan *et al.*, 2016). Forks can occur either due to propagation delay (Antonopoulos, 2017; Decker & Wattenhofer, 2013) or poor connectivity leading to networking partitions (Shahsavari *et al.*, 2019b; Yao, Wang, Leonard & Loguinov, 2009). We differentiate between these types of *natural* forks (which are the focus of this chapter), and *intentional* forks occurring from miners deviating from the protocol (Kwon, Kim, Son, Vasserman & Kim, 2017; Conti, Kumar, Lal & Ruj, 2018b), or changes in the code or protocol, which create a new independent network (which are outside the scope of this chapter) (Tschorsch & Scheuermann, 2016).

Forks are undesirable since they create inconsistencies across the local copies of the ledger, which reduce the reliability of responses to queries about the blockchain data. Thus, the occurrence

of forks implies that blockchain networks are eventually consistent and disturb the notion of immutability in blockchains. As a consequence, users have to wait for an amount of time (usually measured in block confirmations) to be reasonably certain that a transaction is finally committed to the blockchain (Bonneau *et al.*, 2015). For the current Bitcoin network, users typically wait for 6 confirmations (BitcoinWiki), but each user is free to choose its own threshold. Forks caused by malicious manipulations such as Goldfinger attacks can destabilize a cryptocurrency with a significant loss of confidence and destroy its exchange rate (Narayanan *et al.*, 2016). As well, forks can be exploited by malicious entities such as dishonest miners to gain unfair profits by disturbing the normal operation of the system (Conti *et al.*, 2018b).

In light of the above, we can argue that it is imperative for blockchain designers and cryptocurrency analysts to have a comprehensive understanding of the causes and factors related to the formation of forks and have the ability to predict the occurrence of disruptive forks in order to apply countermeasures. To accomplish this, we propose a theoretical model for the analysis of forks in blockchain networks, particularly for Bitcoin. Our model can be useful to predict the long-term impact of proposed changes to Bitcoin, as well as assess the health of *altcoins* networks based on hard forks of the Bitcoin code.

In this chapter, we extend our previous work presented in the previous chapter, which presents a theoretical model for performance modeling and analysis of the Bitcoin network using an Erdös-Rényi random graph model (Erdős & Rényi, 1959). However, the previous chapter was concerned with the propagation of a single block at a time through different gossiping waves, using the Bitcoin inventory protocol, and assumes that no forks can occur. For the current chapter, the main contributions are as follows:

1. We extend our wave-based model to consider the simultaneous propagation of concurrent blocks (forks).
2. We present equations to estimate the fork occurrence probability in the network, and the number of participating nodes in different fork branches (branch weight).
3. We implement our theoretical model using the network simulator OMNET++ and compare to historical Bitcoin data.

4. We provide a sensitivity analysis of various blockchain network parameters for forks, such as the number of connections per node and block propagation delay.

The chapter continues with Section 4.2 in which we present our analytical model to capture the dynamics of forks in the Bitcoin network. Section 4.3 is dedicated to the theoretical and simulation results of our model. Finally, Section 4.4 concludes this chapter. Although the scope of this chapter and the background section is focused exclusively on Bitcoin, the theoretical model presented for analysis of the forks in the Bitcoin network can be adapted to other blockchains such as Ethereum (Buterin *et al.*, 2013).

4.2 Analytical Model

In this section, we first define the notion of *natural* fork caused by propagation delay in a blockchain network. Then, we propose a random graph model for theoretical modeling and analysis of block propagation in the Bitcoin blockchain when a fork occurs. To accomplish this, we use a graph model which was introduced by Erdös and Rényi. This graph has properties suitable for modeling peer-to-peer overlay networks used by blockchain systems.

4.2.1 Fork model

Suppose a block b is the tip of a blockchain $\mathcal{B}$. We call the height h_b of block b as the number of blocks preceding this block in the blockchain starting from the genesis block. In other words, the height of a block is the length of the blockchain to reach it. For the genesis block, h_g is 0. As previously mentioned, the blockchain for Bitcoin is replicated to the full nodes connected together in a very wide P2P network. A fork occurs when there are at least two blocks b and b' that have the same height. Precisely, a fork caused by propagation delay exists when:

$$\exists\, b, b' \in \mathcal{B} \; and \; b \neq b' \mid h_b = h_{b'} \tag{4.1}$$

Assume a node with a blockchain of height of h_b receives block b' with height of $h_{b'}$ where $h_{b'} > h_b$. Because of the longest chain selection rule, it will switch its blockchain tip to block b'. Blocks b and b' can be either in the same branch (then block b is the ancestor of block b') or in different branches. In the prior case, the node will retrieve the intermediate blocks from the network, and in the latter case, block b will become orphaned (removed from the main branch). After this process, the local state of the blockchain maintained by the node should be consistent with that of the rest of the network, assuming this is the longest chain known.

4.2.2 Fork probability analysis

PoW mining for the Bitcoin network can be modeled as a Poisson process and the inter-block time (the time difference between two consecutive mined blocks) follows an exponential distribution. We calculate the probability density function (PDF) of a block to be mined as follows:

$$f(t;\lambda) = \lambda e^{-\lambda t} \tag{4.2}$$

where λ is obtained from the following equation:

$$\lambda = \frac{1}{E[T]} = \frac{1}{t_B} \tag{4.3}$$

where t_B is referred to as the inter-block time and is the average time required to mine a new block. This process is *memoryless* (i.e., the probability distribution is independent of its history) and the probability that another block is mined before the currently proposed block is fully disseminated over the network can be obtained as follows:

$$F(t) = P(T \leq t_{prop}) = \int_0^{t_{prop}} f(t)\, dt = 1 - e^{-\frac{t_{prop}}{t_B}} \tag{4.4}$$

where t_{prop} is the time needed for a block to fully propagate over the network. Equation (4.4) expresses the probability of forks as a function of block propagation time and inter-block time. However, block propagation time itself is derived using several parameters, e.g., block size, bandwidth, the average number of connections per node, the total number of participating nodes, which are already discussed in our previous paper (Shahsavari *et al.*, 2019b).

4.2.3 Block propagation model

To model the P2P overlay network, we use an Erdös-Rényi graph $G(V, L)$ where V and L are the set of vertices and the set of links between participating nodes respectively. For example, if there is a link between node i and node j, then $(i, j) \in L$.

Moreover, we represent our random graph using $G_p(N)$, where N is the total number of participating nodes and p is the independent probability that there exists a link between any two arbitrarily selected nodes in the P2P overlay network. In this chapter, we assume that N is significantly large (e.g., $N \approx 10000$ in the current Bitcoin network (bit, 2018).

Consider a P2P overlay network that consists of N participating nodes: suppose a node mines a block b with height h_b at time t. We refer to this initial node as n_0.

As already mentioned in Chapter 2, node n_0 sends an *inv* message to the set $\mathcal{W}_1 = \{n_1^1, n_2^1, ..., n_M^1\}$ of neighboring nodes in its connection pool. For simplicity, we assume all nodes are connected to M nodes in their connection pool. We assume the sending node sends the *inv* message to its neighbors in succession with a very small negligible delay ϵ between each message.

Since this is the first time that an *inv* message is being sent for this block, none of the $\mathcal{W}_1$ neighboring nodes have the block and will respond the *inv* message with the $getdata$ message with 100% certainty. Upon receiving the $getdata$ response, n_0 will send the complete block to the requesting neighbors. We call this the first step of the block dissemination process wave W_1 (depicted in Figure 4.1a). The time taken from sending the *inv* message until receiving the block by the neighboring nodes is called the *wave length* and denoted by T.

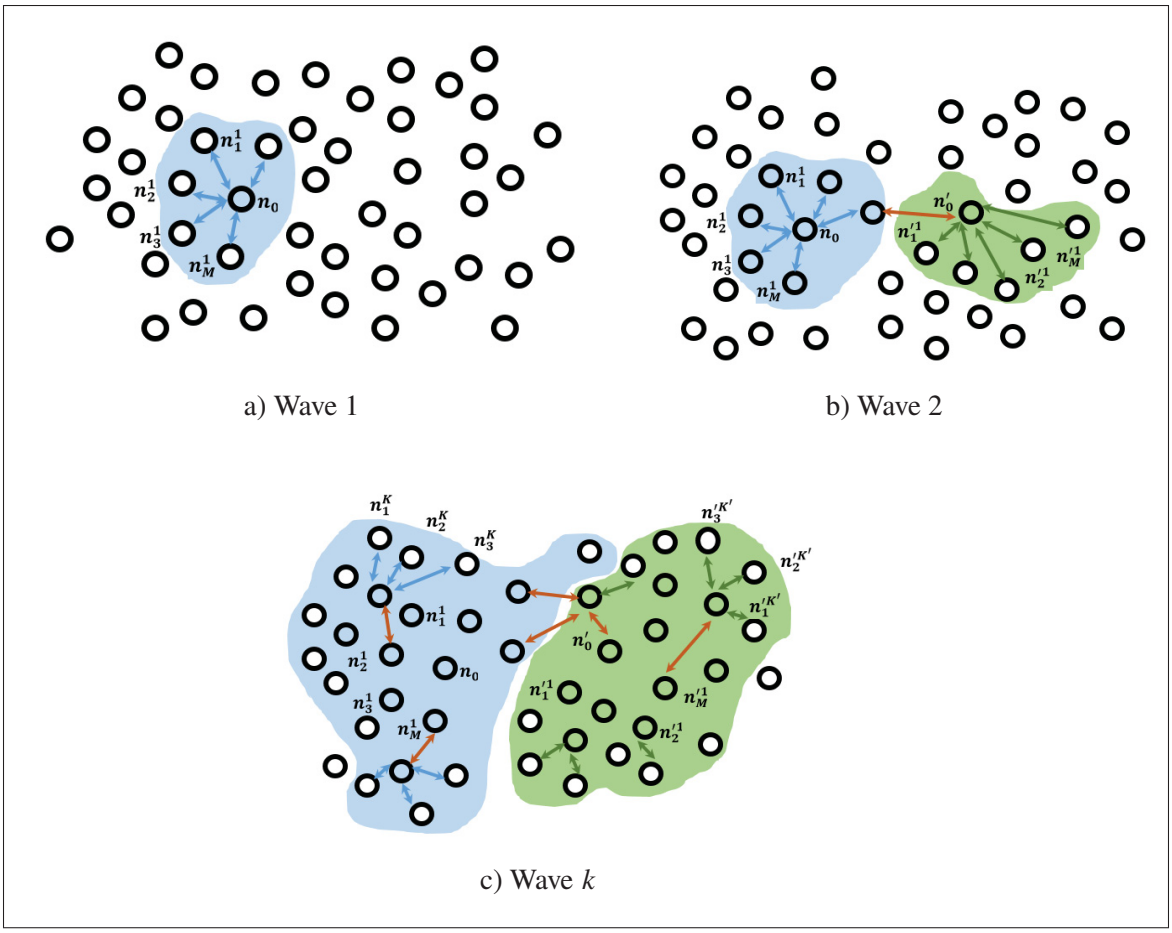


Figure 4.1 Different waves of block dissemination in the proposed blockchain network A fork occurs when node n'_0 starts to propagate its block (b') while the block from n_0 (b) is not fully disseminated. Blue nodes are the nodes that have already received the block b and green nodes are the nodes that have already received the block b'. Accordingly, blue and green arrows indicate successful block transfer. Red arrows show timed-out *inv* messages

For this wave W_1, we define a random variable called the *forwarding probability* p_{f_1}. The forwarding probability is the probability that an *inv* message, containing the information about the newly mined block, will be accepted by the neighboring nodes contacted by a sender node. For wave 1, it is clear that $p_{f_1} = 1$, as the sender node is the block creator n_0.

Suppose another miner node in the network which is not a member of $\mathcal{W}_1$, finds block b' with height $h_{b'}$ at time t', where $t < t' < t + T$ and $h_b = h_{b'}$. We denote this second initiator

node as n'_0. Similarly to n_0, node n'_0 sends an *inv* message to the set $\mathcal{W}'_1 = \{n'^1_1, n'^1_2, ..., n'^1_M\}$ of neighboring nodes in its connection pool. Again, we assume the sending node sends the *inv* message to its neighbors in succession with a very small negligible delay ϵ' between each message. We call this process wave W'_1 (depicted in Figure 4.1b).

However, some of the nodes may have already received a block with a height of h_b in wave W_1 and thus will not accept the block propagated in the wave W'_1. Hence, we calculate the forwarding probability for the wave W'_1 as follows:

$$p_{f'_1} = \frac{N - 1 - |\mathcal{W}_1|}{N - 1} = \frac{N - 1 - Mp_{f_1}}{N - 1} \tag{4.5}$$

Subsequently, the nodes which have received block b will send *inv* messages for that block to the neighbors in their connection pool. We call this process wave W_2. Some of the receiving nodes may not accept the block, if they have already received blocks b or b' and will not send a $getdata$ message. Hence, the forwarding probability for wave W_2 can be calculated as follows:

$$p_{f_2} = \frac{N - 1 - |\mathcal{W}_1| - |\mathcal{W}'_1|}{N - 1} = \frac{N - 1 - Mp_{f_1} - Mp_{f'_1}}{N - 1} \tag{4.6}$$

The set of nodes that will reply positively with a $getdata$ message in wave W_2 can be expressed as: $\mathcal{W}_2 = \{n^2_1, n^2_2, ..., n^2_{|\mathcal{W}_2|}\}$ where $|\mathcal{W}_2| = \lceil p_{f_1} p_{f_2} M^2 \rceil$.

The forwarding probability for wave W_2 is:

$$p_{f'_2} = \frac{N - 1 - |\mathcal{W}_1| - |\mathcal{W}'_1| - |\mathcal{W}_2|}{N - 1} = \frac{N - 1 - Mp_{f_1} - Mp_{f'_1} - p_{f_1} p_{f_2} M^2}{N - 1} \tag{4.7}$$

Each subsequent wave of both blocks will follow a similar pattern as above. In general, we express the forwarding probability of wave W_i as follows:

$$p_{f_i} = \frac{(N-1) - \sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{f_k} - \sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{f'_k}}{N-1} \qquad (1 < i \leq K) \tag{4.8}$$

where K is the total number of waves needed for the block b to be propagated over the blockchain network. Note that some nodes in the network may never accept the block (if it has received block b' first). Concerning block b', the forwarding probability of wave W'_i can be calculated as follows:

$$p_{f'_i} = \frac{(N-1) - \sum_{j=1}^{i} M^j \prod_{k=1}^{j} p_{f_k} - \sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{f'_k}}{N-1} \qquad (1 < i \leq K') \tag{4.9}$$

where K' is the total number of waves needed for block b' to be propagated over the blockchain network.

For values of t other than $t < t' < t + T$, Equations (4.8) and (4.9) can be generalized for $t + mT < t' < t + (m+1)T$ where $m = 0, 1, 2, \ldots$:

$$p_{f_i} = \frac{(N-1) - \sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{f_k} - \sum_{j=1}^{i-1-m} M^j \prod_{k=1}^{j} p_{f'_k}}{N-1} \qquad (1 < i \leq K) \tag{4.10}$$

and

$$p_{f'_i} = \frac{(N-1) - \sum_{j=1}^{i} M^j \prod_{k=1}^{j} p_{f_k} - \sum_{j=1}^{i-m-1} M^j \prod_{k=1}^{j} p_{f'_k}}{N-1} \qquad (1 < i - m \leq K') \tag{4.11}$$

Equations (4.10) and (4.11) are recursive equations and enable us to calculate the number of waves required for all participating nodes to receive *either one* of the blocks b or b':

$$\sum_{i=1}^{K} M^i \prod_{j=1}^{i-1} p_{f_j} + \sum_{i=1}^{K'} M^i \prod_{j=1}^{i-1} p_{f'_j} = N - 1 \tag{4.12}$$

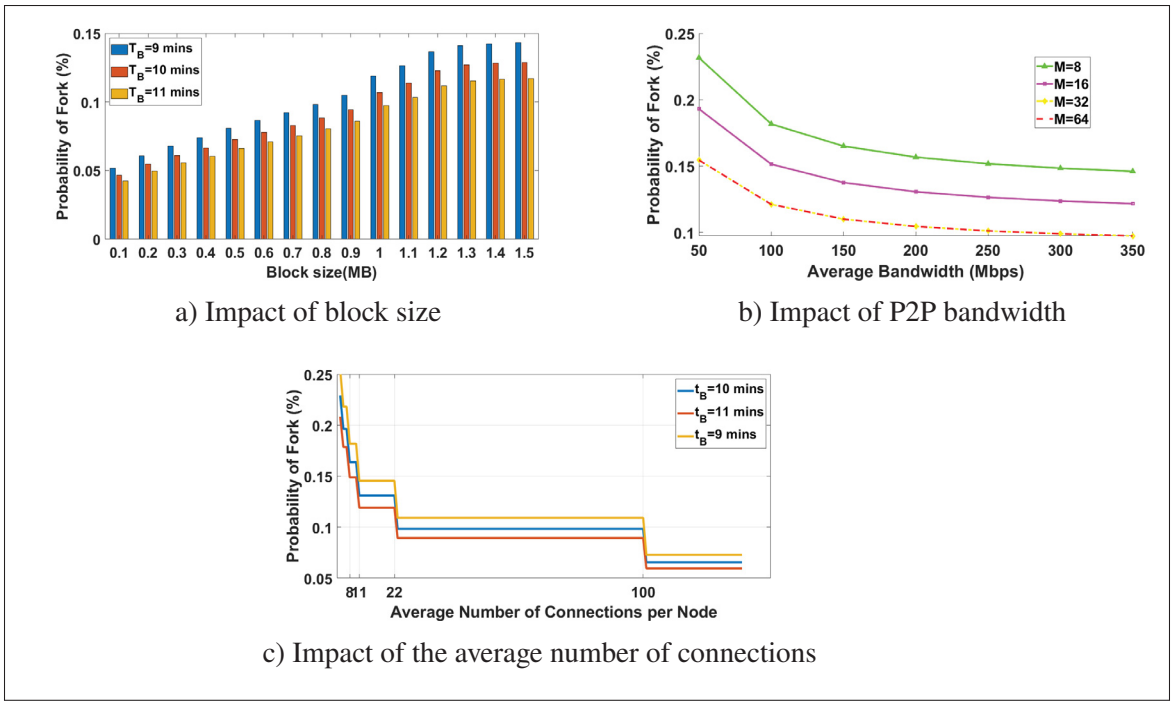
Figure 4.1c illustrates the process of block dissemination during waves W_k and W'_k. To calculate K and K', we follow an iterative approach using Algorithm 5.1. In the first iteration, $p_{f_1} = 1$. Then, we calculate $p_{f'_1}$. We substitute them into Equation (4.12). If the equality is satisfied, the algorithm stops and returns K and K'. Otherwise, we start the second iteration and calculate p_{f_2} using p_{f_1} and $p_{f'_1}$. Then again substitute in the Equation (4.12). If it is now satisfied, we stop and take the K and K'. Otherwise, we continue to find $p_{f'_2}$. This process continues until it finds the appropriate values of K and K'.

The first term in Equation (4.12) indicates the total number of nodes that accept block b as the blockchain head and the second term indicates the nodes which accept the block b'. We call these terms the *weight* of each branch. The branch weight allows determining if the fork is carried over beyond b and b'. If there is a skew towards one branch, the total hash rate working behind that branch is likely to be greater than the other branch, resulting in a greater difference in block mining time, which increases the probability that the next block of the faster branch will fully propagate and orphan the slower branch. Conversely, if both branches have similar weights, the expected block time will be similar and the likelihood of competing blocks being propagated simultaneously increases.

Algorithm 4.1 Calculating number of waves K and K'

Input : The values of N, m, M
Output : K, K', Matrix $\mathcal{P}[p_{f1}...p_{f_K}]$, and $\mathcal{P}'[p_{f'1}...p_{f'_K}]$

$\mathcal{P}[1] \leftarrow 0$
$\mathcal{P}'[1] \leftarrow 0$
$K \leftarrow 1$
$K' \leftarrow 0$
$C \leftarrow 0$; /* Controller */
while C=*0* **do**
  for $i = 1$ *to* K **do**
    if $i - m > 0$ **then**
      $\mathcal{P}[i]=\frac{(N-1)-\sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{f_k}-\sum_{j=1}^{i-1-m} M^j \prod_{k=1}^{j} p_{f'_k}}{N-1}$
      if $\mathcal{P}$ *[i]* ≥ 0 **then**
        $K = K + 1$
        $\mathcal{P}'[i-m] = \frac{(N-1)-\sum_{j=1}^{i} M^j \prod_{k=1}^{j} p_{f_k}-\sum_{j=1}^{i-1-m} M^j \prod_{k=1}^{j} p_{f'_k}}{N-1}$
        if $\mathcal{P}'$ *[i − m]* ≥ 0 **then**
          $K' = K' + 1$
        else
          $C \leftarrow 1$
        end if
      else
        $C \leftarrow 1$
      end if
    else
      $\mathcal{P}[i]=\frac{(N-1)-\sum_{j=1}^{i-1} M^j \prod_{k=1}^{j} p_{f_k}}{N-1}$
      if $\mathcal{P}$ *[i]* ≥ 0 **then**
        $K = K + 1$
      else
        $C \leftarrow 1$
      end if
    end if
  end for
end while
return *K, K′, $\mathcal{P}$, and $\mathcal{P}'$*



a) Impact of block size

b) Impact of P2P bandwidth

c) Impact of the average number of connections

Figure 4.2 Fork probability analysis based on network parameters

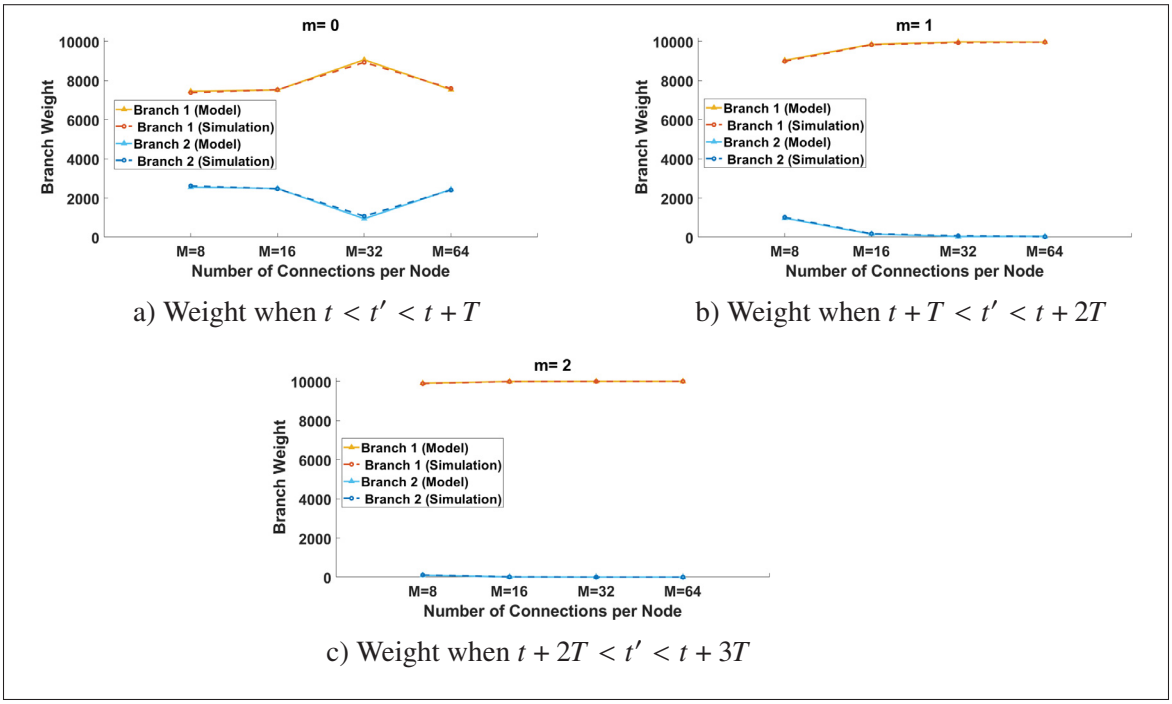
4.3 Results and Analysis

In this section, we first present the results of fork probability analysis in the Bitcoin network based on the following parameters: the block size, the average P2P bandwidth, and the average number of connections per node (see Figure 4.2). Then, we present the results of a fork branch weight analysis based on the difference in the time at which the two blocks are mined (see Figure 4.3).

4.3.1 Settings and implementation

Parameters: To assess the impact of block size on the fork occurrence probability, we use the empirical data provided by (bit, 2018). We extract ~1500 Bitcoin blocks (blocks 504016 through 522429) with reported propagation delay values and block sizes. We consider median values for block size intervals of 100 kB. For all other experiments in this chapter, we consider the average block size as 1MB. For the network settings, we set all parameters the same as the

settings in our previous paper (Shahsavari *et al.*, 2019b).



a) Weight when $t < t' < t + T$

b) Weight when $t + T < t' < t + 2T$

c) Weight when $t + 2T < t' < t + 3T$

Figure 4.3 Branch weight analysis based on t'

Implementation: We implement our model using a discrete event-based simulator called OMNET++ (OmNet). Simulation results are calculated using the event log files generated by the simulation. We also used *Matlab* for theoretical analysis.

Fork probability experiments: To study the impact of different parameters on the probability of fork occurrence in the Bitcoin blockchain, we conduct three experiments. In the first experiment, we estimate the fork probability in the current Bitcoin blockchain as a function of the block size. We carry out this experiment for three different values of inter-block time ($t_B = 9, 10, 11$ *mins*) where $t_B = 10$ *mins* is the average inter-block time in Bitcoin and $t_B = 9, 11$ *mins* are 1-minute deviations from the current average. The results of this experiment are depicted in Figure 4.2a. The fork occurrence probability increases proportionally to the block size. Also, note that decreasing the block time (t_B) increases the fork probability and vice versa. Therefore, the probability of a fork can be manipulated by adjusting the mining difficulty.

To validate the above results, we extract the historical data of 99,120 Bitcoin blocks (blocks 90392 through 189512). During that period, we observe that 90 forks happened, for a reported rate of fork occurrence of 0.09% in the Bitcoin network. Our model results of 0.1% at the block size of 1MB are therefore very close to the empirical result.

In another set of experiments, we study the joint impact of the bandwidth and the average number of connections per node M. As depicted in Figure 4.2b, the probability of fork decreases when the bandwidth or the number of connections increases. For $M = 32$ and $M = 64$, since the block propagation delay is the same for both (shown in (Shahsavari *et al.*, 2019b)), the curves are almost identical. This corroborates the fact that forks are caused by network delay.

Figure 4.2c shows a detailed sensitivity analysis for the average number of connections per node M. There is no significant difference in the probability of fork occurrence when $22 \leq M \leq 99$, which is a significant margin. Since M is around 32 for the real network, we can therefore claim that Bitcoin is not sensitive to the value of M, currently centered around 32. This experiment also confirms that a lower value of t_B increases the chance of having a fork.

Branch weight experiments: We conduct another set of experiments based on the time instant at which the competitor node begins to propagate its block while another node has already started propagating its proposed block. For this set of experiments, we study the sensitivity of the network with respect to the average number of connections M.

In the first experiment, we set $m = 0$ (see Equation (4.11)). This means that the second miner finds and starts to propagate its block (we call it $block\ 2$) during the first wave of dissemination for $block\ 1$. Figure 4.3a shows the results of this experiment. The simulation results closely match our theoretical results. Generally, increasing M yields a heavier weight for the first branch. In particular, note the special case of $M = 32$ where the difference between the two branches is maximal. This is justified by the fact that the Bitcoin network experiences minimal traffic overhead when $M = 32$ (as shown in our previous paper (Shahsavari *et al.*, 2019b)): the first block is disseminated much more efficiently during each wave.

In the next experiments, we set $m = 1$ and $m = 2$, which means the second miner starts propagation its block during the second and third waves of dissemination for $block\ 1$, respectively. The results of these experiments are shown in Figures 4.3b and 4.3c. Again, the simulation results and the theoretical results are almost the same. Increasing the M or m will increase the weight of the first branch. When both M and m have high values, the weight of the second branch is near zero and we can thus claim that the impact of this fork is almost negligible on the consistency of the Bitcoin network: $block\ 2$ will quickly be orphaned.

4.4 Conclusion

In this chapter, we present an analytical model to estimate the probability of fork occurrence in the Bitcoin network, using a random graph to model the Bitcoin overlay network and dissemination waves to model the inventory-based block propagation protocol. We investigate the effect of several blockchains and network parameters on the probability of forks. Our results show that reducing the block time compromises the security of the blockchain by increasing the probability of a fork. The average number of connections per node currently has no impact on the probability of forks, since Bitcoin currently operates within a stable range of 22-99 connections. In addition, we investigate the impact of the time difference between two concurrent blocks and the average number of connections per node on the weight of fork branches. If the later miner starts to propagate its block too late and the number of connections per node is sufficiently high, the impact of the fork on the network is almost negligible.

CHAPTER 5

TOWARD QUANTIFYING DECENTRALIZATION OF BLOCKCHAIN NETWORKS WITH RELAY NODES

In this chapter, we present a methodology for quantifying the decentralization degree of a blockchain network. To accomplish this, we use two well-known graph models of Erdös-Rény and Barabási–Albert in order to study the blockchain network topology. We then quantify the decentralization degree using the clustering coefficient of our network models. We validate our approach through extensive simulations and analyze the decentralization degree with respect to network parameters such as the number of connections per node and the peer selection algorithm. Our results expose the trade-off between the average shortest path and the decentralization degree. Furthermore, we observe the impact of the average shortest path on the network speed and traffic overhead. Finally, we demonstrate that the presence of hub-like nodes such as relay gateways negatively impacts the decentralization degree of blockchain networks.

5.1 Introduction

The first public blockchain was Bitcoin and was introduced by Nakamoto in 2008 (Nakamoto, 2009) since then, DLT has continued to evolve through many more advanced public and private blockchains such as Ethereum (Wood *et al.*, 2014) and Hyperledger Fabric (Androulaki *et al.*, 2018). Despite vast differences in design, operation, and application, the fundamental properties of DLT remain network decentralization and data immutability.

In computer networks, decentralization comprises shifting from the traditional client-server architecture to a peer-to-peer (P2P) network in which all nodes have the same role. In blockchain networks, decentralization is usually expressed at the application layer as the execution and storage of transactions without a trusted third party, or at the consensus layer through a byzantine fault-tolerant protocol. However, an overlooked aspect is the decentralization of the public blockchain network itself, which is sensitive to the peer selection strategy and network protocol. In other words, even in a permission-less blockchain network where nodes can freely join and

connect, the network can be more or less decentralized depending on how each node selects its neighbors to maintain connections to within the P2P network since these connections affect how transactions and blocks propagate throughout the system.

In practice, blockchain-based systems have encountered scalability and performance issues (Eyal & Sirer, 2014; Atzei, Bartoletti & Cimoli, 2017; Brandenburger, Cachin, Kapitza & Sorniotti, 2018). Thus, there have been proposals for performance improvements, which range from attempts at speeding up the blockchain overlay networks (e.g. (Corallo, 2016b, 2019; Ozisik *et al.*, 2017; Basu *et al.*, 2019; Klarman *et al.*, 2018; Fadhil, Owenson & Adda, 2017)) to proposals for increasing the throughput of the system (e.g. (Croman *et al.*, 2016; Gueta *et al.*, 2019; Yang *et al.*, 2019b; Yu, Nikolic, Hou & Saxena, 2018)). While it is clear that these proposals are beneficial to the blockchain systems in terms of performance, it is not yet known what impact (positive or negative) they have on the other fundamental property of blockchain, which is network decentralization. In this chapter, we seek to address this gap by formally studying decentralization as a property of the P2P network graph.

Network decentralization is also overlooked in other common aspects of blockchain systems. For instance, Bitcoin uses a bootstrapping stage where participating nodes connect to seed nodes. This bootstrapping phase prevents blockchains from having a completely random topology, particularly, if the nodes remain connected to the seed nodes for a long time. This is important since a full decentralization network should have a completely random topology where nodes have no preference when selecting a peer to maintain a connection. Relay nodes, which are the focus of this chapter, can have the same impact on the network topology. As well, uneven geographical distribution is another cause of centrality if peers use proximity-aware connections.

While the main aim of blockchain-based systems is to remove the need for a trusted third party (TTP), a poorly decentralized blockchain network can be prone to be a single point of failure (network partitions) or vulnerable to denial of service (DoS) attacks. Furthermore, poor network decentralization can lead to governance issues, as a minority of central nodes can enforce a certain protocol version by limiting communication among nodes that support a different

version. Therefore, a methodology for quantifying network decentralization is crucial in order to analyze these proposals deeply and ensure they do not have unintended side effects on network decentralization. Furthermore, our proposed criterion for analytical and numerical analysis of network decentralization will help blockchain designers compare multiple systems along that dimension.

In this chapter, we focus our attention on studying the impact of relay networks on decentralization. Relay networks are sub-networks that consist of powerful nodes which maintain many connections simultaneously in order to reduce block and transaction propagation times (Corallo, 2019; Basu *et al.*, 2019). Relay networks affect the peer selection strategy of the entire P2P network, which will preferentially connect to the relay nodes. To do so, we propose an analytical approach based on the random graph models of Barabási–Albert (BA) (Barabási & Albert, 1999) and Erdös-Rény (ER) (Erdős & Rényi, 1959), which are suitable for modeling permissionless blockchains with and without a relay network, respectively.

The contributions of this chapter are as follows:

1. We present an analytical approach for quantifying the decentralization degree in blockchain networks based on the peer selection strategy (random vs. prioritizing relays) for blockchain networks with different architectures.
2. We verify our approach by implementing a complex network generator and running extensive simulations. Furthermore, we validate our model using an experimental dataset mined from the Bitcoin network.
3. We present simulation results and analysis of decentralization based on several important metrics such as the average shortest path and the average number of connections.
4. We provide a detailed comparison between blockchain networks with varying architectures and topologies with respect to decentralization and network speed.
5. We study the impact of the relay networks on the decentralization degree of the blockchain network.

The rest of this chapter is organized as follows. In Section 5.2, we describe the system and graph models applied in our paper. In Section 5.3, we verify our complex network generator and validate our simulation with the real Bitcoin network. Simulation results and related discussions are presented in Section 5.4. Finally, Section 5.5 concludes this chapter.

5.2 System model and analysis

In this section, we first perform a decentralization analysis of permissionless blockchains using the clustering coefficient. Then we propose the average shortest path as an indicator of the network speed. These two metrics represent the trade-off between decentralization and performance, respectively.

The clustering coefficient is a metric that captures the tendency of nodes in a graph to form a cluster. For a simple graph, the clustering coefficient is bounded between zero and one and is defined as *local clustering coefficient (LCC)* and *global clustering coefficient (GCC)*.

Local clustering coefficient: this measure is also known as Watts—Strogatz (Watts & Strogatz, 1998) clustering coefficient and for any arbitrary node i in the network, the local clustering coefficient can be calculated as follows:

$$c_i = \frac{2L_i}{k_i(k_i - 1)} \tag{5.1}$$

where L_i denotes the number of edges between the k_i neighbors of the node i. Consequently, the average network clustering coefficient can be calculated as follows:

$$C = \frac{1}{N} \sum_{i=1}^{N} c_i \tag{5.2}$$

where N is the number of participating nodes.

Global clustering coefficient: this measure is defined as follows:

$$C = \frac{number\ of\ closed\ triplets}{total\ number\ of\ triplets} \tag{5.3}$$

where the triplet is an ordered set of three nodes that are connected together by either two (in open triplets) or three (in closed triplets) edges.

Note that the average clustering coefficient defined in Equation 5.2 and the global clustering coefficient defined in Equation 5.3 are not equivalent. The local clustering coefficient reflects the fraction of pairs of neighbors of a given individual node that are connected together, and hence the average value of the tendency of the individual nodes to form a cluster, while the global clustering coefficient reflects the overall structure of the nodes in the network. Although both may exhibit the same behavior in most cases, nevertheless those can diverge in some extreme networks (Bollobás & Riordan, 2003; Estrada, 2016) which are out of the scope of this chapter. In this chapter, we study both of the mentioned measures in blockchain networks.
According to the above, a higher clustering coefficient indicates a higher degree of decentralization due to the higher number of closed loops in the graph. Closed loops are trios of nodes that are fully connected together (i.e. to form a triangle). This kind of formation is beneficial for decentralization since the fully connected nodes can directly communicate without an intermediate node. In contrast, low values of the clustering coefficient signify that there are fewer alternative paths in the system, hence, there exist some centralized nodes through which the traffic must necessarily flow.

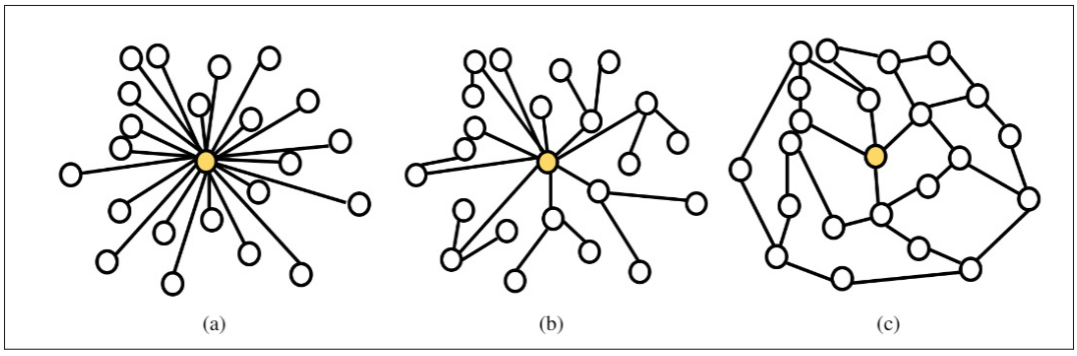


Figure 5.1 Importance of the average shortest path

5.2.1 Performance analysis

The shortest path d_{ij} is the path between node i and node j with the least number of steps. The average shortest path length can be obtained from the following equation:

$$D = \frac{1}{n(n-1)} \sum_{i \neq j} d_{ij} \tag{5.4}$$

In blockchain networks with a gossip protocol, the average shortest path plays a very important role in the speed of information propagation. Figure 5.1 demonstrates this fact by comparing three different networks together. in all of the networks above, the colored node is the initiator node that intends to disseminate its information using a gossip protocol. The initiator node in the network (a) is able to disseminate the information in only one gossip round. In network (b), the initiator node will need two gossip rounds to disseminate the information. In network (c) more than two gossip rounds are needed.

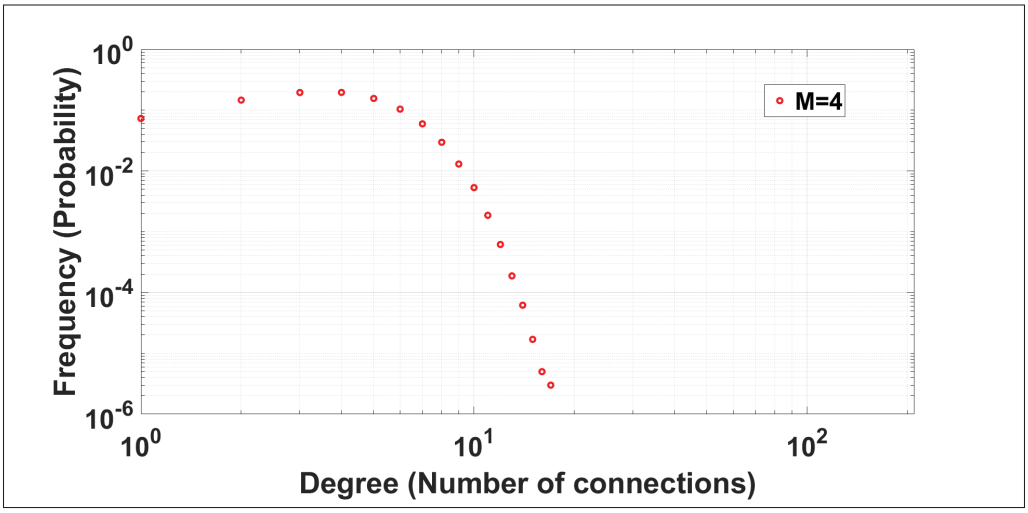


Figure 5.2 Degree distribution of nodes with M and M_0 set to 4 for ER network with a network size of $N = 1000$

5.3 Verification and validation

In this section, we first verify our complex network generator which can be used for generating P2P networks with a given size and known average number of connections per node as well as specific conditions imposed by the blockchain protocol. Then, we validate our model by comparing it with the real Bitcoin network.

5.3.1 Simulation Model and Verification

We now provide details on our methodology to generate networks that will satisfy the properties required to be considered a BA or ER graph. Our generator is very important for our analysis which requires many networks to be generated and studied in order to understand the impact of several network characteristics on decentralization. We show that the graph generated by our generator has the same features as expected in theory We verify that our implementation is correct by running over 1000 trials for each simulation with different initial random seeds. Furthermore, we carefully controlled the generated networks in order to ensure each generated network is unique.

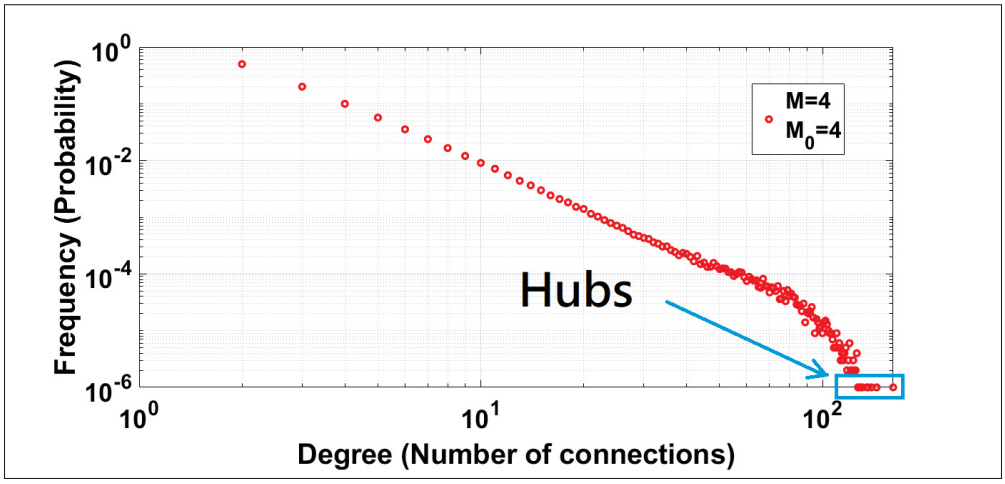


Figure 5.3 Degree distribution of nodes with M and M_0 set to 4 for BA network with a network size of $N = 1000$

ER Network: The methodology for generating the ER network is presented in Algorithm 5.1. This algorithm starts with an initiator node. The rest of the nodes join the network respectively and get connected to existing nodes with a probability of p. We used this algorithm to generate an ER network of 1000 nodes with a connection probability of $p = 0.004$. Regarding these values and the Equation 2.3, we expect the average degree of the nodes to be around 4 with a Poisson distribution. In fact, the majority of the nodes should have around 4 connections to other nodes. The degree distribution of nodes in 1000 repeats of this algorithm is depicted in Figure 5.2, which validates our expectations. As mentioned, this model is suitable for blockchains with no relay networks deployed and nodes periodically refresh their connection pool and forget the initial seeds.

Algorithm 5.1 Erdös-Rény model generation algorithm

Input : The values of N and p
Output : E-R adjacency matrix $\mathcal{A}$

$\mathcal{A}$=$0_{N\times N}$
$E \leftarrow 0$
$Ctrl \leftarrow 0$
for $i = 2$ *to* N **do**
    for $j = 1$ *to* $i - 1$ **do**
        r $\leftarrow Rand(1)$
        ; /* r is a random number uniformly selected from the interval $(0, 1)$ */
        if $p > r$ **then**
            $\mathcal{A}[i,j] \leftarrow 1$
            $\mathcal{A}[j,i] \leftarrow 1$
        end if
    end for
end for
return $\mathcal{A}$

BA Network: Algorithm 5.2 briefly describes the methodology we used to generate a power-law scale-free network. This algorithm starts with M_0 initiator nodes connected together via $M_0 - 1$ links as a complete graph. Then, the rest of the nodes join the network respectively and one by one and get connected to each of the existing nodes with a probability of p_i as defined in

equation (2.4). During the attachments, the algorithm controls the average degree of the nodes in order to keep it around M. In order to ensure that the generated network has the mentioned features, we conducted a simulation with 1000 nodes. As shown in Figure 5.3, our generated network is a BA network according to (Barabási & Albert, 1999). This simulation is conducted for $M = M_0 = 4$. A few nodes appear with a high degree of connections and are labeled as hubs. In our experiments, we expect to observe a few hubs around the initial seed size. Furthermore, the majority of the nodes have a degree around the average M.

5.3.2 Model validation

We validate our generator against a simulation of the Bitcoin network. We set the maximum number of outgoing connections to 8 and the maximum number of incoming connections to 117, in accordance to the Bitcoin protocol. The average degree is set to 32 connections per node(Decker & Wattenhofer, 2013). We conduct the simulation for 10,000 nodes, which is the real size of the Bitcoin network (Bit, 2021). A schematic of the generated network can be seen in Figure 5.4. The output result contains GCC, LCC, and the average shortest path length (ASPL) in each network as presented in Table 5.1. In order to compare our results with the experimental data mined from the Bitcoin network, we use the concept of dissemination waves (Shahsavari *et al.*, 2019b). Block propagation is modeled using a set of subsequent waves, each of which covers one hop in the Bitcoin P2P network. In this model, LBM is modeled as a set of long waves, and HBM is modeled as a set of short waves. Each block transfer in LBM is equivalent to roughly three block transfers in HBM. This concept is depicted in Figure 5.5. Our simulation results show the ASPL amounts of 3.05 and 3.22 for ER and BA models respectively. According to results reported in (Shahsavari *et al.*, 2019b), 100% of block propagation takes 3.33 waves (i.e. three long waves plus one short wave) which is almost the same as BA algorithm results. But the ER algorithm underestimates it to $\frac{1}{3}$ of a wave. Thus, the BA algorithm is a better choice for simulating the Bitcoin network.

Algorithm 5.2 Barabási–Albert model generation algorithm

Input : The values of N, M and M_0
Output B-A adjacency matrix $\mathcal{A}$
:

```
1  A=0_{N×N}
2  E ← 0; Ctrl ← 0
3  for i = 1 to M_0 do
4      for j = 1 to M_0 do
5          A[i,j] ←1
6          A[j,i] ←1
7          E ← (E + 2)
8      end for
9  end for
10 for i = M_0 + 1 to N do
11     Degree ← 0
12     for j = 1 to i − 1 do
13         while Degree < M/2 do
14             Randomly target one of nodes (except node i)
15             a ← Degree of node i/E
16             r ← Rand(1)
17             // r is a random number uniformly selected from the interval (0, 1)
18             if a > r then
19                 A[i,j] ←1
20                 A[j,i] ←1
21                 E ← (E + 2)
22             else
23                 Ctrl ← 1
24                 while Ctrl = 1 do
25                     Randomly target one of
26                     nodes (except node i)
27                     a ← Degree of node i/E
28                     r ← Rand(1)
29                     // r is a random number uniformly selected from the interval (0, 1)
30                     if a > r then
31                         A[i,j] ←1;A[j,i] ←1; E ← (E + 2);Ctrl ← 0;
32                     end if
33                 end while
34             end if
35         end while
36     end for
37 end for
38 return A
```

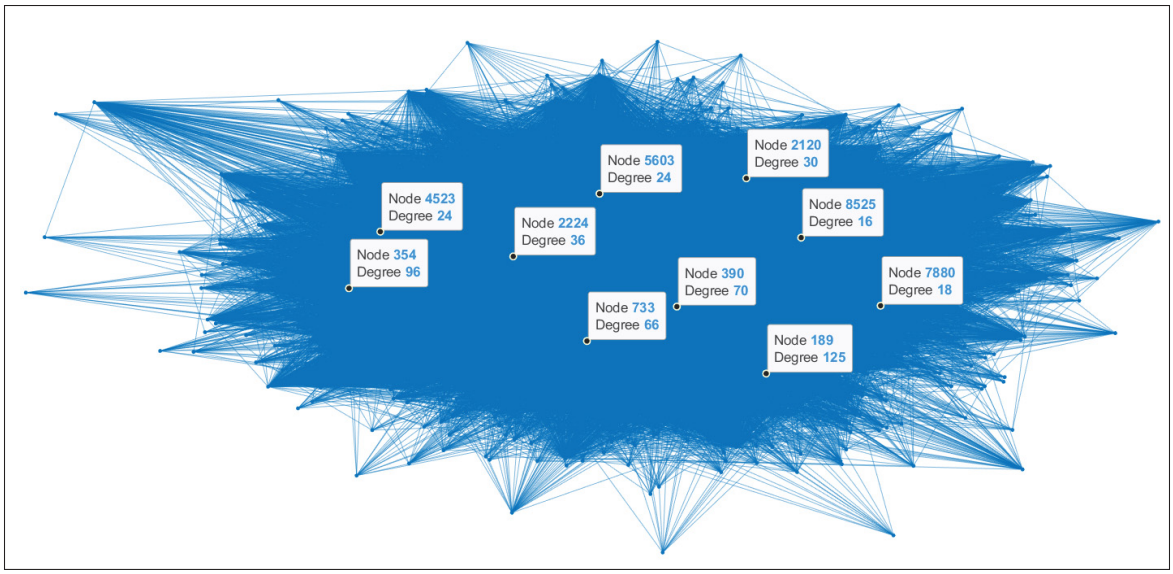


Figure 5.4 Schematic of a generated network for Bitcoin

Generation Algorithm	ASPL	GCC	LCC	Achieved M
ER	3.05	0.033	0.032	32
BA	3.32	0.0228	0.0637	31.99

Table 5.1 Simulation results for Bitcoin. Targeted M was 32.

5.4 Results and analysis

In this section, we present simulation results and analysis of network decentralization. We study the impact of several network parameters, such as the average number of connections per node, peer selection strategy, and relay network size over the decentralization degree of the network, which is expressed in clustering coefficients.

Methodology: We conducted extensive simulations in order to study the impact of the network architecture and peer selection strategy including the average number of connections per node on the overall decentralization and speed of the blockchain networks. Simulations are carried out for a network with a size of 10,000 nodes.

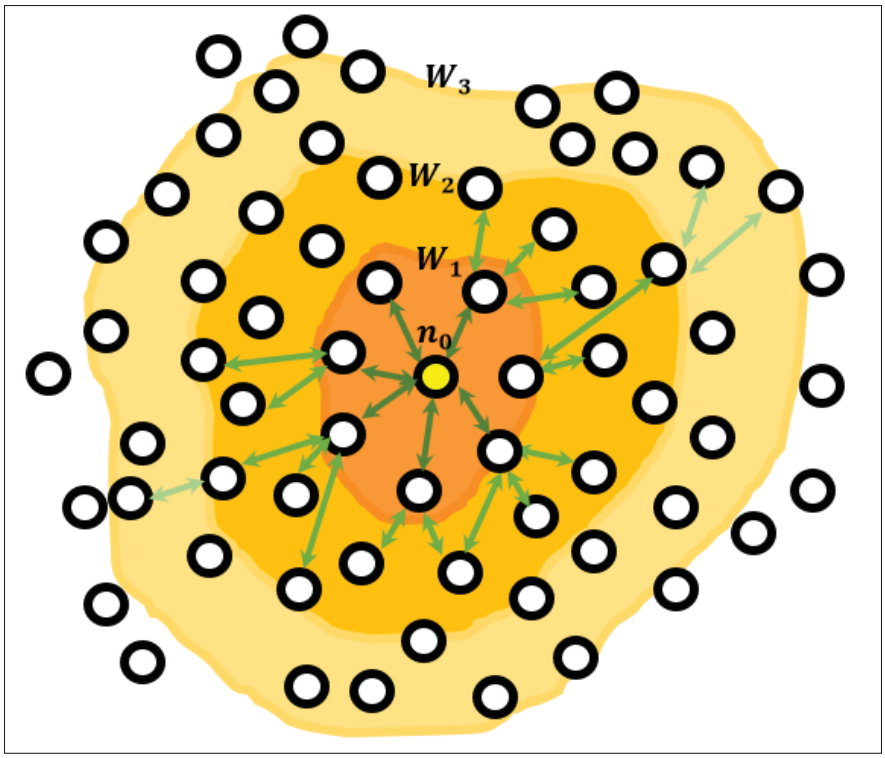


Figure 5.5 Concept of dissemination waves in the Bitcoin network
n_0 is the miner of the block and W_1, W_2, and W_3 are the first, second, and third dissemination waves

5.4.1 Peer selection strategy

We first study the effect of the peer selection strategy and the average degree of the nodes on the decentralization degree of the blockchain networks. The ER network employs a uniformly random peer selection strategy, while the BA network employs preferential attachment to the relay nodes. The clustering coefficient measures the degree of decentralization of the system and has a value between 0 and 1. With a value of 0, the network is completely centralized with a tree-like structure. With a value of 1, the network is a complete graph which is fully decentralized, since each node can communicate with any other node without any intermediary.

As seen in Figure 5.6, both GCC and LCC are decreased in a BA network with the increase of the average number of connections per node and then tend to a constant amount. For $M = 32$, GCC reaches the minimum amount. The network tends to form fewer closed loops and

instead, more star-like nodes with a higher degree of connections appear. This situation in the blockchain networks can be referred to as the appearance of hub-like nodes such as relay gateways.

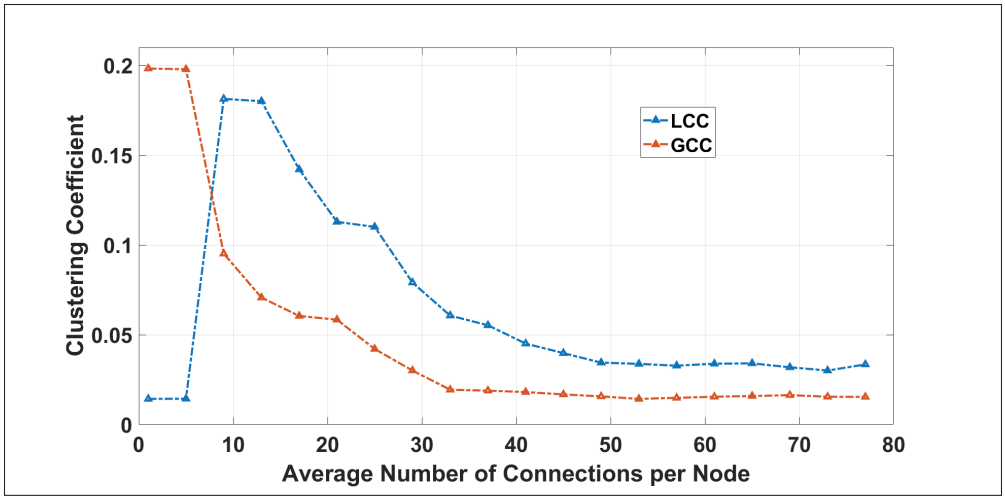


Figure 5.6 Local and global clustering coefficient of BA network for $N = 10000$

In another test, we repeated the experiment above for an ER network. As can be seen in Figure 5.7, the decentralization degree increases linearly with the increase in the metric p (hence the average number of connections per node). However, at the low values of M shown here, the ER networks have a worse absolute decentralization degree than the BA network counterpart at the same value of M. This is because at these low values of M, the ER network is sparsely connected, and thus the paths between nodes contain a lot of redundancy.

At higher values of M, ER networks eventually outperform the BA networks. The reason is that in an ER graph, it tends to a complete graph with an increase of p. This means, in the absence of hubs and relay networks, when every node selects its peers randomly with the same probability, the degree of decentralization will increase.

In light of the above, we claim that relay networks hamper the decentralization degree in blockchain networks with a sufficiently high average number of connections per node.

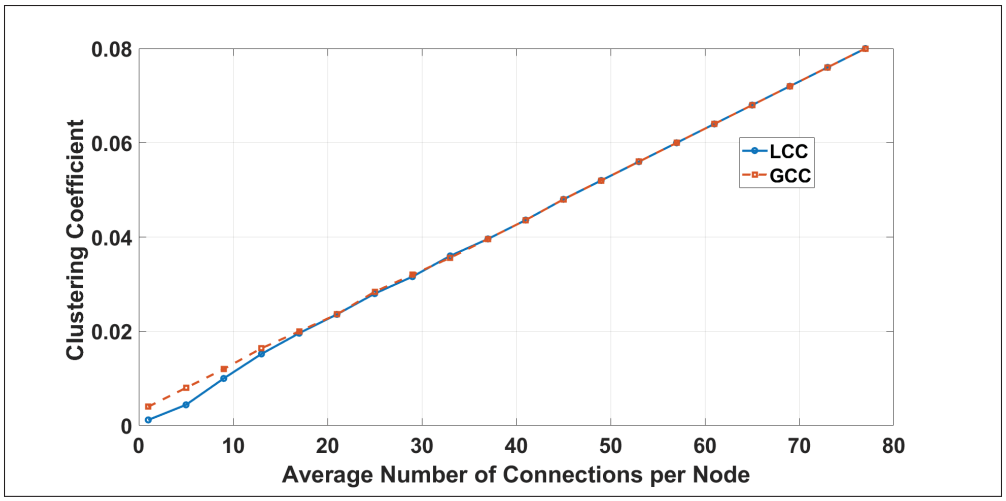


Figure 5.7 Local and global clustering coefficient of ER network for $N = 10000$

5.4.2 Shortest Path Analysis

In the next experiment, we study the trade-off between decentralization and performance by analyzing the shortest path length of the ER and BA networks. As depicted in Figure 5.8 for both the BA and ER networks, increasing the average degree of the network decreases the average shortest path length. Note that there are no results for $M < 8$ for an ER network because such networks are not fully connected. Partitioned networks returned an amount of infinite for the shortest path length and it led to the amount of infinite for the average after extensive simulations. However, these networks contain a partition and cannot achieve consensus. Due to the gossiping protocol, a smaller average for the shortest path means faster information propagation. In particular, the result from the BA network experiment indicates that the growing presence of a relay network will improve performance, at the expense of decentralization (as seen in the previous graph Figure 5.6). In the ER network, an increase in the number of connections has a limited impact on the average shortest path, as the uniform random peer selection strategy does not efficiently leverage those additional connections to reduce the path length.

Note that we assume that all P2P connections have the same bandwidth and geographical distance. However, we can generalize our results by taking into account both additional parameters.

Nevertheless, increasing the network speed by increasing the average number of connections per node is not a good idea since it comes at the cost of increased overheads such as traffic overhead in the network (Shahsavari *et al.*, 2019b).

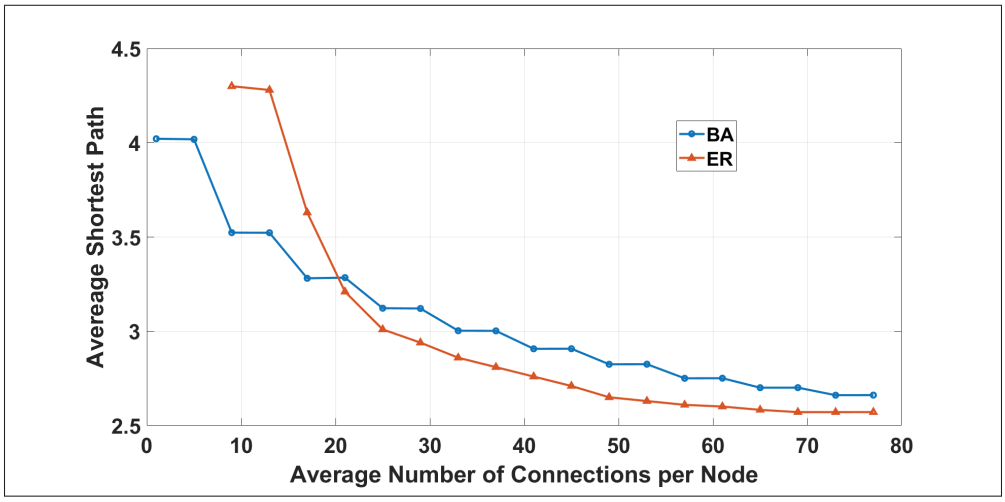


Figure 5.8 Average shortest path for ER and BA networks for $N = 10000$

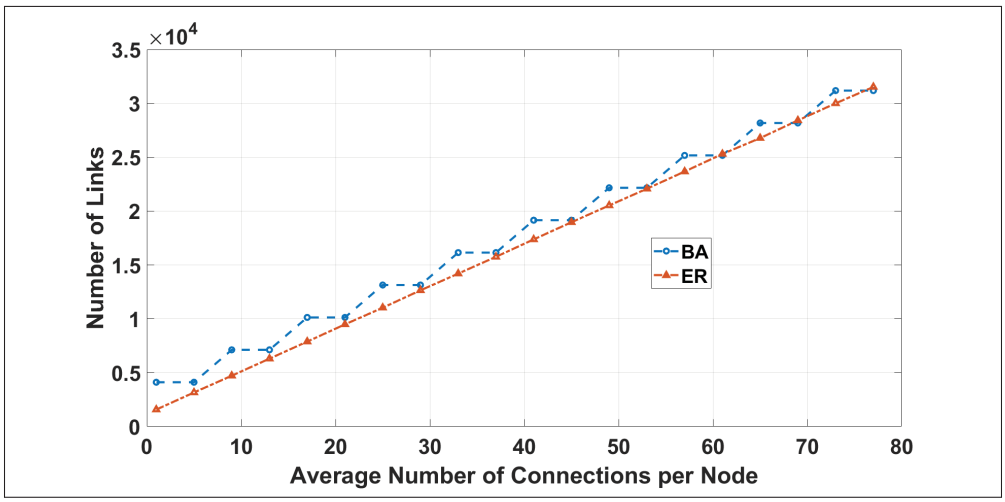


Figure 5.9 Total number of links (traffic overhead) for N=10000

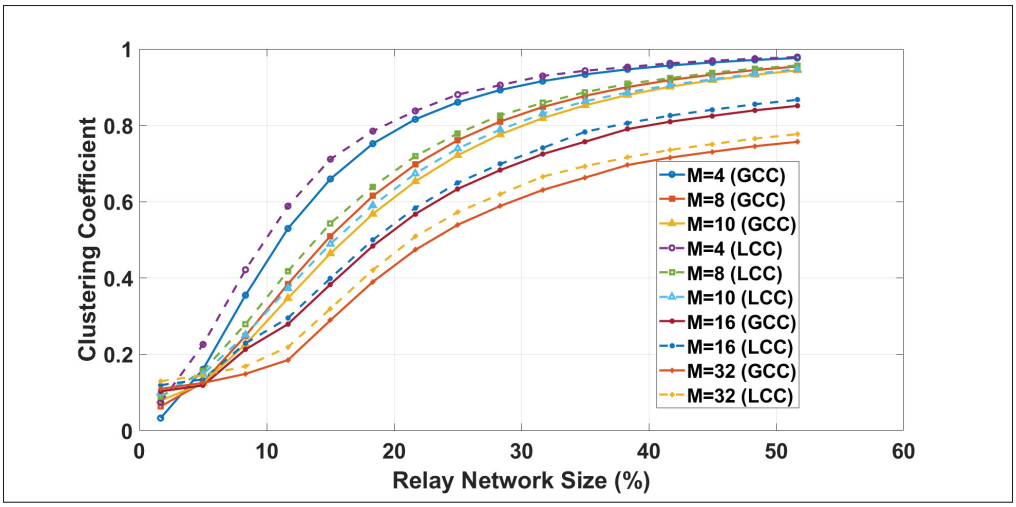


Figure 5.10 Local and global clustering coefficient for a BA network with $N = 10000$

5.4.3 Traffic Overhead Analysis

As mentioned in Section 2.1.9, the total number of links represents the lower bound of the traffic handling overhead. As shown in Figure 5.9, in both BA and ER networks, the total number of links increases with the increase of the average degree as a step-like and sub-linear function. The overall consequence is that in a BA network, the presence of a growing relay network (i.e., an increase in the average degree of the network) will lead to a faster network but it comes at the cost of decentralization and minimum traffic handling overhead. But in an ER network, the trade-off is different as increasing the average number of connections improves the decentralization and network speed but increases the minimum traffic overhead.

5.4.4 Relay Network Size

We also wish to understand the effect of the relay network size on the overall decentralization degree of the network. To accomplish this goal, we conducted another set of experiments in which the overall size of the network was kept constant (at $N = 10000$), while varying the percentage of nodes participating in the relay network. We repeated the experiment for different amounts of the average degree of the network. The results are depicted in Figure 5.10. As

it can be seen, for a constant network size, when the size of the relay network tends to 50%, the network tends to be fully decentralized. For $M = 4$, for a relay network size of 15%, the decentralization degree is around 70%. For $M = 8$ and $M = 10$, this amount is around 55% and 45% respectively. As well, for $M = 16$ and $M = 32$, this amount goes down to around 40% and 30% respectively.

As a clear consequence, for a constant size of a blockchain network, a bigger percentage of the nodes participating as a relay node means a more decentralized network that is fairer.

5.5 Conclusion and Future works

In this chapter, we presented an analytical methodology for quantifying the decentralization degree in blockchain networks based on the peer selection strategy. To accomplish this, we implemented a complex blockchain network generator using two graph models: Barabási–Albert and Erdös-Rény. We analyzed and compared decentralization, the average shortest path as an indicator of the network speed, and the number of links as an indicator of minimum traffic handling overhead in blockchain networks with different architectures through extensive simulations. The obtained results disclosed that the decentralization degree of the network extremely depends on the topology and the architecture of the network. We have proven that the use of hubs and relay networks drastically reduces the decentralization degree of the network. Although increasing the number of connections per node can decrease the average shortest path and consequently decrease the block propagation delay, nevertheless in networks with deployed relay nodes it comes at cost of a reduced amount of decentralization.

CHAPTER 6

PERFORMANCE MODELING AND ANALYSIS OF HOTSTUFF FOR BLOCKCHAIN CONSENSUS

Byzantine Fault-Tolerant (BFT) protocols are classical algorithms that offer a faster and more energy-efficient consensus mechanism compared to Proof-of-Work (PoW), which is typically used by cryptocurrencies such as Bitcoin. Synchronous BFT systems are hard to implement and vulnerable to attacks that aim to disrupt the synchrony of the system. Practical BFT (PBFT), which is a partially synchronous protocol, is a high-performance consensus algorithm that provides strong safety in the presence of a bounded number of faulty participants. Hotstuff is one such partially synchronous BFT State Machine Replication (SMR) protocol that aims to address the aforementioned issues. PBFT is becoming a popular choice for blockchain consensus, especially in permissioned systems (e.g. Ripple, Stellar, etc). However, it is not well understood how Hotstuff, and PBFT consensus in general, behave under varying conditions that are commonly found in blockchain networks.

In this chapter, we present a theoretical model for the Hotstuff consensus mechanism which accurately predicts blockchain-related metrics such as the transaction throughput and expected confirmation time using important networking parameters such as the number of replicas, link latency, and packet loss. Furthermore, we validate our model through extensive simulations carried out using OMNeT++.

6.1 Introduction

Byzantine Fault-Tolerant (BFT) protocols are classical algorithms that offer a faster and more scalable consensus mechanism compared to PoW. The first BFT was introduced and formulated as the Byzantine generals problem by Lamport *et al*. (Lamport, Shostak & Pease, 1982). But the first synchronous solution was proposed in (Pease, Shostak & Lamport, 1980). Practical Byzantine Fault Tolerance (PBFT), which is a partially synchronous protocol, provides a high-performance consensus algorithm and was proposed by Liskov *et al*. in (Castro *et al.*, 1999). It

provides strong safety in the presence of a bounded number of faulty participants and is robust against Sybil attacks. The number of tolerable faulty replicas quite depends on the synchrony situation of the system. A synchronous protocol can guarantee safety when up to one-half of total replicas are faulty (Fitzi, 2002) while in asynchronous and partially synchronous protocols, safety is provided when up to one-third of total replicas are faulty (Dwork, Lynch & Stockmeyer, 1988). In practice, synchronous BFT systems are hard to implement since most of them need a large number of operation rounds with exactly synchronized participants (i.e. all the replicas must start each round and finish it at exactly the same time). As well these protocols are extremely vulnerable to attacks that aim to disrupt the synchrony of the system.

Hotstuff is one of the partially synchronous BFT State Machine Replication (SMR) protocols (Yin *et al.*, 2018) that aims to address the issues mentioned above. Hotstuff is capable of achieving both linear (and hence fast) view change as well as responsiveness (i.e. achieving the consensus at the speed of wire). To accomplish this, Hotstuff uses threshold signatures in a three-chain commit rule. Hotstuff can pipeline proposals in order to reach higher throughput. This means any leader can propose its own block without waiting for the parent block to get committed.

Hotstuff operates over a reliable and secure Peer-to-Peer (P2P) network of $N = 3f + 1$ replicas where f is the number of faulty replicas. The leader gets selected and proposes its own proposal and collects data from the replicas during an algorithm known as view change. Although a stable leader can drive the protocol to consensus in two phases of message exchanges (the first round for guaranteeing the proposal uniqueness and the second round for convincing replicas for voting for a proposal), nevertheless it is bug-prone (Abraham *et al.*, 2017) and far from simple (Mickens, 2014). Instead, Hotstuff can operate on three phases (excluding decide phase) that allow the leader to simply pick the highest Quorum Certificate (QC) that it knows (Yin *et al.*, 2018).

There are some researchers that study Hotstuff and its performance. A few of them introduce improved variants of Hotstuff from the performance view point (Abraham, Malkhi, Nayak, Ren & Yin, 2020; Cheng, 2022; bau; Jalalzai, Niu, Feng & Gai). But to the best of our knowledge,

there is no analytical performance model that can explain the relationship between the throughput of Hotstuff and architecture design and setup, especially, the number of replicas in the systems. In this chapter, we seek to address the aforementioned research gap by modeling the performance of the original HotStuff algorithm. As it is the principal component of a relatively new consensus system, it is imperative to analyze its performance in order to gain insights into the behavior and dynamics of the system, thereby facilitating the optimal configuration of the network without requiring iterative benchmarking. In this chapter, we present a theoretical model for the Hotstuff consensus mechanism and derive explicit mathematical equations to describe the dynamics and behavior of the system and identify the relationship between different metrics affecting the performance of the Hotstuff blockchain. we validate our theoretical model through extensive simulations using the network simulator OMNeT++, which is a discrete event simulator. As well, We estimate the throughput or the number of transactions per second (TPS) in Hotstuff. Furthermore, we identify the most effective parameters which impact its performance (e.g., network size, number of nodes, and transaction processing time) and show how the performance of Hotstuff is affected by the mentioned metrics.

The rest of this chapter is organized as follows: In Section 6.2, we present our proposed analytical approach for modeling the performance of Hotstuff. In Section 6.3, we validate our developed model through extensive simulations and study the performance of Hotstuff. Finally, Section 6.4 concludes chapter.

6.2 Proposed Analytical Model

This section presents the system model and the relationship between entities in the system. We derive explicit equations that explain the relationship between transaction throughput, latency, and network specifications.

6.2.1 System model

Network model: we assume a system of $N = 3f + 1$ replicas with the same processing power operating over a fully connected P2P network in which replicas are directly connected pairwise (i.e., each replica is directly connected to $N - 1$ replicas). Any replica with a missing connection should be considered a Byzantine node or a faulty replica. We assume transmission delay has an exponential distribution in P2P links with a mean of $1/\mu$. Although in reality, the latency of the links does not necessarily follow an exponential distribution, nevertheless developing a theoretical model without simplification assumptions will not be a trivial job since the relationship between parameters will become intractable. The exponential distribution is fully described in (Papoulis & Pillai, 2002) and has frequently been used for modeling asynchronous networks in literature (e.g. (Fukś, Lawniczak & Volkov, 2001; Sukhov, Astrakhantseva, Pervitsky, Boldyrev & Bukatov, 2016)). As well, we assume all of the messages reach to destination with a constant probability of $1 - \rho$ (or get lost or dropped with a constant probability of ρ). When a client submits a request to the network, it sets a timer with the amount of τ_o and waits. If it does not receive at least $f + 1$ `Decide` message for that request before the timer expires, it will abort and cancel the request. Note that replicas also set a timer in each phase, and can give up a certain phase if they do not receive the related messages within a predetermined interval of time. Hence, although the Hotstuff-based blockchain system can utilize TCP, (and thus all of the packets are expected to be delivered to the destination eventually) the packets should also arrive at their destination within a maximum delay of δ. Otherwise, they will be considered lost messages. If a replica's timer expires, it will send a time-out message instead of a positive/negative vote and will give up the phase. If a leader gathers time-out messages from a quorum of replicas, it will broadcast a time-out certificate (TC) instead of a quorum certificate and the round will be aborted. In this chapter, we treat a timed-out replica as a faulty node. Also, we assume all the replicas are acting independently.

Replica model: we assume a normal or honest replica always sends a valid response to any valid request. In contrast, we assume a faulty replica never sends a valid response to a valid request.

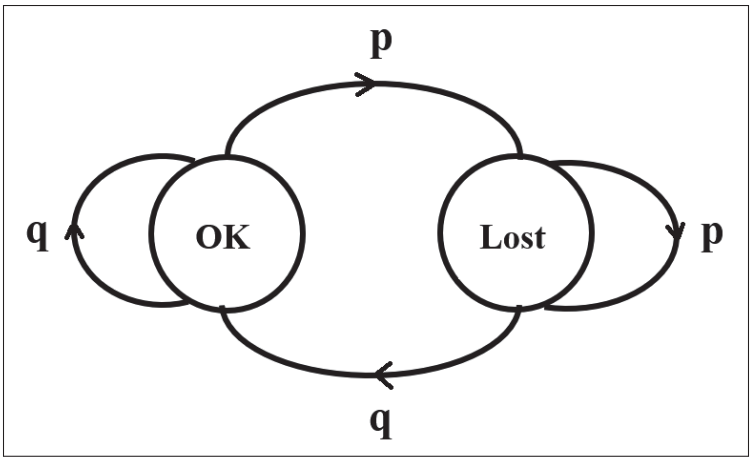


Figure 6.1 Bernoulli loss model
In this model, a certain packet gets lost with a probability of p and reaches its destination with a probability of q

Loss model: We assume the packet loss over all the links follows the Bernoulli loss model (Yajnik, Moon, Kurose & Towsley, 1999; Blanc, Avrachenkov & Collange, 2009) with a probability of p. This model is illustrated in Figure 6.1. Bernoulli's model is also a geometric distribution, which is a two-state model. In this model, one state (i.e. OK) is the state in which a certain packet reaches its destination with a probability of q, without getting lost or corrupted. The other state (i.e. Loss) represents a packet loss. In this state, a certain packet does not reach the destination due to a reason such as a packet drop or getting corrupted. Packet loss can either happen due to a packet drop in the queue or due to excessive delay. Therefore, the overall packet loss probability can be estimated as follows:

$$\rho = p + (1 - p)P\{\delta(k) \geq \tau\} \tag{6.1}$$

Where $\delta(k)$ is the network delay for packet k, and τ is the maximum tolerable network delay for this packet.

6.2.2 Analytical model

In order to consider the proposal as committed, the requester client needs to receive at least $f + 1$ `Decide` messages for the proposal that consists of that request (see Figure 6.2). The probability mass function of receiving at least d `Decide` messages by the client can be calculated as follows.

$$\mathbb{P}\{Y_D = d\} = \frac{N - f - 1}{N - 1} \sum_{d'=d}^{N} \mathbb{P}\{Y_D = d | X_D = d'\} \mathbb{P}\{X_D = d'\} \tag{6.2}$$

Where $\mathbb{P}\{X_D = d'\}$ is the probability mass function that d' replicas receive a `Decide` message for a certain view number. We use factor $\frac{N-f-1}{N-1}$ since apart from the leader, only $N - f - 1$ replicas send honest votes. $\mathbb{P}\{X_D = d'\}$ can be computed as follows

$$\mathbb{P}\{X_D = d'\} = \sum_{c=f+1}^{N} \mathbb{P}\{X_D = d' | Y_C = c\} \mathbb{P}\{Y_C = c\} \tag{6.3}$$

Where $\mathbb{P}\{Y_C = c\}$ is the probability mass function that the leader receives c positive `Commit` votes at the end of the commit phase. In order to finish the commit phase successfully, c must be equal to or greater than $2f + 1$ and can be computed as follows.

$$\mathbb{P}\{Y_C = c\} = \frac{N - f - 1}{N - 1} \sum_{c'=2f}^{N} \mathbb{P}\{Y_C = c | X_C = c'\} \mathbb{P}\{X_C = c'\} \tag{6.4}$$

In the equation above, $\mathbb{P}\{X_C = c'\}$ is the probability mass function that c' replicas receive `Commit` message from the leader during the commit phase for the same view number as above and can be computed as follows. It is to be noted that we assume the replicas always receive a self-message without any loss or corruption.

$$\mathbb{P}\{X_C = c'\} = \sum_{pc=2f}^{N} \mathbb{P}\{X_C = c' | Y_{PC} = pc\} \mathbb{P}\{Y_{PC} = pc\} \tag{6.5}$$

Where, $\mathbb{P}\{Y_{PC} = pc\}$ is the probability mass function of receiving pc `Pre-commit` votes by the leader and in the same way can be computed as below

$$\mathbb{P}\{Y_{PC} = pc\} = \frac{N-f-1}{N-1} \sum_{p'c'=pc}^{N} \mathbb{P}\{Y_{PC} = pc | X_{PC} = p'c'\}\mathbb{P}\{X_{PC} = p'c'\} \tag{6.6}$$

Where, $\mathbb{P}\{X_{PC} = p'c'\}$ is the probability mass function that $p'c'$ replicas receive the `Pre-commit` message from the leader during the same view number as above and can be computed as follows.

$$\mathbb{P}\{X_{PC} = p'c'\} = \sum_{p=2f}^{N} \mathbb{P}\{X_{PC} = p'c' | Y_P = p\}\mathbb{P}\{Y_P = p\} \tag{6.7}$$

Similarly, $\mathbb{P}\{Y_P = p\}$ is the probability mass function of receiving p `Prepare` votes by the leader at the end of prepare phase. It can be calculated as follows.

$$\mathbb{P}\{Y_P = p\} = \frac{N-f-1}{N-1} \sum_{p'=p}^{N} \mathbb{P}\{Y_P = p | X_P = p'\}\mathbb{P}\{X_P = p'\} \tag{6.8}$$

Where $\mathbb{P}\{X_P = p'\}$ is the probability mass function that p' replicas receive the `Prepare` messages from the leader during the prepare phase and can be calculated as follows

$$\mathbb{P}\{X_P = p'\} = \frac{N-f-1}{N-1} \binom{N-1}{p'} (1-\rho)^{p'+1} \rho^{N-p'-1} \tag{6.9}$$

Where ρ is calculated from the Equation 6.1. It is to be noted that in the equation above, $\frac{N-f-1}{N-1}(1-\rho)$ is the probability that the leader is selected from the honest nodes and receives the request from the client successfully via a P2P link.

In Equation 6.2, $\mathbb{P}\{Y_D = d | X_D = d'\}$ can be rewritten as follows.

$$\mathbb{P}\{Y_D = d | X_D = d'\} = \binom{d'}{d}(1-\rho)^d \rho^{d'-d} \tag{6.10}$$

In the same way in Equation 6.4

$$\mathbb{P}\{Y_C = c | X_C = c'\} = \binom{c'}{c}(1-\rho)^c \rho^{c'-c} \tag{6.11}$$

Similarly in Equation 6.6

$$\mathbb{P}\{Y_{PC} = pc | X_{PC} = p'c'\} = \binom{p'c'}{pc}(1-\rho)^p c \rho^{p'c'-pc} \tag{6.12}$$

And in Equation 6.8

$$\mathbb{P}\{Y_P = p | X_P = p'\} = \binom{p'}{p}(1-\rho)^p \rho^{p'-p} \tag{6.13}$$

So far, we are able to calculate the probability of reaching a consensus successfully (denoted as P_s). Consensus happens when the client receives $f + 1$ or more `Decide` votes from the replicas. Therefore,

$$P_s = \sum_{i=f+1}^{N} \mathbb{P}\{Y_D = i\} \tag{6.14}$$

For estimating the transaction throughput, we need to calculate the mean time of reaching a consensus as follows:

$$\mathbb{E}[T_c] = \mathbb{E}[T_s] + (1 - p_s)\tau_o \tag{6.15}$$

where $\mathbb{E}[T_c]$ and $\mathbb{E}[T_s]$ are the expected values of time for reaching a consensus (consist of all successful and unsuccessful attempts) and the expected value of the time for a successful consensus attempt, respectively. As well, τ_o is the value of the timer set by the clients as a time-out value. We assume all the clients always set the same amount for the timer.

Assume n_p, n_{pc}, n_c and n_d are the number of replicas that receive `Prepare`, `Pre-commit`, `Commit`, `Decide` messages from the leader of a certain view during phases prepare, pre-commit, commit, and decide respectively (see the Figure 6.2).

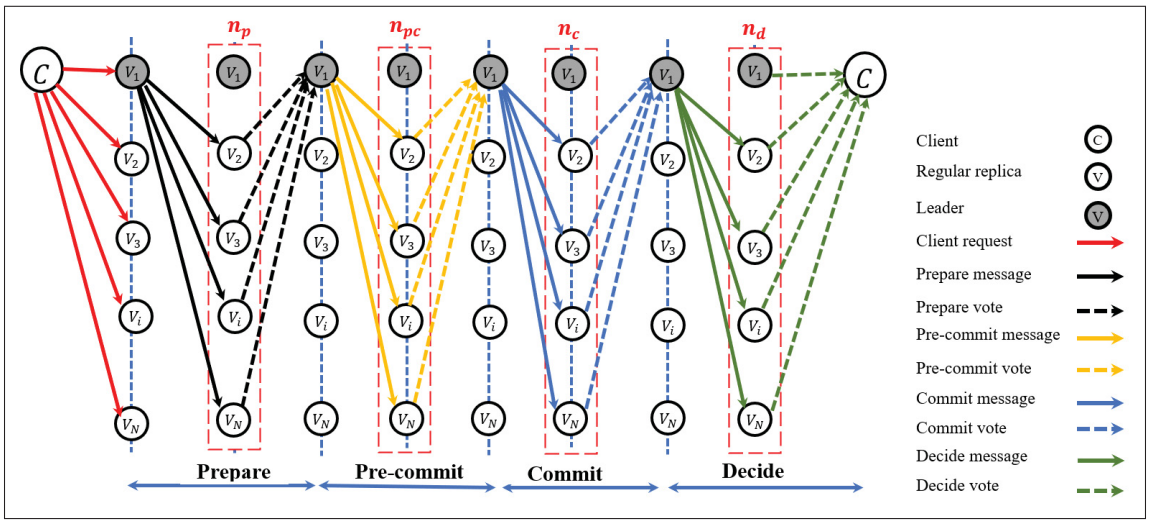


Figure 6.2 Overview of the Hotstuff protocol: Block proposals require 4 consecutive phases to be committed

The process of sending the proposal from the leader to replicas in each round can be considered as an asynchronous process since leaders broadcast the proposal to validators at almost the same time. Clearly,

$$n_p = \frac{N-f-1}{N-1}(1-\rho)(N-1) = (N-f-1)(1-\rho) \tag{6.16}$$

In the equation above, $\frac{N-f-1}{N-1}$ is the probability that the leader gets selected from honest nodes. As well,

$$n_{pc} = n_c = n_d = (N-1)(1-\rho) \tag{6.17}$$

Assume M nodes are sending messages to the same destination with an exponentially distributed delay. The expected time to receive i messages at the destination can be calculated as follows (Papoulis & Pillai, 2002):

$$\mathbb{E}[T_M] = \frac{1}{\mu} \sum_{j=0}^{i-1} \frac{1}{M-j} \tag{6.18}$$

where μ is the parameter of the exponential distribution (e.g. μ^{-1}= mean P2P delay between the nodes). Hence,

$$\mathbb{E}[T_{n_p}] = \frac{1}{\mu} \sum_{i=0}^{2f-1} \frac{1}{n_p - i} \tag{6.19}$$

where $\mathbb{E}[T_{n_p}]$ is the expected time takes for the leader to receive $2f$ valid `Prepare` votes from the n_p replicas at the end of prepare phase. It is to be noted that we assume the leader receives his vote to himself in a very small and negligible amount of time. In the same way we can write:

$$\mathbb{E}[T_{n_{pc}}] = \frac{1}{\mu} \sum_{i=0}^{2f-1} \frac{1}{n_{pc} - i} \tag{6.20}$$

$$\mathbb{E}[T_{n_c}] = \frac{1}{\mu} \sum_{i=0}^{2f-1} \frac{1}{n_c - i} \tag{6.21}$$

$$\mathbb{E}[T_{n_d}] = \frac{1}{\mu}\sum_{i=0}^{f-1}\frac{1}{n_d - i} \tag{6.22}$$

we are now able to calculate $\mathbb{E}[T_s]$ as follows:

$$\mathbb{E}[T_s] = \frac{3}{\mu} + \mathbb{E}[T_p] + \frac{1}{\mu}(\sum_{i=0}^{2f-1}\frac{1}{n_p - i}\sum_{i=0}^{2f-1}\frac{1}{n_{pc} - i} + \sum_{i=0}^{2f-1}\frac{1}{n_c - i}) \tag{6.23}$$

where $\mathbb{E}[T_p]$ is the mean proposal (i.e. block) processing time and is the time each replica spends to check the proposal and prepare a vote. Now we can calculate $\mathbb{E}[T_c]$ from Equation (6.15). The throughput of the system can be calculated simply as follows:

$$\gamma = \frac{1}{\mathbb{E}[T_c]} \tag{6.24}$$

It is to be noted that in order to calculate $\mathbb{E}[T_s]$, we only considered the time interval from the instant the leader starts the prepare phase until it receives $2f + 1$ commit votes from the replicas. However, the latency experienced by the client to get the response from the network of replicas can be calculated as follows.

$$\mathbb{E}[T_D] = \mathbb{E}[T_s] + \frac{1}{\mu}(2 + \sum_{i=0}^{f-1}\frac{1}{n_d - i}) \tag{6.25}$$

The throughput of the system from the client's view point can be calculated simply as follows.

$$\gamma_c = \frac{1}{\mathbb{E}[T_D]} \tag{6.26}$$

6.3 Model Validation and Analysis

In this section, we validate our theoretical model by comparing it with simulation results and explain the configuration we set up for simulation. We then discuss the results and the impact on the expected performance of the basic Hotstuff network, as well as potential trade-offs.

Implementation: We used $OMNeT++$ (OmNet) as a discrete event-based simulator of the network. As well, we used $Matlab$ for the calculation and analysis of the theoretical model. We used NED language for describing the network topology and assigning values to networking parameters (e.g. p2p latency, network size, etc). We implemented the basic Hotstuff mechanism in each module using $C++$. For each experiment, we changed random number seeds using an *ini* file.

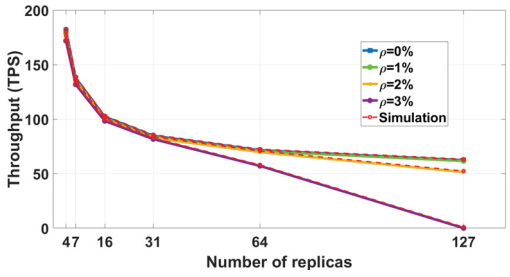
Settings: In this chapter, we carry out the experiments for a variety of network sizes but limit our experiments to 127 replicas since for bigger networks throughput becomes very low and as we will see, in some conditions, it tends to zero. In each experiment, we considered a network with a size of N as $N = 3f + 1$ (therefore $N - 1$ is divisible by 3). We started with the smallest possible network ($N = 4$ and $f = 1$) and doubled the network size in each of the next experiments.

For all experiments, we assumed each proposal consists of only one transaction. Block processing time for blocks with more than one transaction may not necessarily increase linearly with increasing the number of transactions embedded in the block since it depends on the code, the number of threads, and the size of the block header. Moreover, we assumed all the replicas have the same processing power and consequently the same processing delay as validators. In OMNeT++, a processing delay can be simulated using a self message scheduled for a certain event.

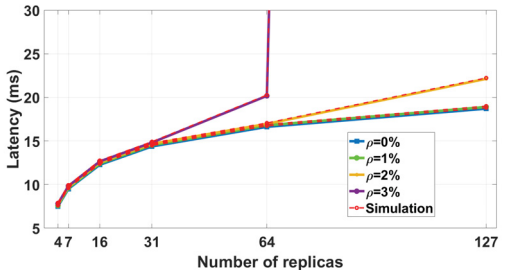
We set a direct P2P link between any two arbitrary validators. Hence, the network is a complete graph. We set a random P2P link delay for each experiment with an exponential distribution with a mean amount of $1/\mu = 1ms$. Assuming an exponential distribution for link delay is common and adequate for many use cases (Hassan & Jain, 2003). In OMNeT++, one can create

a channel between two arbitrary modules in the Ned file with an exponential distribution and assign the mean value.

In order to simulate parameter ρ, we implemented all the links as ideal links without any packet loss, drop, or corruption. Instead, in the receiver module, packets get deleted upon arrival with a probability of ρ or go to the next level with a probability of $1 - \rho$. Finally, we assumed that Byzantine or faulty nodes can either send invalid responses or do not send any vote to `Prepare, Pre-commit, and Commit` messages. In addition, we assumed that receivers always delete invalid messages. In the simulation code, faulty nodes always send an invalid response that gets deleted in the destination node. Furthermore, the replicas whose valid votes get deleted due to packet loss are also treated as faulty nodes.



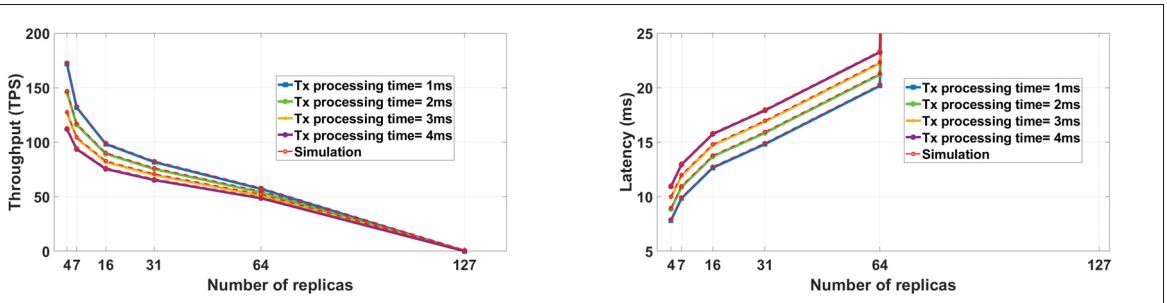
a) Throughput vs. network size for different values of packet loss



b) Latency vs. network size for different values of packet loss

Figure 6.3 Theoretical vs. simulation results

Results and discussion: In Figure 6.3a and Figure 6.3b, we depict the results of our experiments for transaction throughput and latency versus the network size (i.e. the number of replicas) for different amounts of packet loss percentage. For this experiment, we kept the transaction processing time equal to $1ms$. We started with a loss-less network in which $\rho = 0$ and therefore every packet reaches to its destination without getting lost or corrupted. We increased the amount of packet lost by one percent in each of the next experiments while keeping every other metric constant. First, as we can see, the simulation results closely follow the theoretical amounts. Second, as is expected, when increasing the number of replicas, throughput goes down and



a) Throughput vs. network size for different values of transaction processing time

b) Throughput vs. network size for different values of transaction processing time

Figure 6.4 Theoretical vs. simulation results

latency time goes up. As it can be seen, with a higher packet loss probability, throughput and latency get worse faster when increasing the number of replicas. When $\rho = 3\%$ and $N = 127$, the throughput tends to zero, and the latency tends to infinite. This means that the consensus never happens and the client aborts the request since the timer is expired.

In the next set of experiments, we kept $\rho = 3\%$ and changed the transaction processing time in each replica from $1ms$ to $4ms$. Results are depicted in Figure 6.4a and Figure 6.4b. Obviously, increasing the transaction processing time has a negative impact on both throughput and latency. Increasing the transaction processing time has a stronger effect on the throughput of smaller networks while in the larger networks the effect of increased transaction processing time can be negligible. In addition, when $N = 127$, the latency tends to the infinite. This implies that the consensus attempt failed and the client aborted the request.

6.4 Conclusion

In this chapter, we proposed a theoretical performance model for the basic Hotstuff consensus algorithm and validated it with a set of simulations carried out using OMNeT++. As well, we presented a performance analysis and studied the system throughput for a variety of network sizes with different amounts of the packet loss ratio and transaction processing time. We observed

that the throughput is highly dependent on the number of replicas, transaction processing time, and packet loss ratio in the network.

Setting a too low amount of time-out in clients can lead them to frequently abort their queries while setting a too high amount of time-out may slow down the network and consequently reduce the throughput. Moreover, setting an inappropriate time-out amount in validators can slow down the network in another way. Our future work is to leverage this model to dynamically determine the optimal timeout value for the clients and validators in order to maximize the throughput. Our future work is to estimate the optimal value of the time-out set in the clients regarding the network specifications. As well, we will extend our existing performance model to the pipelined Hotstuff.

CONCLUSION AND RECOMMENDATIONS

In this chapter, we conclude and summarize the findings of this research and present future works.

7.1 Summary

In this thesis, we developed analytical models for the analysis of blockchain networks. To accomplish this, we studied blockchain networks in two separate categories of the public (permission-less) and private (permissioned) blockchains. We exploited these models to investigate the impact of network topology and key blockchain parameters on three different aspects of blockchain networks in terms of performance, security (forks), and decentralization.

In Chapter 3, we presented a performance model for Bitcoin as the representative of public PoW-based blockchains, We proposed a random graph model for performance modeling and analysis of the inventory-based protocol for block dissemination in the Bitcoin network. This model addresses the impact of key blockchain parameters on the overall performance of Bitcoin. We derived some explicit and closed-form equations for block propagation delay and traffic overhead in the Bitcoin network. We also adapted our model to study the impact of deploying a relay network and investigate the effect of the relay network size on the network performance and decentralization of PoW-based blockchains. We implemented our model using the popular network simulator OMNet++. We validated the accuracy of the proposed theoretical model and its implementation with the dataset mined from the Bitcoin network. Our results show the trade-off between the default number of connections per node, network bandwidth, and block size in order to compute the optimal block propagation delay over the network. Although the throughput of Bitcoin can be increased by choosing a bigger size for blocks, this can cause a significant increase in the block propagation time. The delay can be reduced by increasing the average default number of connections per node but this has the drawback of increased traffic

overhead in the network. As well, we found and proved that bigger relay networks can jeopardize the decentralization of the Bitcoin network.

In Chapter 4, we investigated the cause and length of forks for the Bitcoin network. We developed theoretical formulas which model the Bitcoin consensus and formation of forks in the network, based on an Erdös-Rényi random graph construction of the overlay network of peers. This model was also validated with simulation and historical data gathered from the Bitcoin network. We implemented this model using the popular network simulator OMNet++. We validated the accuracy of the proposed theoretical model and its implementation with the dataset mined from the Bitcoin network. We leveraged this model to estimate the weight of fork branches. We showed that under current conditions, Bitcoin will not benefit from increasing the number of connections per node. furthermore, we showed that reducing the block time compromises the security of the blockchain by increasing the probability of a fork. The average number of connections per node currently has no impact on the probability of forks, since Bitcoin currently operates within a stable range of 22-99 connections. In addition, we investigate the impact of the time difference between two concurrent blocks and the average number of connections per node on the weight of fork branches. If the later miner starts to propagate its block too late and the number of connections per node is sufficiently high, the impact of the fork on the network is almost negligible.

In Chapter 5, we presented a methodology for quantifying the decentralization degree of a blockchain network. To accomplish this, we used two well-known graph models of Erdös-Rény and Barabási–Albert in order to study the blockchain network topology. We then used the clustering coefficient of the network models to quantify the decentralization degree. We analyzed and compared decentralization, the average shortest path as an indicator of the network speed, and the number of links as an indicator of minimum traffic handling overhead in blockchain networks with different architectures through extensive simulations. The obtained results

disclosed that the decentralization degree of the network extremely depends on the topology and the architecture of the network. We have proven that the use of hubs and relay networks drastically reduces the decentralization degree of the network. Although increasing the number of connections per node can decrease the average shortest path and consequently decrease the block propagation delay, nevertheless in networks with deployed relay nodes it comes at cost of a reduced amount of decentralization.

In Chapter 6, we presented a theoretical model for the Hotstuff consensus mechanism as the representative of private blockchains. This model was able to accurately predict blockchain-related metrics such as the transaction throughput and expected confirmation time using important networking parameters such as the number of replicas, link latency, and packet loss. In this chapter we found setting a too low amount of time-out in clients can lead them to frequently abort their queries while setting a too high amount of time-out may slow down the network and consequently reduce the throughput. Moreover, we showed that setting an inappropriate time-out amount in validators can slow down the network in another way.

7.2 Future works

In this thesis, we presented significant contributions in the performance modeling and analysis of blockchain networks. Nevertheless, there exist several research gaps as well as possible extensions from our previous works.

Bitcoin and Ethereum have almost the same information propagation mechanism. Nevertheless, they have still considerable differences. In future work, the presented model for Bitcoin can be adopted for Ethereum and it will not be a trivial job. Another possible future work is to extend our fork model for the analysis of eclipse attacks in blockchain networks. As well, the mentioned model can be leveraged jointly with our methodology for quantifying decentralization

in blockchain networks to study the impact of decentralization of the blockchain networks on the fork probability.

Our future work in the context of private blockchains is to leverage the developed model for Hotstuff to dynamically determine the optimal timeout value for the clients and validators in order to maximize the throughput. In addition, it can be adapted for other BFT-based protocols.

BIBLIOGRAPHY

State machine replication in the Libra Blockchain. *https://diem-developers-components.netlify.app/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2020-05-26.pdf*. Accessed: 20-05-2022.

(2018). Bitnodes API v1.0. *available at https://bitnodes.earn.com*.

(2019). Satoshi Client Node Discovery. *https://en.bitcoin.it/wiki/Satoshi*$_{C}lient_{N}ode_{D}iscovery$.

(2021). Bitnodes. *See https://bitnodes.io/dashboard/?days=1825*.

Abraham, I., Gueta, G., Malkhi, D., Alvisi, L., Kotla, R. & Martin, J.-P. (2017). Revisiting fast practical byzantine fault tolerance. *arXiv preprint arXiv:1712.01367*.

Abraham, I., Malkhi, D., Nayak, K., Ren, L. & Yin, M. (2020). Sync hotstuff: Simple and practical synchronous state machine replication. *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 106–118.

Agrawal, S. & Daudjee, K. (2016). A performance comparison of algorithms for byzantine agreement in distributed systems. *2016 12th European Dependable Computing Conference (EDCC)*, pp. 249–260.

Albert, R. & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1), 47.

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., McCallum, P. & Peacock, A. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100, 143–174.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y. et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. *EuroSys*, pp. 1–15.

Antonopoulos, A. M. (2017). *Mastering Bitcoin: Programming the open blockchain*. " O'Reilly Media, Inc.".

Aoki, Y. & Shudo, K. (2019). Proximity neighbor selection in blockchain networks. *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 52–58.

Atlam, H. F., Alenezi, A., Alassafi, M. O. & Wills, G. (2018). Blockchain with Internet of Things: Benefits, challenges, and future directions. *International Journal of Intelligent Systems and Applications*, 10(6), 40–48.

Atzei, N., Bartoletti, M. & Cimoli, T. (2017). A survey of attacks on ethereum smart contracts (sok). *International conference on principles of security and trust*, pp. 164–186.

Atzori, M. (2015). Blockchain technology and decentralized governance: Is the state still necessary? *Available at SSRN 2709713*.

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. *Available: https://blockstream.com/sidechains.pdf*.

Bai, Q., Zhou, X., Wang, X., Xu, Y., Wang, X. & Kong, Q. (2018). A Deep Dive into Blockchain Selfish Mining. *arXiv preprint arXiv:1811.08263*.

Banafa, A. (2017). IoT and Blockchain Convergence: Benefits and Challenges. *IEEE Internet of Things*.

Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S. & Danezis, G. (2019). SoK: Consensus in the age of blockchains. *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pp. 183–198.

Barabási, A.-L. & Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439), 509–512.

Basu, S., Eyal, I. & Sirer, E. G. (2019). Falcon. *See https://www.falcon-net.org*.

Beikverdi, A. & Song, J. (2015). Trend of centralization in Bitcoin's distributed network. *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 1–6.

Benčić, F. M. & Žarko, I. P. (2018). Distributed ledger technology: Blockchain compared to directed acyclic graph. *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1569–1570.

Berger, C. & Reiser, H. P. (2018). Scaling Byzantine consensus: A broad analysis. *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, pp. 13–18.

Bessani, A., Sousa, J. & Alchieri, E. E. (2014). State machine replication for the masses with BFT-SMART. *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 355–362.

Biais, B., Bisiere, C., Bouvard, M. & Casamatta, C. (2018). The blockchain folk theorem. *Swiss Finance Institute Research Paper*, (17-75).

Biryukov, A., Khovratovich, D. & Pustogarov, I. (2014). Deanonymisation of clients in Bitcoin P2P network. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 15–29.

BitcoinWiki. Bitcoin Wiki: Confirmation. *available at https://en.bitcoin.it/wiki/Confirmation*.

Blanc, A., Avrachenkov, K. & Collange, D. (2009). Comparing some high speed TCP versions under bernoulli losses. *Proc. International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports*, pp. 59–64.

Bollobás, B. & Riordan, O. M. (2003). Mathematical results on scale-free random graphs. *Handbook of graphs and networks: from the genome to the internet*, 1–34.

Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A. & Felten, E. W. (2015). Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. *2015 IEEE Symposium on Security and Privacy*, pp. 104–121.

Bowden, R., Keeler, H. P., Krzesinski, A. E. & Taylor, P. G. (2018). Block arrivals in the Bitcoin blockchain. *CoRR*, abs/1801.07447.

Bradbury, D. (2013). The problem with Bitcoin. *Computer Fraud & Security*, 2013(11), 5–8.

Brandenburger, M., Cachin, C., Kapitza, R. & Sorniotti, A. (2018). Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric. *arXiv preprint arXiv:1805.08541*.

Buterin, V. et al. (2013). Ethereum white paper, 2014. *URL https://github.com/ethereum/wiki/wiki/White-Paper*.

Cai, W., Wang, Z., Ernst, J. B., Hong, Z., Feng, C. & Leung, V. C. (2018). Decentralized applications: The blockchain-empowered software system. *IEEE Access*, 6, 53019–53033.

Carbajo, R. S. & Goldrick, C. M. (2017). Decentralized peer-to-peer data dissemination in Wireless Sensor Networks.

Castro, M., Liskov, B. et al. (1999). Practical byzantine fault tolerance. *OSDI*, 99(1999), 173–186.

Castro, M., Druschel, P., Hu, Y. C., Rowstron, A. et al. (2002). *Exploiting network proximity in peer-to-peer overlay networks*.

Cheng, T. (2022). High Performance Consensus without Duplication: Multi-pipeine Hotstuff. *arXiv preprint arXiv:2205.04179*.

Christopher, C. M. (2016). The Bridging Model: Exploring the Roles of Trust and Enforcement in Banking, Bitcoin, and the Blockchain. *Nev. LJ*, 17, 139.

Chu, S. & Wang, S. (2018). The curses of blockchain decentralization. *arXiv preprint arXiv:1810.02937*.

Conti, M., E, S. K., Lal, C. & Ruj, S. (2018a). A Survey on Security and Privacy Issues of Bitcoin. *IEEE Communications Surveys Tutorials*, 1-1. doi: 10.1109/COMST.2018.2842460.

Conti, M., Kumar, E. S., Lal, C. & Ruj, S. (2018b). A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4), 3416–3452.

Corallo, M. (2016a). Bitcoin relay network. *See http://bitcoinrelaynetwork.org*.

Corallo, M. (2016b). BIP 152: compact block relay. *See https://github.com/bitcoin/bips/blob/master/bip-0152. mediawiki*.

Corallo, M. (2019). Bitcoin FIBRE Network. *See http://bitcoinfibre.org/public-network.html*.

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. *International conference on financial cryptography and data security*, pp. 106–125.

Dai, H.-N., Zheng, Z. & Zhang, Y. (2019). Blockchain for Internet of Things: A Survey. *IEEE Internet of Things Journal*, 6(5), 8076-8094.

Decker, C. & Wattenhofer, R. (2013). Information propagation in the bitcoin network. *IEEE International Conference on Peer-to-Peer Computing*, pp. 1–10.

Distler, T. (2021). Byzantine Fault-tolerant State-machine Replication from a Systems Perspective. *ACM Computing Surveys (CSUR)*, 54(1), 1–38.

Donet, J. A. D., Pérez-Sola, C. & Herrera-Joancomartí, J. (2014). The bitcoin P2P network. *International Conference on Financial Cryptography and Data Security (FC)*, pp. 87–102.

Dorri, A., Kanhere, S. S. & Jurdak, R. (2016). Blockchain in internet of things: Challenges and Solutions. *Computing Research Repository (CoRR)*, abs/1608.05187.

Dwork, C., Lynch, N. & Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2), 288–323.

Erdős, P. & Rényi, A. (1959). On Random Graphs I. *Publicationes Mathematicae (Debrecen)*, 6, 290-297.

Erdős, P., Rényi, A. et al. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1), 17–60.

Estrada, E. (2016). When local and global clustering of networks diverge. *Linear Algebra and its Applications*, 488, 249–263.

Eyal, I. & Sirer, E. G. (2014). Majority is not enough: Bitcoin mining is vulnerable. *International conference on financial cryptography and data security*, pp. 436–454.

Eyal, I., Gencer, A. E., Sirer, E. G. & Van Renesse, R. (2016). Bitcoin-NG: A Scalable Blockchain Protocol. *Proceedings of the Usenix Conference on Networked Systems Design and Implementation (NSDI)*, pp. 45–59.

Fadhil, M., Owenson, G. & Adda, M. (2016). A Bitcoin model for evaluation of clustering to improve propagation delay in Bitcoin network. *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pp. 468–475.

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

Fanning, K. & Centers, D. P. (2016). Blockchain and its coming impact on financial services. *Journal of Corporate Accounting & Finance*, 27(5), 53–57.

Fitzi, M. (2002). *Generalized communication and security models in Byzantine agreement*. (Ph.D. thesis, ETH Zurich).

Fu, X., Wang, H. & Shi, P. (2021). A survey of Blockchain consensus algorithms: mechanism, design and applications. *Science China Information Sciences*, 64(2), 1–15.

Fukś, H., Lawniczak, A. T. & Volkov, S. (2001). Packet delay in models of data networks. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 11(3), 233–250.

Garay, J., Kiayias, A. & Leonardos, N. (2015). The bitcoin backbone protocol: Analysis and applications. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 281–310.

Ge, L., Brewster, C., Spek, J., Smeenk, A., Top, J., van Diepen, F., Klaase, B., Graumans, C. & de Wildt, M. d. R. (2017). *Blockchain for agriculture and food: Findings from the pilot study*. Wageningen Economic Research.

Gencer, A. E., Basu, S., Eyal, I., van Renesse, R. & Sirer, E. G. (2018). Decentralization in bitcoin and ethereum networks. *arXiv preprint arXiv:1801.03998*.

Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H. & Capkun, S. (2016). On the security and performance of proof of work blockchains. *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 3–16.

Gordon, W. J. & Catalini, C. (2018). Blockchain Technology for Healthcare: Facilitating the Transition to Patient-Driven Interoperability. *Computational and structural biotechnology journal*, 16, 224–230.

Gueta, G. G., Abraham, I., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M., Seredinschi, D.-A., Tamir, O. & Tomescu, A. (2019). SBFT: a scalable and decentralized trust infrastructure. *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 568–580.

Hao, W., Zeng, J., Dai, X., Xiao, J., Hua, Q., Chen, H., Li, K.-C. & Jin, H. (2019). BlockP2P: Enabling Fast Blockchain Broadcast with Scalable Peer-to-Peer Network Topology. *International Conference on Green, Pervasive, and Cloud Computing*, pp. 223–237.

Hao, Y., Li, Y., Dong, X., Fang, L. & Chen, P. (2018). Performance analysis of consensus algorithm in private blockchain. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 280–285.

Hartmann, A. K. & Mézard, M. (2018). Distribution of diameters for Erdős-Rényi random graphs. *Physical Review E*, 97(3), 032128.

Hassan, M. & Jain, R. (2003). *High performance TCP/IP networking*. Prentice Hall Upper Saddle River, NJ.

Imtiaz, M. A., Starobinski, D., Trachtenberg, A. & Younis, N. (2019). Churn in the Bitcoin Network: Characterization and impact. *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 431–439.

Jalalzai, M. M., Niu, J., Feng, C. & Gai, F. Fast-HotStuff: A Fast and Robust BFT Protocol for Blockchains.

Jalalzai, M. M., Busch, C. & Richard, G. G. (2019). Proteus: a scalable BFT consensus protocol for blockchains. *2019 IEEE international conference on Blockchain (Blockchain)*, pp. 308–313.

Jansen, R. & Hopper, N. (2012, February). Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. *Proceedings of the 19th Symposium on Network and Distributed System Security (NDSS)*.

Klarman, U., Basu, S., Kuzmanovic, A. & Sirer, E. G. (2018). bloXroute: A Scalable Trustless Blockchain Distribution Network WHITEPAPER.

Kumar, R. & Ross, K. W. (2006). Optimal peer-assisted file distribution: Single and multi-class problems. *Proceedings of IEEE Workshop on Hot Topics in Web Systems and Technologies (HOTWEB)*.

Kwon, Y., Kim, D., Son, Y., Vasserman, E. & Kim, Y. (2017). Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 195–209.

Kwon, Y., Liu, J., Kim, M., Song, D. & Kim, Y. (2019). Impossibility of full decentralization in permissionless blockchains. *ACM AFT*, pp. 110–123.

Lamport, L., Shostak, R. & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3), 382–401.

Lange, F., Ballet, G. & Toulme, A. (2016). Ethereum Wire Protocol. *See https://github.com/ethereum/devp2p/blob/master/caps/eth.md*.

Levy, K. E. (2017). Book-smart, not street-smart: blockchain-based smart contracts and the social workings of law. *Engaging Science, Technology, and Society*, 3, 1–15.

Li, C. & Palanisamy, B. (2020). Comparison of decentralization in dpos and pow blockchains. *International Conference on Blockchain*, pp. 18–32.

Li, W., Feng, C., Zhang, L., Xu, H., Cao, B. & Imran, M. A. (2020). A scalable multi-layer pbft consensus for blockchain. *IEEE Transactions on Parallel and Distributed Systems*, 32(5), 1146–1160.

Li, W. & He, M. (2020). Comparative Analysis of Bitcoin, Ethereum, and Libra. *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 545–550.

Lin, I.-C. & Liao, T.-C. (2017). A Survey of Blockchain Security Issues and Challenges. *IJ Network Security*, 19(5), 653–659.

Lin, M.-J. & Marzullo, K. (1999). Directional gossip: Gossip in a wide area network. *European Dependable Computing Conference (EDCC)*, pp. 364–379.

Loruenser, T., Rainer, B. & Wohner, F. (2021). Towards a Performance Model for Byzantine Fault Tolerant (Storage) Services. *arXiv preprint arXiv:2101.04489*.

Merkle, R. C. (1987). A digital signature based on a conventional encryption function. *Conference on the theory and application of cryptographic techniques*, pp. 369–378.

Meshcheryakov, Y., Melman, A., Evsutin, O., Morozov, V. & Koucheryavy, Y. (2021). On performance of PBFT blockchain consensus algorithm for IoT-applications with constrained devices. *IEEE Access*, 9, 80559–80570.

Mickens, J. (2014). The saddest moment. *Login Usenix Mag*, 39(3), 52–54.

Miller, A. & Jansen, R. (2015). Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications. *8th Workshop on Cyber Security Experimentation and Test ({CSET} 15)*.

Misic, J., Misic, V. B., Chang, X., Motlagh, S. G. & Ali, Z. M. (2019). Modeling of Bitcoin's blockchain delivery network. *IEEE Transactions on Network Science and Engineering*.

Momose, A. & Ren, L. (2020). Optimal communication complexity of byzantine consensus under honest majority. *arXiv e-prints*, arXiv–2007.

Mundinger, J., Weber, R. & Weiss, G. (2006). Analysis of peer-to-peer file dissemination. *ACM SIGMETRICS Performance Evaluation Review*, 34(3), 12–14.

Nagayama, R., Shudo, K. & Banno, R. (2019). Simulation of the Bitcoin Network Considering Compact Block Relay and Internet Improvements. *arXiv preprint arXiv:1912.05208*.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. Retrieved from: http://www.bitcoin.org/bitcoin.pdf.

Narayanan, A., Bonneau, J., Felten, E., Miller, A. & Goldfeder, S. (2016). *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ, USA: Princeton University Press.

Nasir, Q., Qasse, I. A., Abu Talib, M. & Nassif, A. B. (2018). Performance Analysis of Hyperledger Fabric Platforms. *Security and Communication Networks*, 2018.

Neudecker, T. & Hartenstein, H. (2019 (accessed March 04, 2019)). "Short Paper: An Empirical Analysis of Blockchain Forks in Bitcoin". Retrieved from: https://dsn.tm.kit.edu/bitcoin/forks.

Neudecker, T., Andelfinger, P. & Hartenstein, H. (2015). A simulation model for analysis of attacks on the bitcoin peer-to-peer network. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1327–1332.

Neudecker, T., Andelfinger, P. & Hartenstein, H. (2016). Timing analysis for inferring the topology of the bitcoin peer-to-peer network. *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pp. 358–367.

Nguyen, C. T., Hoang, D. T., Nguyen, D. N., Niyato, D., Nguyen, H. T. & Dutkiewicz, E. (2019). Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE Access*, 7, 85727–85745.

Nguyen, G.-T. & Kim, K. (2018). A Survey about Consensus Algorithms Used in Blockchain. *Journal of Information processing systems*, 14(1).

Niu, J., Gai, F., Jalalzai, M. M. & Feng, C. (2021). On the performance of pipelined HotStuff. *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10.

Oikonomou, K. & Stavrakakis, I. (2007). Performance analysis of probabilistic flooding using random graphs. *IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6.

OmNet. Omnet++ simulator. *available at http://www. omnetpp. org*.

Otsuki, K., Aoki, Y., Banno, R. & Shudo, K. (2019). Effects of a simple relay network on the bitcoin network. *Proceedings of the Asian Internet Engineering Conference*, pp. 41–46.

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

Ozisik, A. P., Andresen, G., Bissias, G., Houmansadr, A. & Levine, B. (2017). Graphene: A new protocol for block propagation using set reconciliation. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (pp. 420–428). Springer.

Papoulis, A. & Pillai, S. U. (2002). *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education.

Pass, R., Seeman, L. & Shelat, A. (2017). Analysis of the blockchain protocol in asynchronous networks. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 643–673.

Pease, M., Shostak, R. & Lamport, L. (1980). Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2), 228–234.

Perboli, G., Musso, S. & Rosano, M. (2018). Blockchain in logistics and supply chain: A lean approach for designing real-world use cases. *Ieee Access*, 6, 62018–62028.

Puthal, D., Malik, N., Mohanty, S. P., Kougianos, E. & Yang, C. (2018). The blockchain as a decentralized security framework [future directions]. *IEEE Consumer Electronics Magazine*, 7(2), 18–21.

Rabah, K. (2017). Challenges & Opportunities for Blockchain Powered Healthcare Systems: A Review. *Mara Research Journal of Medicine & Health Sciences*, 1(1), 45–52.

Ramaswamy, V. & Penny, D. (2021). On the Performance of PBFT-based Permissioned Blockchain Networks in Constraint Environments. *ICC 2021-IEEE International Conference on Communications*, pp. 1–6.

Rauchs, M., Glidden, A., Gordon, B., Pieters, G. C., Recanatini, M., Rostand, F., Vagneur, K. & Zhang, B. Z. (2018). Distributed ledger technology systems: A conceptual framework. *Available at SSRN 3230013*.

Sajjad, M., Nasir, M., Muhammad, K., Khan, S., Jan, Z., Sangaiah, A. K., Elhoseny, M. & Baik, S. W. (2017). Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities. *Future Generation Computer Systems*.

Shahsavari, Y., Zhang, K. & Talhi, C. (2022a). Toward Quantifying Decentralization of Blockchain Networks With Relay Nodes. *Front. Blockchain 5: 812957. doi: 10.3389/fbloc*.

Shahsavari, Y., Zhang, K. & Talhi, C. (2019a). A theoretical model for fork analysis in the bitcoin network. *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 237–244.

Shahsavari, Y., Zhang, K. & Talhi, C. (2019b). Performance modeling and analysis of the bitcoin inventory protocol. *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pp. 79–88.

Shahsavari, Y., Zhang, K. & Talhi, C. (2020). A theoretical model for block propagation analysis in bitcoin network. *IEEE Transactions on Engineering Management*.

Shahsavari, Y., Zhang, K. & Talhi, C. (2022b). Performance Modeling and Analysis of Hotstuff for Blockchain Consensus. *IEEE International Conference on Blockchain Computing and Applications (BCCA)*.

Stathakopoulou, C., Decker, C. & Wattenhofer, R. (2015). A faster Bitcoin network. *Tech. rep., ETH, Zurich,. Semester Thesis*.

Stoykov, L., Zhang, K. & Jacobsen, H.-A. (2017). VIBES: fast blockchain simulations for large-scale peer-to-peer networks. *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, pp. 19–20.

Sudhan, A. & Nene, M. J. (2018). Peer Selection Techniques for Enhanced Transaction Propagation in Bitcoin Peer-to-Peer Network. *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 679–684.

Sukhov, A. M., Astrakhantseva, M., Pervitsky, A., Boldyrev, S. & Bukatov, A. (2016). Generating a function for network delay. *Journal of High Speed Networks*, 22(4), 321–333.

Sukhwani, H., Martínez, J. M., Chang, X., Trivedi, K. S. & Rindos, A. (2017). Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric). *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 253–255.

Swan, M. (2015). Blockchain thinking: The brain as a decentralized autonomous corporation [commentary]. *IEEE Technology and Society Magazine*, 34(4), 41–52.

Thakkar, P., Nathan, S. & Vishwanathan, B. (2018). Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. *arXiv preprint arXiv:1805.11390*.

Tschorsch, F. & Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3), 2084–2123.

Van Der Hofstad, R. (2016). *Random graphs and complex networks*. Cambridge university press.

Vasek, M., Thornton, M. & Moore, T. (2014). Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. *International Conference on Financial Cryptography and Data Security*, pp. 57–71.

Vukolić, M. (2015). The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. *International Workshop on Open Problems in Network Security (iNetSec)*, pp. 112–125.

Wang, W., Hoang, D. T., Hu, P., Xiong, Z., Niyato, D., Wang, P., Wen, Y. & Kim, D. I. (2019). A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. *IEEE Access*.

Watson, M., Begen, A. & Roca, V. (2011). RFC 6363. *See https://datatracker.ietf.org/doc/html/rfc6363*.

Watts, D. J. & Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, 393(6684), 440–442.

Wilkinson, S., Boshevski, T., Brandoff, J., Prestwich, J., andPatrick Gerbes, G. H., Hutchins, P., Pollard, C. & Buterin, V. (2016). Storj A Peer-to-Peer Cloud Storage Network.

Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. 151(2014), 1–32.

Wu, K., Peng, B., Xie, H. & Huang, Z. (2019). An Information Entropy Method to Quantify the Degrees of Decentralization for Blockchain Systems. *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*.

Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C. & Rimba, P. (2017a). A Taxonomy of Blockchain-Based Systems for Architecture Design. *2017 IEEE International Conference on Software Architecture (ICSA)*, pp. 243-252. doi: 10.1109/ICSA.2017.33.

Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C. & Rimba, P. (2017b). A taxonomy of blockchain-based systems for architecture design. *IEEE International Conference on Software Architecture (ICSA)*, pp. 243–252.

Yajnik, M., Moon, S., Kurose, J. & Towsley, D. (1999). Measurement and modelling of the temporal dependence in packet loss. *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now*, 1, 345–352.

Yang, F., Zhou, W., Wu, Q., Long, R., Xiong, N. N. & Zhou, M. (2019a). Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism. *IEEE Access*, 7, 118541–118555.

Yang, L., Bagaria, V., Wang, G., Alizadeh, M., Tse, D., Fanti, G. & Viswanath, P. (2019b). Prism: Scaling Bitcoin by 10,000 x. *arXiv preprint arXiv:1909.11261*.

Yang, Y. (2018). Linbft: Linear-communication byzantine fault tolerance for public blockchains. *arXiv preprint arXiv:1807.01829*.

Yao, Z., Wang, X., Leonard, D. & Loguinov, D. (2009). Node isolation model and age-based neighbor selection in unstructured P2P networks. *IEEE/ACM Transactions on Networking (TON)*, 17(1), 144–157.

Yasaweerasinghelage, R., Staples, M. & Weber, I. (2017). Predicting latency of blockchain-based systems using architectural modelling and simulation. *IEEE International Conference on Software Architecture (ICSA)*, pp. 253–256.

Yin, M., Malkhi, D., Reiter, M. K., Gueta, G. G. & Abraham, I. (2018). HotStuff: BFT consensus in the lens of blockchain. *arXiv preprint arXiv:1803.05069*.

Yu, H., Nikolic, I., Hou, R. & Saxena, P. (2018). OHIE: blockchain scaling made simple. *arXiv preprint arXiv:1811.12628*.

Zhang, J., Gao, J., Wu, Z., Yan, W., Wo, Q., Li, Q. & Chen, Z. (2019). Performance Analysis of the Libra Blockchain: An Experimental Study. *2019 2nd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 77–83.

Zhang, K. & Jacobsen, H.-A. (2018). Towards Dependable, Scalable, and Pervasive Distributed Ledgers with Blockchains. *IEEE International Conference on Distributed Computing Systems (ICDCS)*.