6-24-2020

A Private Bitcoin Payment Network with Reduced Transaction Fees and Confirmation Times

Enes Erdin
eerdi001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A PRIVATE BITCOIN PAYMENT NETWORK WITH REDUCED TRANSACTION FEES AND CONFIRMATION TIMES

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Enes Erdin

2020

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Enes Erdin, and entitled A Private Bitcoin Payment Network with Reduced Transaction Fees and Confirmation Times, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Alexander Perez-Pons

Hemang Subramanian

Selcuk Uluagac, Co-major Professor

Kemal Akkaya, Co-major Professor

Date of Defense: June 24, 2020

The dissertation of Enes Erdin is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development and
Dean of the University Graduate School

Florida International University, 2020

DEDICATION

Hediye, Kerem, ve Ahsen'e

ACKNOWLEDGMENTS

I want to thank the National Science Foundation for their generous support. It would be impossible for me to write this thesis without them. I wholeheartedly want to thank my supervisor Kemal Akkaya. He is not only one of the best researchers I have ever met but also a perfect mentor who tries to make me the best researcher and a good person. His insights and criticism were crucial during my graduate studies. I want to extend my appreciation to Selcuk Uluagac. His lively approach, research skills, and the way how he tackles hard problems were really enlightening for me. I can't express enough how much I learned from them both academically and socially. Thank you.

This dissertation would not have been possible without my collaborator and friend Mumin Cebe. His knowledge in different areas of science and engineering led me to learn new approaches and I feel I went further ahead in critical thinking after all of our discussions with him. Another collaborator Senay Solak is one of the brightest persons I have ever met in my life.

I feel fortunate to meet many people at FIU. The time in the ADWISE/CSL Lab would be really boring if I never met Burak, Abbas, Abdullah, and Ahmet. These smart people were always nice to me and I truly believe they will reach their goals in the near future.

Lastly, but not least, I would like to thank my wife Hediye for the patience she has shown in this journey of mine. My gratitude extends to my sister Esra, my fathers Halil and Esat, and my mothers Figen and Kadriye. Their support, help, and prayers were undeniably rewarding.

ABSTRACT OF THE DISSERTATION

A PRIVATE BITCOIN PAYMENT NETWORK WITH REDUCED TRANSACTION FEES AND CONFIRMATION TIMES

by

Enes Erdin

Florida International University, 2020

Miami, Florida

Professor Kemal Akkaya, Co-major Professor

Professor Selcuk Uluagac, Co-major Professor

Since its introduction, Bitcoin cryptocurrency has revolutionized the way payment systems can be designed in a purely distributed manner through its novel Blockchain data structure. While Bitcoin has opened new opportunities, it has been long criticized for its slow transaction confirmation times and high transaction fees. To address this issue, one of the recently emerging solutions is to build a payment channel network (PCN) on top of Bitcoin where the transactions can be made without writing to blockchain. Specifically, a PCN is a network where the users connect either directly or indirectly to send payments to each other in a trustless way. Being backed by the blockchain technology, PCNs satisfy a robust and flexible medium where the exchange of assets become frictionless and thus enable faster transactions with negligible fees. For example, Lightning Network, a second layer network built on top of the Bitcoin network, is being actively developed and it makes Bitcoin possible to be used for micro-payments. However, PCNs including LN bring new challenges on centralization, robustness and privacy as they accept more users. Such problems threaten the very idea of decentralization that comes with blockchain. Therefore, in this dissertation we target the problem of PCN topology formation that will come with ideal features and continue to grow without violating such characteristics. Specifically, we focused on the design of methods for obtaining peer-to-peer (P2P) decentralized PCN topologies. Inspiring from the multi-commodity flow problem, we first developed an optimal solution to establish the perfect PCN topology by utilizing mixed-integer programming. We solve this problem for the required capacities within the network for uninterrupted operation. Second, as mixed integer programming is proved to be NP-compete in complexity, we developed a heuristic optimization approach to take the solution into the polynomial-time domain. Third, to further enable scalability, we developed a new sub-optimal heuristic algorithm using the Dijkstra's shortest path algorithm. Finally, we turned our attention to privacy preservation problem for transactions and augmented each of the proposed approaches with privacy guarantees. Evaluation results indicate that our proposed approaches can enable desirable PCN topology features while respecting the privacy requirements.

TABLE OF CONTENTS

CHAPTER PAGE

1. INTRODUCTION 1
1.1 Challenges in Payment Channel Networks 4
1.2 Our Contributions 5
1.3 Outline of The Dissertation 7

2. LITERATURE REVIEW 9
2.1 Payment Channel Networks 9
2.2 Topological Observations on Lightning Network 17
2.3 Dijkstra's Shortest Path in Network Formation 18

3. BACKGROUND 20
3.1 Bitcoin and Blockchain 20
3.2 Smart Contracts 22
3.3 Off-chain Payment Channels 22
3.4 Payment Channel Networks 25
3.5 Lightning Network 27
3.6 Multi-Commodity Flow Problem 30

4. PAYMENT NETWORK FORMATION VIA MIXED INTEGER OPTIMIZATION 31
4.1 Problem Motivation and Definition 32
4.2 Proposed Optimization Model for PCN Creation 34
4.2.1 Overview 34
4.2.2 Formulation of the Model 34
4.2.3 Fair Distribution of Network Design Costs Among Nodes 37
4.3 Evaluation 38
4.3.1 Experimental Setup and Implementation 38
4.3.2 Metrics and Benchmarks 39
4.3.3 Experiment Results and Discussion 41
4.4 Conclusion 46

5. A PAYMENT NETWORK FORMATION HEURISTIC by LINEAR PROGRAMMING 49
5.1 Suitability of LN for Real-time Payments 49
5.2 Problem Motivation and Definition 50
5.3 Flow and Network Optimization Model 53
5.3.1 Formulation of the Network Optimization Model 54
5.3.2 Fair Distribution of Network Design Costs Among Nodes 57
5.4 Pruning-based Heuristic 58
5.4.1 Simplifying the MIOP Model 59

5.4.2 Algorithm for Building the Topology 59
5.5 Experimental Evaluation 62
5.5.1 Experiment Setup and Assumptions 62
5.5.2 Performance Metrics and Benchmarks 63
5.5.3 Experiment Results 65
5.6 Conclusion 76

6. A SCALABLE PAYMENT NETWORK FORMATION HEURISTIC USING DIJKSTRA'S SHORTEST PATH ALGORITHM 79
6.1 Proposed Heuristic Algorithm 80
6.1.1 Approach Overview 80
6.1.2 Finding Paths 81
6.1.3 Defining Edge Weights 83
6.2 Evaluation 87
6.2.1 Experimental Setup and Implementation 87
6.2.2 Metrics and Benchmarks 88
6.2.3 Experiment Results and Discussion 89
6.3 Conclusion 100

7. PRIVACY IN PAYMENT CHANNEL NETWORKS 102
7.1 Privacy in Lightning Network 106
7.2 Privacy Implications of MIOP Solution 107
7.2.1 Privacy Analysis of MIOP Solution 108
7.3 Privacy Implications of Linear Programming Heuristic 109
7.3.1 Incorporation of Privacy in the Heuristic 109
7.3.2 Privacy-Guaranteed Heuristic Experiments 110
7.4 Extending the Dijktra's Shortest Path Algorithm based Heuristic for Privacy Guarantees 114
7.4.1 Privacy Analysis of Dijkstra based Heuristic 116
7.4.2 Evaluation of Dijkstra based Heuristic with Privacy Considerations . . 119
7.5 A Comparative Analysis of the State of the Art PCN Implementations . 121
7.5.1 Network Architectures 121
7.5.2 Types of Blockchain Networks 124
7.5.3 State-of-the-art PCN Proposals 125

8. CONCLUSION AND FUTURE WORK 127

BIBLIOGRAPHY 130

VITA 136

LIST OF FIGURES

FIGURE PAGE

1.1 A simple representation of blockchain data structure. Each block starts with the hash value of the previous block. 2

3.1 Off-chain payment mechanism between two Blockchain nodes 23

3.2 A Simple Payment Channel Network. Alice can initiate a transfer to Bob utilizing channels between Alice-Charlie and Charlie-Bob. 26

3.3 An overview of the envisioned Payment Channel Network among retailers. 27

3.4 The money is forwarded via the intermediary nodes locked by a hash of a pre-image. The ownership is transferred only if the pre-image of the hash gets revealed. 28

3.5 How LN topology was looking in May 2018. Circles around some nodes highlight highly centralized ones. Image is taken from [Sto18] 29

4.1 An overview of the envisioned Payment Channel Network (PCN) for the EV charging example. 33

4.2 Betweenness centrality comparison between the networks with respect to γ and c^f 41

4.3 Capacity of the networks 42

4.4 Number of edges in networks 43

4.5 Number of edges in the network vs. γ for different c^f 44

4.6 Total Capacity of the network vs. γ for different c^f 45

4.7 Number of edges in the network vs. c^f for different γ 46

4.8 Total Capacity of the network vs. c^f for different γ 47

4.9 Time elapsed vs. c^f for different γ 48

5.1 An overview of the envisioned Payment Network among retailers. 53

5.2 A scene from the flow of the pruning heuristic. a) Initial table entries before pruning. b) Revised table entries and edges after pruning. . . 61

5.3 16 node torus topology 65

5.4 Heuristic approach compared to MIOP solution 67

5.5 Comparison of results related to heuristic for Full Connected network, Barabasi-Albert model, Torus Topologies vs. Random Topologies . . 70

5.6 Betweenness centrality and Cut Node Results of full connected, Torus, BA model and random topologies after applying heuristic 71

5.7 Comparison of Torus, BA model, Hypercube and Star topologies. Number of edges and standard deviation. 74

5.8 Comparison of Torus, BA model, Hypercube and Star topologies. Total network capacity and total time. 75

5.9 Further scalability tests for Torus and BA model generated topologies. Number of edges and standard deviation. 77

5.10 Further scalability tests for Torus and BA model generated topologies. Total network capacity and total time. 78

6.1 Initially fully connected mesh network topology. 86

6.2 Effect of W_i for a network of 20 nodes, $L_c = 3000$ 87

6.3 Optimal vs. Heuristic Comparisons - Betweenness Centrality. 90

6.4 Optimal vs. Heuristic Comparisons - Network capacity. 90

6.5 Optimal vs. Heuristic Comparisons - Total edges. 91

6.6 Optimal vs. Heuristic Comparisons - Utilization. 91

6.7 Optimal vs. Heuristic Comparisons - Standard Deviation. 92

6.8 Optimal vs. Heuristic Comparisons - Total Time (s). 92

6.9 Ideal Parameter Selection - Betweenness Centrality 93

6.10 Ideal Parameter Selection - Total Flow Capacity of the Network 93

6.11 Ideal Parameter Selection - Total Edges 94

6.12 Ideal Parameter Selection - Standard Deviation 94

6.13 Ideal Parameter Selection - Histogram of the number of hops for transactions (Normalized- percentage) 95

6.14 Ideal Parameter Selection - Total Time (s) 95

6.15 Effect of T_a - Betweennes Centrality 95

6.16 Effect of T_a - Network Capacity 96

6.17 Effect of T_a - Total number of Edges 96

6.18 Effect of T_a - Histogram 96

6.19 Effect of T_a - Standard Deviation 97

6.20 Effect of T_a - Total time (s) 97

6.21 Scalability Test Results - Betweenness Centrality 98

6.22 Scalability Test Results - Network Capacity 98

6.23 Scalability Test Results - Edges 98

6.24 Scalability Test Results - Histogram 99

6.25 Scalability Test Results - Standard Deviation 99

6.26 Scalability Test Results - Total Time (s) 99

6.27 Scalability Test Results - Number of cut nodes 100

7.1 For a typical example the sender u_s initiates a transaction to u_r through the intermediary nodes $u_{1\ldots N}$. A node can open as many channels as she wishes. The channels are assumed to be bidirectional. 102

7.2 Attackers can appear in the network in different places. ① The attacker is on the path of a payment. ② The attacker is not on the path of a particular payment but she can partially observe the changes in the network. ③ The attacker colludes with the other nodes. 104

7.3 Heuristic applied to topologies of BA and torus with and without privacy - Number of Edges and Standard Deviation 111

7.4 Heuristic applied to topologies of BA and torus with and without privacy - Total Network Capacity and Total Time 112

7.5 Privacy-preserving flow from X to Y. 113

7.6 Explanation of the rerouting mechanism 115

7.7 Relationship anonymity through multi-hop payments 118

7.8 Privacy aware method for 500 nodes - Betweenness Centrality 121

7.9 Privacy aware method for 500 nodes - Network Total Flow Capacity . . 121

7.10 Results for the Privacy aware method for 500 nodes - Total Edges 122

7.11 Privacy aware method for 500 nodes - Number of Hops 122

7.12 Privacy aware method for 500 nodes - Standard Deviation 123

7.13 Results for the Privacy aware method for 500 nodes - Total Time (s) . . 123

CHAPTER 1

INTRODUCTION

History of money undoubtedly went through many changes in years where technological developments played the most vital role in these changes. The world witnessed a new kind of representation for money when anonymous-and-infamous *"Satoshi Nakamoto"* posted his white paper on a cryptography mailing list stating that he was working on a new electronic cash system which is peer-to-peer (P2P) and works without any trusted third party [Sat09]. Following that, in 2009, with the introduction of Bitcoin software v0.1 *"Bitcoin cryptocurrency"* was born which also introduced the *blockchain* data structure. Since then many new cryptocurrencies were introduced but Bitcoin took over and it is the largest cryptocurrency among all others.

In Bitcoin, the money is owned, protected, and transferred by the use of Public Key Infrastructure (PKI). Anyone can generate a public-private key pair where the public key represents the address or identity of the user, which is called pseudonym. The user proves the ownership of her money by showing a provable signature related to that public address. So if a party owns some money, she can transfer it to a public address whenever she wants just by signing a transaction request. Being a public network, joining or leaving the Bitcoin network is completely free. When a user wants to make a transaction, she sends the transaction request to the active participants of the network which are called "miners". The miners first verify if there exists such money in such an address by inspecting the blockchain. They later verify if the transaction is signed by that address. After these, they put the transaction request in a "block". A block is composed of many such transaction requests from other users. As these miners are distributed around the globe a question arises: since each miner will prepare a different block from the others,

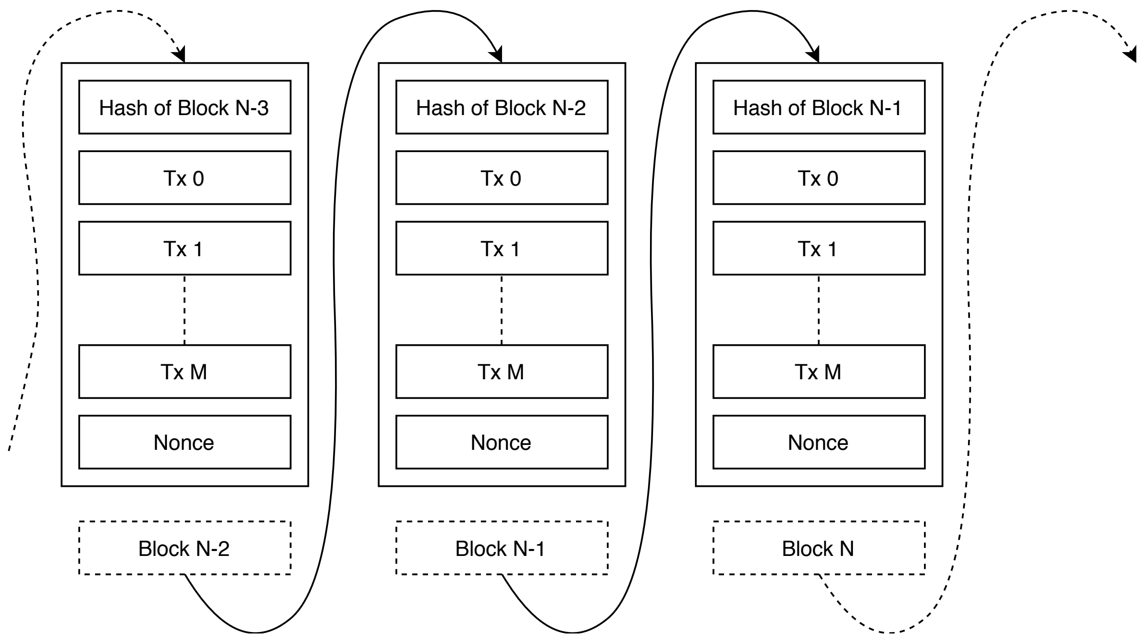


Figure 1.1: A simple representation of blockchain data structure. Each block starts with the hash value of the previous block.

which block will be accepted in the network? This is solved by utilizing a proof-of-work (PoW) scheme whereby constantly changing a nonce value in the block, every miner calculates a hash value for the block they prepared. If one of them can find a hash value smaller than a globally determined value, that block gets approved by the miners and they start to compete for finding the next block. The use of the word "compete" is suitable here because when a miner finds a valid block, she wins all of the transaction fees and other incentives in the block. An important fact about a block is that each block starts with the hash value of the previously accepted block. In that way, the blocks are connected to each other forming a chain of blocks, called the blockchain. The blockchain data structure is depicted briefly in Fig. 1.1.

Although Bitcoin introduced an unorthodox way of owning and transferring money, it has limitations. First, the size of a block has an upper limit. Second, the PoW scheme is time-consuming, and finding a block takes around ten minutes.

These two limitations bring scalability problem which hinders the successful adaptation of Bitcoin in daily payments. In order to solve this problem “payment channel” idea was introduced which eventually turned into the “Payment Channel Network” idea.

Payment channel networks [PD16, Fou16] are established by gathering nodes (users) that have cryptographically secured payment channels between each other. A payment channel gets established by the mutual agreement of the peers. To honor the future payment requirements, the parties put money as collateral in the channels. In a payment channel, when a party wants to make a payment to the other party, she simply gives the ownership of some of her money to the other side. In the case of a network of the payment channels, a node can transfer money to another node in the network as long as she can find a path of channels with enough funds to the destination node.

Lightning Network. One of the widely used PCNs today is the Lightning Network which has 13,027 nodes and 36,011 channels [1ML20]. The Lightning Network (LN) is a second layer network operating on top of the Bitcoin network. It is a cryptographically secure and practical solution to solve the scalability problem of the Bitcoin network by utilizing the off-chain concept. It is proposed for micropayments where the transaction fee will be comparable to the payment amount as opposed to the fees in the Bitcoin network. It is a fast and low-cost alternative to standard on-chain cryptocurrency-based payments. However, LN also has its problem that defeats the purpose of decentralization. We elaborate on these problems below. Throughout the thesis, cryptocurrency, money, and token will be used interchangeably.

1.1 Challenges in Payment Channel Networks

While PCNs bring many advantages and enable several applications, establishing a PCN poses several challenges as explained below:

The Security Challenge When a user wants to transfer money to another node, the security of the funds of the channels on the route should be satisfied. Such a design is challenging because, in case of an existence of a malicious node, the node will steal the funds in the channels or the channel fees gathered on the path or even a malicious user may not exist but an honest user can go offline and can not accept any payments. In LN, the security features are satisfied cryptographically by the use of Hash Time-Locked Contracts, a smart contract. Other cryptocurrencies similarly follow the use of smart contracts for the security of both payments and nodes. Blockchain is in the center of resolutions in the case of a dispute as a mediator. The peers submit their evidence to the blockchain on time and blockchain decides on the faulty/malicious party.

The Decentralization Challenge In general, most of the payment channel networks are not governed or moderated by 3rd party companies, meaning, they are public networks. Hence, attending and leaving the network is fully at the discretion of a user. Although that feature sounds good in terms of user freedom, that brings a burden on the network. Payment channel networks can be seen as social networks where some of the nodes will be highly attractive to other users (influencers). Hence, these attractive nodes will start to become central/dominant nodes in the network which is termed as *influential nodes* in social networking. Although we can assume these highly central nodes will not deviate from the protocol being utilized, they can learn too much about the dynamics of the network and about the users which then turns into possible threats to the participants. The first threat is learning about the spending habits of the users. The second threat is, by learning the user identities, censoring the users' activities in the network, e.g., denying the payment requests.

Privacy Challenge In cryptocurrency networks the privacy of the users is satisfied with simple pseudonyms but this is not the best solution as attacks on deanonymizing the users exist [BKP14]. Due to the nature of the payment channel networks, a party should disclose the amount of the payment being forwarded to the other party in a payment path because otherwise there is no way for others to know how much will be forwarded. Although moving to the second layer brings some more privacy enhancements to the Bitcoin network, the existence of this challenge stays there. For example, information about the identity of the payer, identity of the payee, amount of the payment, and investments done by nodes in the network can leak to other users not particularly involved in the payment path. When the businesses start to get involved in PCNs their *trade secrets* will start to become threatened [Sub17]. The aim of building a payment channel network is to ease the daily transactions of the customers. An ecosystem of retailers and shoppers are highly anticipated by PCN users and developers.

1.2 Our Contributions

In this dissertation, we concentrate on the following question:
Current payment channel networks provide limited dependability in terms of privacy and decentralization. The centralization of some of the nodes results in having powerful nodes in the network where they can learn more about the trade secrets of the users. Furthermore, they can observe and hence censor the users from connecting to each other. Is it possible to build a payment channel network from scratch with

enough capacity that satisfies higher privacy with avoiding monopoly and establishing a P2P network topology?

We briefly describe our contributions as follows:

Formation of a payment channel network by means of mixed-integer programming optimization Chapter 4 forms a payment channel network by mixed-integer programming optimization. The aim of that study is to answer the question of how much capacity should be assigned to the channels in the network in order to satisfy a durable operation. As an assumption for a scenario electric vehicles (i.e., payers or customers) get service from charging stations (i.e., payees or retailers) where payees come together and form a basis for the payment channel network. The customers connect to the PCN over the payees. In this chapter, we formalize the requirements, discuss the experimental setup, and give evaluation results.

Development of a heuristic which utilizes linear programming In Chapter 5 the mixed-integer optimization model offered in Chapter 4 is extended by turning it into a linear programming optimization model because mixed-integer programming does not scale well and computationally infeasible as it being NP-complete in terms of computational complexity. In this chapter, we analyzed the equations in order to take the mixed-integer optimization model into the linear programming domain. With careful investigation and assumptions, we proposed a sub-optimal heuristic that is much more scalable than the mixed-integer optimization model. We use a generic scenario where there are payers and payees. In this chapter, we discuss the experimental setup and evaluate the results of the proposed heuristic experiments.

Development of a scalable PCN formation heuristic which utilizes Dijkstra's shortest path algorithm In Chapter 6, in the light of the optimization results, we propose a heuristic for PCN formation by utilizing Dijkstra's shortest path algorithm. The previous chapters dealt with mixed-integer programming optimization and linear programming optimization models, and they were computationally infeasible for a very large network. So, in this chapter, we introduced a new heuristic after making analysis and borrowing ideas from our previous solutions. In a generic network environment, the retailers come together to form a consortium, and the customers declare their payment intent for a time period. The proposed idea also enforces the participants to make at least 3-hops in order to satisfy privacy. We evaluate the results of the proposal and discuss privacy implications.

A Privacy Study on Payment channel networks In Chapter 7 we added privacy preservation to each of the approaches proposed in the previous chapters. We show that by encouraging a private network formation we can protect the privacy of the users while preventing monopolies in the network. In this chapter, we also made a survey on what current state-of-the-art payment channel networks offer in terms of privacy. Currently, most of the studies concentrate on security, channel allocation, channel balancing, and usability features of the payment channel networks. In this sense, we observe that privacy is well understudied. In fact, some of the studies never mention privacy while others motivate the readers that their models' privacy implications will be studied in another work.

1.3 Outline of The Dissertation

Chapter 2 gives a summary of the literature related to this research problem. Chapter 3 gives background about the terminology used in the dissertation. In Chapter 4 we develop a mixed-integer optimization model for payment channel network formation and channel balance allocation in the network. Chapter 5 further extends the model developed in Chapter 4 in terms of scalability. Basically, the mixed-integer model is converted into a linear programming model by a heuristic approach. Further scalability improvement of these models is obtained in Chapter 6 with a heuristic algorithm utilizing the Dijkstra's shortest path algorithm. We present methods on how our proposals can be extended for privacy in Chapter 7. We evaluate the network metrics with these extensions and we also compare our privacy extended approach with the state of the art studies. Chapter 8 summarizes the dissertation and aims to direct readers to possible future research topics.

CHAPTER 2

LITERATURE REVIEW

2.1 Payment Channel Networks

There are several recent efforts in both industry and academic community to build payment channel networks (PCN). These efforts can be classified in two categories. The first category relies on building PCN for intra-blockchain operations. One example study in this category is Lightning Network [PD16]. It allows transferring Bitcoin between parties over already existing off-chain link without any confirmation delay and transaction fee. The similar idea is followed by Raiden to build PCN for Ethereum [Rai17]. The second category of works relies on building inter-blockchain operations to allow transfers between different cryptocurrencies without expensive on-chain confirmation. Examples include Inter-Ledger [TS15] and Atomic-CrossChain [Tea].

Existing PCNs are still in infant phase and therefore there are privacy challenges in routing of the payment. Blockchain community has started to identify this challenge and offered solutions to resolve privacy problem in PCNs [RMSKG17, MMSK+17, PZS+16]. However, all these studies design privacy-preserving routing solutions by assuming the availability of a perfect decentralized P2P PCN topology. None of these studies investigated the problem of network formation (i.e., formation of that P2P topology) and its effects on privacy. In addition, the community of PCN developers assume that there is an already established perfect PCN topology where payments are being forwarded between nodes.

Lightning Network (LN): LN [PD16] is the first widely deployed PCN. Nodes in LN utilize *"Hash Time Locked Contracts"* (HTLC) to transfer money from one user to the other. Publicly announced payment channels are discovered by the participants through gossip messaging. So every node knows most of the topology. The directional capacities in the payment channels are not advertised but the total capacity in the channel is known to the sender to calculate a path. The payment path is calculated by the sender. The sender encrypts the path by using the public keys of the intermediary nodes. By utilizing this mechanism, known as *"onion-routing"*, the intermediary nodes know only the addresses of the preceding and the following nodes. None of the intermediary nodes can decide on the origin or the destination of the message by looking at the payment package. LN uses Bitcoin as the underlying cryptocurrency. If any of the channel capacity on the path is not sufficient, the transaction gets cancelled. Three software were introduced based on this specification and these are `lnd` [Tec], `eclair` [ACI], and `c-lightning` [Ele]. If any peer tries to cheat the other one, as a de-incentive mechanism, all of the money in the multisig wallet gets forwarded to the victim peer by the blockchain. LN currently serves with 12860 nodes and 35534 channels. Total capacity of the network is 924 Bitcoin.

Raiden, [Rai17], offers a very similar framework to the LN. However, in Raiden the users can exchange not only Ethereum but also the tokens they create for themselves. In that sense, Ethereum offers a higher flexibility.

After the introduction of the payment channel network idea, the community concentrated on different aspects of it. We see works about security, privacy, and scalability issues in PCNs.

Spider Network: In spider network [SVA+18] authors propose applying packet-switching based routing idea to PCN which can be seen in traditional networks (e.g., UDP, TCP/IP). A payment is split into many micro-payments so that the probability of a successful transfer will be higher. It is also aimed that the network will have better-balanced channels. In this network there are nodes with special functionalities and they are called *spider routers*. Routers communicate with each other and they know the capacities of the channels in the network. The sender initially sends the payment to a router. When the packet arrives at a router it is queued up until the funds on candidate paths are satisfactory to initiate the transaction. The authors of spider network do not mention privacy. They only plan utilizing onion-routing as a future work. However, it is known that in packet-switching the source and the destination of the message should be embedded in the network packet. Thus, an intermediary node will know the sender the recipient. The micro-payments might follow separate paths, which would help keeping business volume private, if the recipients were kept private. Hijack of a spider router will let an attacker learn all the information within the network. An honest-but-curious node will know the source and the destination of the payments.

SilentWhispers: The payment network underlying in SilentWhispers [MMSKM17] is the *credit networks*, however, the idea can be well applied to the PCNs. SilentWhispers utilizes landmark routing approach, where landmarks are at the center of the payments. In their attacker model, they define 2 attackers. First the attacker is not on the payment path, i.e., a 3rd party node. In the second type they assume a landmark is honest-but-curious. Here landmarks know the topology but they do not explicitly know the available balance in the channels. When a sender wants to send money to a recipient, she communicates with the landmarks for her intent. Then landmarks start communicating with the possible nodes from "sender-to-landmark" to the "landmark-to-recipient" to form a payment path with max-flow algorithm. Each node in the path discloses the channel balance availability for the requested transfer amount to the landmarks. In this approach, the important point is that a landmark does not receive information from all of the nodes, but she receives information from a subset of the nodes. Then landmarks jointly decide on the feasibility of the transaction request by doing a multi-party computation. During the transfer phase when an intermediary node realizes the transaction to the next node, she informs the landmark. Landmarks acknowledges the transactions and when all of the transactions are executed on the intended path the transaction is marked successful. In this approach, the sender and the receiver are kept private (but landmark knows the sender-recipient pair). Payment amount is also private for the nodes who do not take part in transaction. Moreover, the balances of the channels within the network are not known by the other nodes. The approach is decentralized and landmarks are, by design, the trusted parties. As duty definitions for the nodes are ambiguous (who will be a landmark?) the approach can easily slide towards the centralized fashion.

SpeedyMurmurs [RMSKG17] SpeedyMurmurs is a routing protocol to be applied to PCNs especially designed for LN in mind. In this approach there are well-known landmarks as seen in SilentWhispers. The difference of this approach from SilentWhispers is that the nodes on the candidate path exchange their neighbors' information anonymously. So if a node is aware of a path closer to the recipient she forwards the payment in that direction, called "shortcut path" in the payment. In the shortcut path the intermediary node does not necessarily know the recipient but a neighbor close to the recipient. Another point in the study is that the payment routing is divided into small chunks and forwarded to the recipient. They hide the identities of the sender and the recipient by generating anonymous addresses for them. Intermediary nodes also hide the identities of their neighbors by anonymous addresses as well. Although it may be complex but applying de-anonymization attacks on the network will yield SilentWhispers. The algorithm is a decentralized approach and with unfair role distribution it may turn into a centralized approach.

PrivPay: PrivPay [MSKMP15] is a hardware-oriented version of SilentWhispers. The calculations taking place in the landmark are done in a tamper-proof trusted hardware. Hence, the security and privacy of the network is directly related with the soundness of the trusted hardware. The trusted hardware is in the center of operations and it is managed by a central node which yields high centrality. In this study, receiver privacy and payment amount privacy is achieved by spoofed payments.

Flash: Flash [WXJW19] is utilizing a modified max-flow algorithm to better handle constantly changing balances in the channels. Payments in smaller amounts are called mice and larger amounts are called elephant payments and they are treated in different ways. Small payments are sent randomly over pre-computed paths. However, for large payments, the nodes are probed to discover channels with available funds. That brings is to the fact that in order to calculate the available capacity over a path, directional capacities on the path must be known and disseminated to the rest of the network. That reveals the amount deposited into the channel by each participant of the channel. It is also possible to find out the volume of the payment passed through a channel by constantly observing changing capacities by probing.

Flare: Flare [PZS+16] is a LN routing protocol. There are beacons, like landmarks as in landmark routing. However, the difference is that the beacons are chosen by the users. Meaning that, a sender cleverly selects a node and informs it as it is chosen as a beacon. The routing tables are exchanged between sender, recipient, and beacons which is very similar to node discovery algorithm in LN.

Rayo and Fulgor: Rayo and Fulgor [MMSK+17] are two multi-hop routing protocols for PCNs (Fulgor is suitable for LN only). They develop these protocols against the security flaw coming from hash distribution in LN. In LN the same hash of the pre-image is distributed on the path when a payment takes place so the authors argue that this creates a problem for the privacy and relational anonymity between the sender and the recipient. To solve that problem they introduce multi-hop HTLC contracts. In their non-blocking approach, Rayo a non-blocking payment routing, there is a global payment identifier system which helps the nodes to order the payments with respect to their identifier number. For that reason, Rayo is prone to relationship anonymity attacks if the attacker is located in the payment path.

Bolt: Bolt [GM17] is hub-based payment system where it supports only payment cases if there is only one intermediary node between sender and recipient. Bolt utilizes zere-knowledge proof based cryptocurrencies, so its wide application seems not possible for now. However, it satisfier very strong relationship anonymity if the intermediary node is honest. On the other hand, depending on a single node tends to make the PCN offered by this approach a centralized one.

Anonymous Multi-Hop Locks (AMHL): In AMHL proposal [MMSS$^+$19] the authors offer a new HTLC mechanism. What they try to solve is that in PCNs, on a payment path, the sender agrees to pay some fee to each of the intermediaries for their service. However, if two of these intermediaries maliciously collude they can eliminate honest users in the path and consequently steal their fees. In order to solve this they introduce a pre-sending communication phase in which the sender distributes a one-time key to the intermediary nodes. Although the HTLC mechanism is improved for the security of the users the sender privacy is not protected; each of the intermediaries learns the sender. However, relationship anonymity can be satisfied.

In Sprites, [MBB$^+$19], the authors offer a new channel establishment method. With this method, in a multi-hop payment not all of the channels on the path are locked but only the worst-case collateral cost channel is locked. With their framework they lower the waiting time of a payment in the path so the opportunistic cost of locking some money for a payment for some time will be decreased. They offer opening channels based on the established channels.

Other than the PCNs there are also Payment Hubs in which the sender and the recipient is separated by a single hub. In TumbleBit [HAB$^+$17], the unidirectionally operating hub generates tokens and these tokens are sold or bought. The aim in this is keeping the sender and recipient completely anonymous. Here a tumbler acts as the point of trust for anonymity. The recipient knows the payment is coming from the Tumbler but actual sender is kept secret.

On the other hand, Bolt (Blind Off-chain Lightweight Transaction) [GM17] offers unlinkable channels in order to keep the peers' identity anonymous. Bolt utilizes either a cryptocurrency or a mixer as the building block to satisfy anonymity. In their proposal there is a well-known node, for example the retailer. The retailer for sure receives a payment from one of the customers but it can not differentiate the users from each other, hence the anonymity of the user gets satisfied. The idea was proposed in 2017 and by 2020 the development is in its pre-beta phases. Bolt currently uses Z-cash network as the first layer network. Zero-knowledge-proof based Z-cash is cryptographically more successful compared to other main-stream cryptoccurrencies. The proposal is not applicable for cryptocurrencies other than zero knowledge proof ones.

There have been no studies about how to design a payment network an overlay transferring service of payments where the sole purpose of the network is forwarding payments, there is no actual trade between the members of PCN. Moreover, we investigate how to build a PCN overlay to come up without any associated *forwarding fee* by sharing network formation cost between parties fairly. In this work we concentrate on how the PCN topology should be established such that the channels' depletion can be prevented and also by enforcing at least 3-hops in a multi-hop payment they investigate how initial channel balances change while the sender/receiver privacy and the relationship anonymity can be satisfied more strongly. In PCNs, the secure communication protocols are undoubtedly important, however, if the network topology is not ideal, e.g. star topology, some of the nodes will be in an advantageous position to learn about the users and corresponding payments in the network. We assume a real use case where shoppers pay to the merchants. We propose that the merchants create a fully distributed P2P topology and the customers connect to this network through merchants. Hence the merchants will undertake the financial load of the network while earning money and by utilizing the mechanisms offered by PCNs the privacy of the users will be satisfied.

Works targeting the privacy of the LN is not common. Right now, due to being in the early development stage, LN enthusiasts have been discussing the privacy brought by LN on public forums or in communication channels. An attack to the undisclosed directional balance is introduced in [HJNAP$^+$19]. In this study, the authors send transactions with a never existing signature and observe the reaction of the nodes on the path. They increase the payment amount until they get an unsuccessful transfer information. Authors in [RMT19] study on three different snapshots of LN and calculate the robustness of the topology with respect to different known attacks and node failures in terms of privacy and transaction success rate.

However, these studies and studies on novel routing protocols in payment channel networks are orthogonal to our study. Although the researchers in the academia propose novel methods or protocols to provide more privacy for PCNs, the practical world seems not to be driven by these proposals. In [GMSR$^+$19] and [JKLT19], the authors come with really inclusive and comprehensive surveys on off-the-chain networks but they never mention about network formation. In this chapter, we concentrate on network formation in LN with improved privacy features.

2.2 Topological Observations on Lightning Network

In [Mar19], the author discusses the topological change in the LN within the first year after its kick-off. In his work the author assumes LN as a weighted graph. However, with such an assumption the node strengths should be determined. The strength of a node is the sum of the weights of the node's edges. The author defines a new strength which takes the "capability of sending a large transaction". For the robustness, anonymity, and synchronizability analysis, he removes some of the highly connected nodes from the network. The removing procedure is removing 1, 2, 5, 10, 25, and 50 nodes, and then he observes the changes in the metrics of the network. This removal is a valid test scenario because in case of an attack to the network, the attacker will aim the most connected nodes in order to disrupt the operations taking place. In addition to looking at the betweenness centrality change in the network after attacking the nodes, the author also looks at the change in the success of the network by randomly removing the nodes. The author also discusses how the payment forwarding success rate increase in the months where LN looks more central. So, the centralization of LN seems inevitable for success.

In [SGNB20], similar to the previous work, authors observe the topological properties of LN in the snapshot taken on 3 January 2019. The authors show the robustness of LN against random and targeted attacks. The find that there is a negative degree assortavity in the network. This implies that the nodes with lower degrees connect mostly to the nodes with higher degree. This phenomenon easily explains how LN can move towards a hub and spoke (central) network. Hub and spoke structure seems to have advantages. For example, randomly attacking the network will remove child nodes, which has almost no major impact on the operation of the network. However, in case of a targeted attack, where the hub node is targeted, there will be a major operational malfunction in the network. They find that average shortest path in the network is 2.806. It is also interesting to observe that highly connected nodes are run by some very few companies. This is another indication that LN is going towards a centralized path.

In [TMSM20] the authors analyze the current LN topology with respect to the wormhole attack. A wormhole attack is an attack where 2 of the intermediary nodes collude in order to steal the forwarding fees in a multi-hop payment. They show that if an adversary can control 2% of the LN network she can learn sensitive data in the network in order to carry out a wormhole attack. They also investigate the concurrency in the current LN in which half of the nodes are busy with forwarding very small amounts. This hinders the scalability of the LN in total. They claim that forwarding \$1.5M in the network completely blocks all operations. Moreover, if central nodes are taken down this cost decreases to \$225K.

2.3 Dijkstra's Shortest Path in Network Formation

Dijkstra's shortest path algorithm finds the minimum length paths between a source node to every other node in a connected graph. To do so, the edges in the network should have non-negative lengths which are also called weights. Yielding the optimal path in a route discovery problem, Dijkstra's shortest path algorithm has found a wide area of applications. However, due to dynamic behavior of some of the systems, it is not unusual to come across modified versions of Dijkstra's shortest path algorithm. Here we list some examples: To start with, *lnd*, an implementation flavor of the LN uses a modified version of Dijkstra's shortest path algorithm where the weight of a link (or channel) is a partial combination of the fee asked by the channel and the HTLC time-lock duration. In [TPA12] the authors propose a new cost calculation heuristic for the communication links in the body area networks. The aim for dynamically adjusting these weights is increasing the battery life of the devices. The weight calculations are centrally done at the access-point. In [BAN17] the authors modify the Dijkstra's shortest path algorithm by introducing a penalty cost proportionally increasing with the increasing number of hops in the path. The aim here is decreasing the passengers' walking distance while also keeping the number of hops a passenger will go through. The weight calculations are performed at the public transport central management office. In all of these studies the nodes and edges are physically fixed. Similar to these works, we also propose assigning and updating cost functions on the edges. However, different from these works we not only want to pick the most efficient path between a sender and a receiver but also want to decide if a path should be created in between or not. Our algorithm runs before a network is formed and the output of it is a guide for the participants to establish a network.

CHAPTER 3

BACKGROUND

This chapter provides the preliminaries and background.

3.1 Bitcoin and Blockchain

There are numerous cryptocurrencies on the market such as Bitcoin, Ethereum, Monero, Zcash that utilize blockchain technology. Most of these digital currencies provide anonymity based on pseudonym addresses. Monero and Zcash, for instance, provide perfect anonymity by employing mixing and zero-knowledge proofs, respectively. However, they are not widely adopted. Currently, Bitcoin is the most widely used digital currency and its market cap is above 60% among all digital currencies.

Blockchain technology, with the introduction of Bitcoin, brought a new distributed database to light which is a public, transparent, persistent, and append-only ledger distributed among the participants. With various cryptographically verifiable methods, called Proof-of-X (PoX), each participant in the network holds the power of moderation of the blockchain. For example, Bitcoin utilizes proof-of-work (PoW) mechanism. PoW requires heavy cryptographic hash calculations that cause scalability problems the results of which will be discussed. Therefore, the cohort of independent participants turns the blockchain into a liberated data/asset management technology free of trusted 3rd party moderators. Although it can be used in many different areas, the most commonly used application of blockchain technology is cryptocurrencies, as it will interchangeably be called money, asset, or token in this dissertation. A cryptocurrency is, supposedly, a cryptographically secure and verifiable currency holder of which use it to purchase goods and services.

Blockchain technology undoubtedly changed how data can be interchanged, stored, and represented. Nonetheless, making a consensus on the final state of a distributed ledger comes with drawbacks. The first drawback is long confirmation times for transactions. For example, in Bitcoin, a block that holds the transaction information is generated at about every 10 minutes. As block generation and verification are done on a very large geographical scale a portion of the network can continue with a different chain due to possible interruptions in communication. In Bitcoin, the longest chain survives. Hence, seeing a transaction in the next block does not necessarily mean that this block will survive because the longest chain temporarily can not be known. If payment is finalized by directly looking at the latest block *double spending* may occur. To prevent double spending it is advised for the Bitcoin users to wait at least 6 blocks which yield around 60 minutes waiting time for the finality of a transaction. That is a drawback which hinders prompt payment capability. Note that, as a block has an upper bound in size, the total waiting time for the users will eventually be longer during the congested times of the network. Nevertheless, if a user is in a hurry for approval of her transaction, she will find herself paying larger fees than what her competitors are paying to the miners. This brings us the second drawback of using Bitcoin for payments. Normal users want to make transactions, the miner nodes, which generate and approve blocks, get fees from the users. The fee amount is not proportional to the transaction amount. The miners will always choose the transactions which offer larger fees for the highest profit and leave the others in the transaction request pool (*mempool*). When the network is congested some senders will be inclined to pay more fees in order to be selected among other competitors. During once of such an extreme case, in the 2017 Bitcoin boom, the payees either had to pay more than $20 for transaction fees or they had to wait 1188 minutes on average [Bro17]. Therefore, such transaction confirmation times are not suitable for applications where timely payment evidence is critical.

3.2 Smart Contracts

Being able to employ smart contracts is another feature that makes blockchain an unorthodox asset management technology. Smart contracts are scripts or bytecodes (in case of Ethereum) which define how transactions will take place based on the event possibilities defined within the contract. As always, the duty of final decision-making is on the blockchain. Hence, the blockchain finalizes the transaction outputs if the smart contracts are utilized. While preparing a contract, the parties should be careful such that when the contract is awarded the transactions will not be reversible in the blockchain.

3.3 Off-chain Payment Channels

Off-chain payment channels mechanism [P$^+$15, Wik] is proposed for saving from transaction fees and time in the current Bitcoin network which constitutes the main motivation of this study. Specifically, an in-advance payment transaction information is provided to the Blockchain for establishing a 2-of-2 multi-signature trustless escrow account, and future successive transactions take place using this shared account. The account activities are to be signed and tracked by the peers without being written to Bitcoin's public ledger. The amount put in the multi-signature account is decided individually by the peers and unless that amount is reached, the transactions can continue. In this scheme, the peers need to pay fees for at least two on-chain transactions: one to open the channel (depositing to the 2-of-2 multi-sig wallet) and one to close (withdrawing from the 2-of-2 multi-sig wallet) it.

The example shown in Fig. 3.1 depicts this concept. Alice opens an off-chain payment channel by instantiating a 2-of-2 signature escrow account with Bob. They both sign the new account separately. Alice then deposits 5 Bitcoins to the 2-of-2

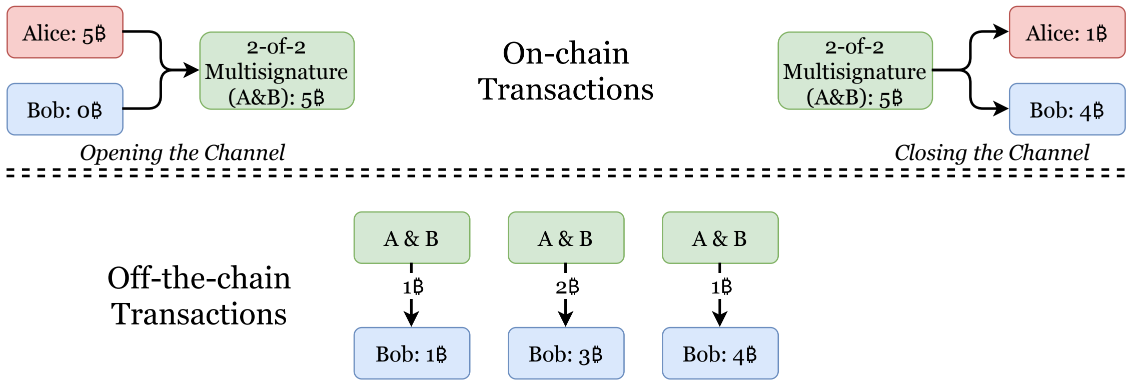


Figure 3.1: Off-chain payment mechanism between two Blockchain nodes

multi-sig (escrow) wallet by performing an on-chain transaction which determines a directional channel capacity, from Alice to Bob, as 5 Bitcoins, it is called the "Opening Transaction". From now on, Alice can make payments to Bob simply by partially giving the ownership of her Bitcoins to Bob until the directional capacity of the channel is exhausted (Alice has no ownership) via establishing "Commitment Transactions". There is no limit for the total number of transactions, and these transactions are not written to the main public ledger. In the figure, we see only 3 off-chain transactions at different times: 1, 2, and 1 Bitcoins. Eventually, when the channel is closed, only the remaining Bitcoins (1 Bitcoin) and the total transferred Bitcoins (4 Bitcoins) are committed respectively to Alice and Bob and written to the public ledger. With this setup, there is no way to accomplish a total transaction volume of more than 5 Bitcoins between Alice and Bob. The off-chain payment channel provides guarantees to Alice and Bob to refund the balance in the escrow account at any time or at a mutually agreed channel expiration time. This guarantee is satisfied by a smart contract called "*Hash Time Locked Contracts (HTLC)*" [Bitb]. The smart contract, mediated by the Blockchain, includes the participants' addresses, their share in the multi-sig wallet, and under what conditions the contract will be honored. When the transaction with the contract is initially published in the blockchain, the channel gets established. With every new transfer, an HTLC is to be created by the peers, a peer gives ownership of some of her money to the other. The proof should be satisfied in a limited period of time. Keeping the state of the channel up-to-date is the duty of the channel participants. So the final state of the channel gets updated in the local database of the parties. When the parties think the channel needs to be closed, e.g. the channel capacity is depleted on one side, or the business is over, or even one party tries to cheat, the parties submit the contract to the blockchain for it to honor the final state of the channel. By the closing transaction, each side receives her own final share in the multi-sig wallet. In the case of a problem or a fraudulent activity, the peers are responsible to open a dispute and the blockchain makes the final decision based on the submitted evidence related to the final state of the channel and the created HTLCs. When a peer (let us assume this is the malicious peer and she wants to close the channel with a state in the past and the other peer is the honest peer) wants to close the channel she publishes the transaction in the blockchain. But the HTLC states that the initial publisher (malicious peer) should wait for a longer time for the other peer to respond (Time-lock feature). When the honest peer discovers this channel closing transaction on the blockchain she makes another transaction with the final state of the channel. The contract states that this objection transaction has a shorter wait time. So the funds in the multi-sig wallet get transferred to the peers. The opposite case where the honest wants to close the channel and the malicious one objects to the final funds is not possible because when the honest peer initiates the channel closing with the final state of the channel, the malicious peer will not be able to open a dispute with a previous state because of the timestamp in the signed contract. So at the end of the wait time, both parties receive their share.

3.4 Payment Channel Networks

Due to the scalability issues mentioned in Section 3.1, researchers, developers, and entrepreneurs are always in the search of solutions to make the cryptocurrency networks scalable. This pursuit has reasonable grounds such that, with its current architecture, the Bitcoin network can handle 7 transactions per second (tps) at maximum and for Ethereum, the second-largest cryptocurrency, maximum achievable transaction rate is 15 tps. Among many offered solutions, the off-chain payment channel proposal attracted enough audience resulting in commercially viable practical implementations of this idea with its derivatives.

When more of these channels come together, by each node having channels with other nodes, a payer can make transactions in a multi-hop fashion. Eventually, that forms a network of payment channels, called the payment channel network (PCN). This is shown in Fig. 3.2 simply, where Alice-Charlie (A-C) and Charlie-Bob (C-B) have channels. Let, A-C and C-B initialize the channels when `time` is `t`. Although Alice does not have a direct channel with Bob, she still can transfer money to Bob by finding a path of channels from her to Bob. When `time` is `t+x1`, Alice initiates a transfer of 10 tokens to Bob. The money is destined to Bob over Charlie. If Charlie honors this transaction Alice gives 10 tokens of her share to Charlie in A-C channel. Likewise, Charlie honors this transaction in the C-B channel. When the transaction is over, A-C and C-B channels get updated. Alice wants to make another transaction of (20 tokens) to Bob when `time` is `t+x2`, and the shares in the channels get updated once again.

LN exploits the off-chain concept to create multi-hop payment paths between participants. To enable this idea in practice, users are supposed to route their payments to any destination through a series of payment channels in a network of

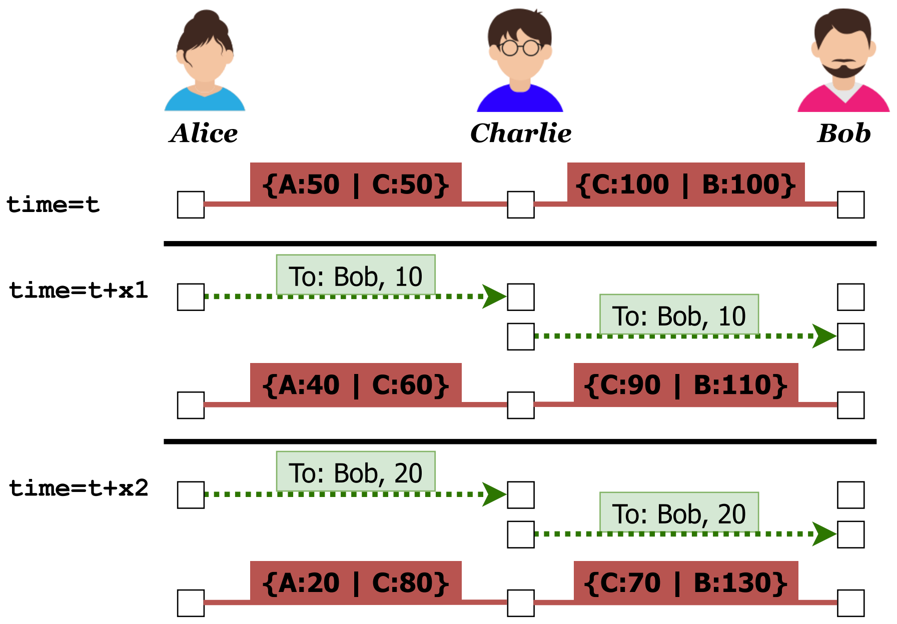


Figure 3.2: A Simple Payment Channel Network. Alice can initiate a transfer to Bob utilizing channels between Alice-Charlie and Charlie-Bob.

nodes. If such a channel/link series exist among the nodes, then a user can utilize one or more of these links (i.e., multi-hop links) to reach another node for making a payment. A sample payment network is shown in Fig. 3.3. Although this is a very simple sketch of the idea, in actual PCNs the intermediary nodes ask for fees for their forwarding service. For example in LN, there are two types of fees. The first one is the *base fee*, which is constant for every transaction, and the second fee is the *proportional fee*. This fee is proportional to the amount being forwarded. So if Alice wants to use 3 intermediary nodes in the payment she has to include the shares of the intermediaries in the payment. This may sound oxymoron as the idea was getting rid of the fees in the Bitcoin network but now we face another kind of fee right now. However, in the current LN, these fees are fractions of US cents, they are extremely low compared to on-chain transaction fees.

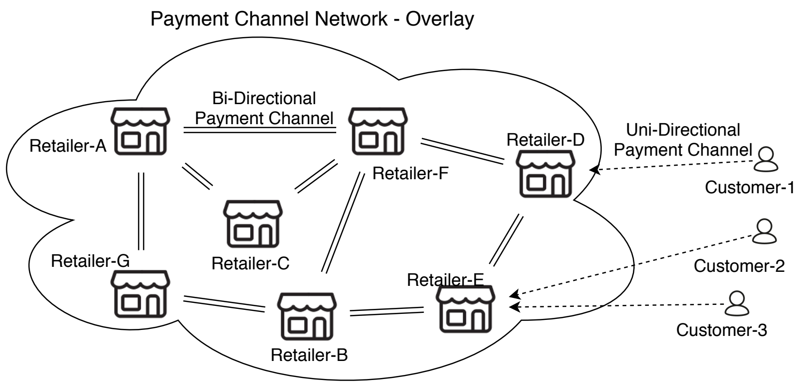


Figure 3.3: An overview of the envisioned Payment Channel Network among retailers.

3.5 Lightning Network

Among the current PCNs, LN is the most widely adopted solution since the introduction of the off-chain payment channel by the Bitcoin community [Bita]. LN exploits the off-chain mechanism to create multi-hop payment paths between participants. These paths are used for accomplishing the Bitcoin transfer between arbitrary parties. To enable this idea in practice, users are supposed to route their payments to any destination through a series of payment channels. If such a channel/link series exist among the nodes, then a user can utilize one or more of these links (i.e., multi-hop links) to reach another node for making a payment. Fig. 3.4 explains how multi-hop payment is done in LN cryptographically. Alice can make a transfer to Bob via Charlie using the off-chain links which are already established. Assuming already established and funded channels, Alice-to-Charlie and Charlie-to-Bob, the payment between Alice and Bob is accomplished in the following way: Bob sends a hash of a secret (i.e., a disposable key) to Alice. First, Alice signs a commitment transaction destined to Bob and forwards it to Charlie. When Alice is forwarding the money to Charlie, she locks the payment with the secret she received from Bob, this secret is the hash of a pre-image. Hence, unless Charlie has the pre-image in

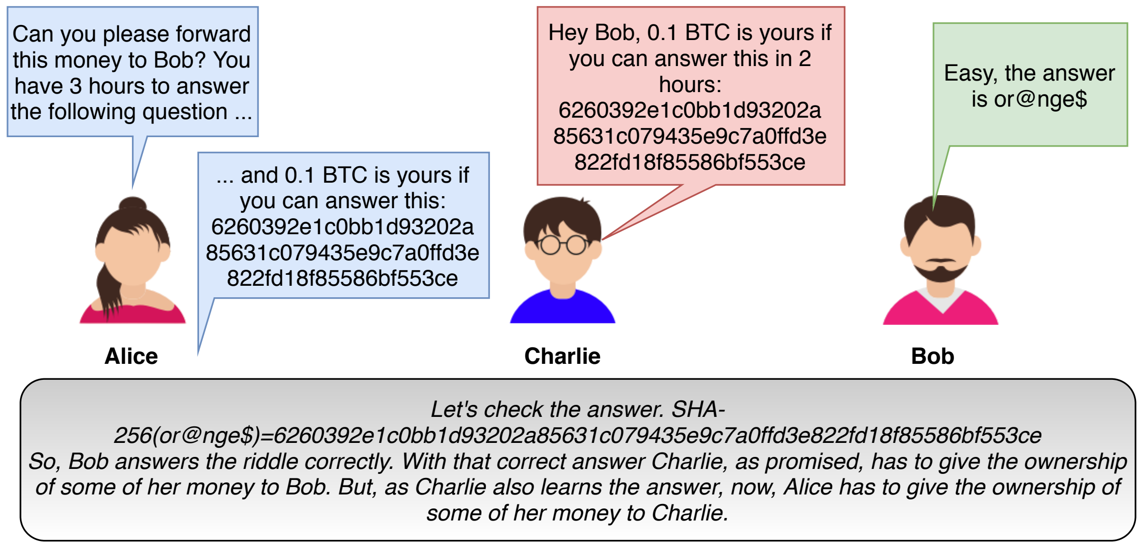
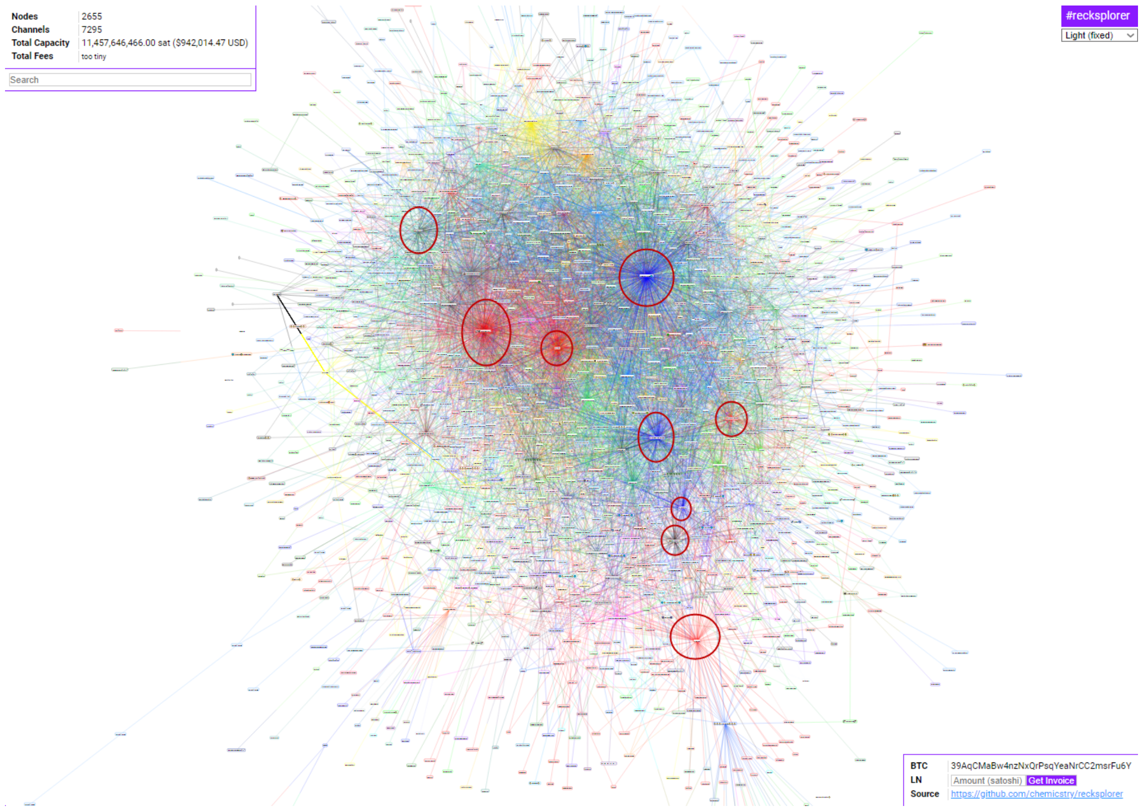

Can you please forward this money to Bob? You have 3 hours to answer the following question ...
... and 0.1 BTC is yours if you can answer this: 6260392e1c0bb1d93202a85631c079435e9c7a0ffd3e822fd18f85586bf553ce
Hey Bob, 0.1 BTC is yours if you can answer this in 2 hours: 6260392e1c0bb1d93202a85631c079435e9c7a0ffd3e822fd18f85586bf553ce
Easy, the answer is or@nge$
Alice
Charlie
Bob
Let's check the answer. SHA-256(or@nge$)=6260392e1c0bb1d93202a85631c079435e9c7a0ffd3e822fd18f85586bf553ce
So, Bob answers the riddle correctly. With that correct answer Charlie, as promised, has to give the ownership of some of her money to Bob. But, as Charlie also learns the answer, now, Alice has to give the ownership of some of her money to Charlie.

Figure 3.4: The money is forwarded via the intermediary nodes locked by a hash of a pre-image. The ownership is transferred only if the pre-image of the hash gets revealed.

hand, he will not be able to unlock the payment. In the second hop, Charlie commits a new transaction locked by the same secret (hash) and forwards it to Bob. To receive the payment, Bob has to reveal the pre-image which he generated at the very first stage of the payment. When Bob reveals the pre-image, he unlocks the contract and gets his payment from Charlie. As Bob revealed the secret, now, Charlie knows it and by revealing the secret to Alice, he unlocks his share and gets his money from Alice. After every transaction, the peers update the state of the channel with new secrets in a trustless manner.

The senders also attach a deadline to the Hash-Lock that is agreed while establishing the channel which is the "Time Lock" part of the HTLC. With the Time-Lock feature, the sender can enforce the receiver to reveal the secret within a certain amount of time. Otherwise, the transaction fails and the sender can claim his/her money back. The process is also depicted in Fig. 3.4. The state of the channel between the peers is kept current by mutual signatures. The final claim of the



Nodes	2655
Channels	7295
Total Capacity	11,457,646,466.00 sat ($942,014.47 USD)
Total Fees	too tiny

BTC	39AqCMaBw4nzNxQrPsqYeaNrCC2msrFu6Y
LN	Amount (satoshi) Get Invoice
Source	https://github.com/chemicstry/recksplorer

Figure 3.5: How LN topology was looking in May 2018. Circles around some nodes highlight highly centralized ones. Image is taken from [Sto18]

remaining funds is done on the on-chain Bitcoin network. In HTLC mechanism, whenever a peer claims funds with fabricated channel usage or claims funds with a more previous channel state there will be enough time for the other peer to open a dispute on the blockchain. If the complainant brings evidence forward to prove her case all of the channel capacity will be funded to the complainant as a punishment. This mechanism is offered to prevent double-spending or forging of the channels as a disincentive. The discovery of intermediary nodes between Alice and Bob is accomplished via running *node and channel discovery mechanism* within LN. This basically allows nodes to obtain a local view of the network's topology so that Alice can discover a route to Bob.

LN is being actively developed and numbers of nodes and channels in the network are increasing steadily. Fig. 3.5 shows a snapshot of the network taken on May 6th, 2018.

3.6 Multi-Commodity Flow Problem

The multi-commodity flow problem is a network flow problem where multiple commodities are carried from sources to destinations. It can be explained by a simple example; let a retailer chain has to deliver 3 different commodities to 3 different retail shops (flow demand) using only a single truck. Now the retailer has to make a decision: given the capacity of the truck and the requests of 3 different retailer shops, how much of these items should be placed in the truck and which retail shops are to be visited for the highest profitability of the company? Producing an integer-based solution satisfying the requirements is shown to be NP-complete [EIS75].

CHAPTER 4

PAYMENT NETWORK FORMATION VIA MIXED INTEGER OPTIMIZATION

This chapter investigates incorporating the off-chain model into a payment network by building off-chain channels among stores (e.g., car charging stations). Consequently, our objective is to build an overlay distributed payment network where customers (e.g., EV owners) can make their payments through pre-established off-chain channels without any on-chain transaction cost and significant transaction confirmation times.

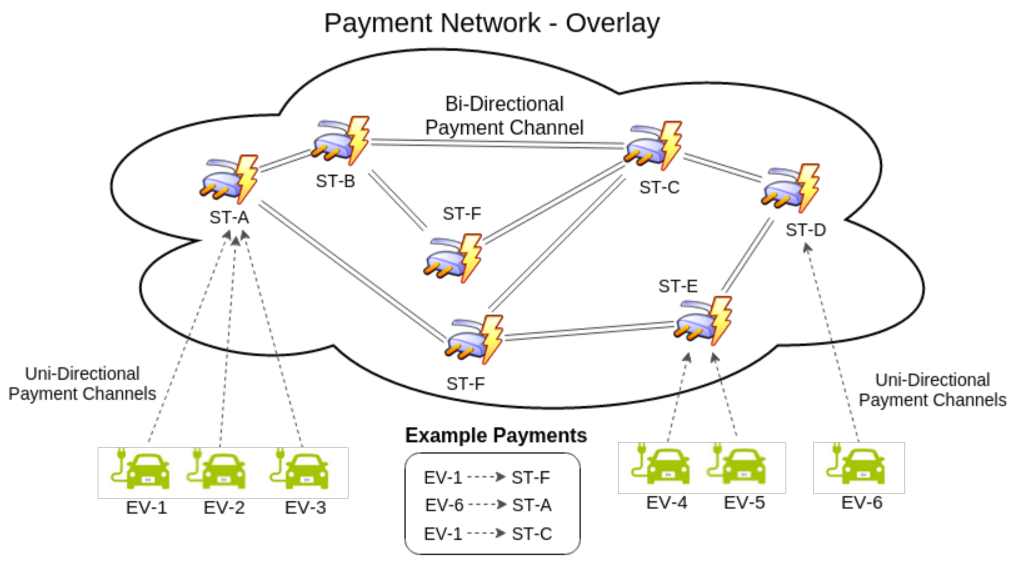
Nevertheless, there are several challenges in building such an overlay network. First of all, the topology of the resultant network is very crucial. The topology should not be like a hub-and-spoke model as in the current Internet backbone, since this will create a certain number of dominating nodes where most of the payments will pass through. This is not only detrimental to the privacy of the payments but can also create a monopoly where certain nodes may eventually would like to charge additional transfer fees. Similarly, hub and spoke or star like models as in the case of current credit card payment models are not desirable either for the same reasons. Instead, the topology should be purely peer-to-peer (P2P) and strive to distribute the off-chain channels to many pairs in order to reduce total transaction fees and increase the privacy of customers. Specifically, we formulate this network design problem as a multi-commodity flow problem where establishment of payment channels between stores are optimized according to different cost-sharing scenarios among parties, while also ensuring the correct routing of the payments.

4.1 Problem Motivation and Definition

The main motivation of our problem is to minimize the transaction fees in Bitcoin payments, and eliminate the long waiting times for transaction verification while optimizing the channel capacities with possible minimum allocation. We propose to utilize the off-chain idea to create payment channels among the suppliers assuming that every node in this network will create in-advance payment channels (i.e., links with certain transaction capacities) with some other nodes. Fig. 4.1 shows an overview of the proposed Payment Channel Network (PCN) with all components. If there exists a channel/link among every store, then a customer can utilize one or more of these links (i.e., multi-hop links) to reach another supplier for making a payment without paying any transaction fee while also maintaining its privacy. For instance, in Fig. 4.1, the EV-1 can get a service or good in ST-C but the payment can be made via ST-A, ST-B (e.g., EV1, ST-A, ST-B, ST-C) in 3-hops using the payment channels already established.

The accomplishment of a payment between EV-1 and ST-C can be considered as Internet packet forwarding, and depends on the availability of a connecting path between pairs and sufficient capacity on each channel along the path. The intermediary nodes (e.g., ST-A and ST-B) work as a router and are not directly involved in the payment between the payer and the payee other than forwarding. Therefore, they can not steal any Bitcoin while forwarding a payment. Note that a customer will only need one unidirectional channel (e.g., sending transaction only) while suppliers may need multiple of these links which are bidirectional (e.g., can send and receive transactions).

Based on these discussions, our problem can be formally defined as follows: "*Let us assume M service providers, and N customers in a region ,e.g. South Florida,*



Payment Network - Overlay
Bi-Directional Payment Channel
ST-A
ST-B
ST-C
ST-D
ST-E
ST-F
ST-F
Uni-Directional Payment Channels
Uni-Directional Payment Channels
EV-1
EV-2
EV-3
EV-4
EV-5
EV-6
Example Payments
EV-1 → ST-F
EV-6 → ST-A
EV-1 → ST-C

Figure 4.1: An overview of the envisioned Payment Channel Network (PCN) for the EV charging example.

and a set of service providers available to each customer. Let us also assume a PCN can be represented as a graph $G = (V, E)$, where V represents nodes and E represents payment channels. In our case, the set V of vertices represents the M service provider accounts and the set E represents the created payment channels among M service providers and N customers. Every edge between service providers has a capacity u that denotes the amount of depositable Bitcoins. Every vertex $v \in V$ has an associated total-capacity C that denotes the upper bound of forwarded Bitcoins over it. Based on these inputs, how can we create a virtual topology PCN among the suppliers in such a way that 1) the average transaction fee for a customer will be minimized; 2) the total investment made by a provider for creating channels with its neighbors (i.e., the cost related to the cration of capacity u) will be minimized; and 3) the privacy of customers will be preserved."

4.2 Proposed Optimization Model for PCN Creation

4.2.1 Overview

The proposed PCN will form a virtual topology on top of the on-chain operations, thus providing a valuable infrastructure which is able to guarantee P2P payment service without requiring any on-chain transactions. Before elaborating on the details of the proposed solution, we first emphasize some important points and summarize the overall approach.

The optimization of the placement of payment channels utilized by a PCN, as well as the fairness in the allocation of network design costs, are fundamental issues for the members who pay for the costs involved. While designing the network, the payment channels to a store should have enough capacity for both routing payments of others and for receiving the payments intended for that station. Moreover, the channel capacity between the stations should be organized in such a way that stations with similar intent and opportunity should contribute to the network in a similar way.

Our approach considers all these issues by proposing a mixed integer programming model inspired by multi-commodity network flow problem [EIS75]. The model minimizes the PCN design cost according to different cost sharing scenarios. These costs are the network flow cost, link establishment cost and unfairness cost in capacity distribution among the stations.

4.2.2 Formulation of the Model

There are N users in the network with I defining the set of the customers (e.g., EV users) and $I = \{EV_1, EV_2, \dots, EV_N\}$ and $N = |I|$. Similarly there are M stores (e.g., car charging stations) (denoted as ST) in the payment system and P is the set of the stores and $P = \{ST_1, ST2, \dots, ST_M\}$ and $M = |P|$. From now on, the term "node", "service provider" and "station" will be used interchangeably. Each EV owner will be registered to a station. Whenever a customer receives service from a store, a payment will be initiated from the registered store to the payee store. Since there will be a money traversal the problem can be formulated with graph representation where vertices are the EV users and the stations and edges are the links between EV users and stations and links in between stations. Let the graph $G(V, E)$ defined such that $V = I \cup P$ and $E = \{[(i, j) : i \in I, j \in P] \cup [(j, j') : j, j' \in P]\}$ Each EV owner, EV_i, is assumed to be making payments to a set of stations, J_i, during the payment period (these refer to the stations within the areas typically traveled by the EV). This period can be of any length, which impacts the size of J_i. Hence, $|J_i|$ is the expected total number of payments that will be made by EV owner EV_i. Let a_{ij} be the expected payment amount by EV owner $i \in I$ to store $j \in J_i$ during the planning period. Given these initially known values, we can define the total "supply" of an EV owner as:

$$s_i = \sum_{j \in J_i} a_{ij} \tag{4.1}$$

for all $i \in I$, and the total "demand" of a store as:

$$d_j = \sum_{i \in I} a_{ij} \tag{4.2}$$

for all $j \in P$.

Let decision variable $u_{jj'}$ denote the capacity of the payment channel on edge (j, j') to be established between $j, j' \in P$, where each unit of capacity incurs a variable cost of $c^v_{jj'}$ (i.e opportunity cost for keeping bitcoins in reserve). In our model, to represent a real-time scenario, we assume that setting up a payment channel will carry a fixed "channel establishment fee (i.e., Bitcoin transaction fee)".

So, a fixed cost of $c^f_{jj'}$ is assumed if a payment channel is established between $j, j' \in P$.

Suppose that the optimization objective involves the minimization of a function of the total cost of establishing payment channels across the entire network (involving fixed and variable costs), while ensuring that all payments by EV owners will be processed following some path on the network where the termination node of the path corresponds to the recipient of a given payment.

To allow for a multi-commodity flow type integer programming formulation which is known to be NP-Complete [Pap81], we further define the following decision variables. We let x^k_{ij} define the payment flow on arc (i, j) for $i \in I, j \in P$ intended for station $k \in J_i$. Moreover, $y^{ik}_{jj'}$ refers to the flow on arc (j, j') for $j, j' \in P$ which has originated from EV owner $i \in I$ with destination $k \in J_i$. In order to indicate the channel 'opening' decision between any two stores, we define the binary variable $z_{jj'}$ such that $z_{jj'} = 1$, if there is positive flow on arc (j, j') for $j, j' \in P$, and $z_{jj'} = 0$, otherwise.

We then can formulate the optimization problem through equations (4.3)-(4.10) as follows:

$$\min \sum_{j \in P} \sum_{j' \in P} c^v_{jj'} u_{jj'} + c^f_{jj'} z_{jj'} \tag{4.3}$$

$$\text{s.t.} \sum_{j \in P} x^k_{ij} = a_{ij} \qquad \forall i \in I, k \in J_i \tag{4.4}$$

We need to have the node transaction conservation equations as:

$$\sum_{j' \in P} y^{ik}_{j'j} + x^k_{ij} - \sum_{j' \in P} y^{ik}_{jj'} = a_{ij} \qquad \forall j, i, k \in J_i \tag{4.5}$$

In addition, the capacity of the links should be large enough to accomodate the flows on the arcs and the fixed cost structure should be defined, while channel capacity from one node to the other should be symmetric:

$$\sum_{i\in I}\sum_{k\in J_i} y_{jj'}^{ik} \leq u_{jj'} \qquad \forall j, j' \in P \quad (4.6)$$

$$\sum_{i\in I}\sum_{k\in J_i} y_{jj'}^{ik} \leq C' z_{jj'} \qquad \forall j, j' \in P \quad (4.7)$$

$$u(j, j') = u(j', j) \qquad \forall j, j' \in P \quad (4.8)$$

$$x, y, u \in R^+, z \in \{0, 1\} \qquad (4.9)$$

where C' is some upper bound for the capacity on a given channel. That C' can also be chosen different for each possible arc.

Finally, in order to assign an EV user to a single predefined station, we include the following equation:

$$x_{ij}^k = a_{ik} \qquad \forall i \in I, k \in J_i \text{ and } j = ST_{ij}^k \quad (4.10)$$

For that equation to hold ST_{ij}^k is a station chosen randomly for delivering a payment from user i to station k starting at station j.

4.2.3 Fair Distribution of Network Design Costs Among Nodes

Being a store in the PCN will impose responsibility to the manager of that store to open payment channels with other peers more than its own demand so that it can forward the payments of other stores. However, these payment channels have an associated cost due to keeping Bitcoin in escrow and related transaction fees. Therefore, the members of PCN strive to keep their investment costs at minimum while participating in the formation of a PCN.

Thus, in the optimization, a new cost, namely unfairness cost represented by Γ, is added in order to provide a mechanism for both sharing network formation costs fairly, and also for obtaining a decentralized network topology. For that purpose the nodes with the minimum and the maximum outbound flows are determined, and the difference between them are multiplied by a cost parameter γ to define Γ as follows:

$$totFlows_j = \sum_{i \in I} \sum_{k \in J_i} \sum_{j' \in P} y^{ik}_{jj'} \qquad \forall j \in P \quad (4.11)$$

$$maxFlow = max_j(totFlows_j) \quad (4.12)$$

$$minFlow = min_j(totFlows_j) \quad (4.13)$$

$$\Gamma = \gamma * (maxFlow - minFlow) \quad (4.14)$$

Note that $maxFlow$, $minFlow$, and $totFlows_j$ are nonnegative decision variables in the appended formulation with fairness considerations.

4.3 Evaluation

4.3.1 Experimental Setup and Implementation

The integer programming formulation for the optimization model was solved using Gurobi Optimizer version 7.2 [Opt17]. The computer running the software has Intel Xeon E5-2630 v4 @ 2.20GHz CPU and 64 GB of RAM and running on Arch Linux Kernel version 4.14.4.

For the optimization implementation, the available number of stations are denoted as $number\_of\_stations$, the number of customers, i.e. $number\_of\_EV$, will be 8 times the $number\_of\_stations$ in the model. The payment planning period may change based on the habits of the customers. For a typical network, the payment plan and the optimization can be done on a daily basis, however, for certain setups this can be done on a weekly or monthly basis. To this end, in our model we assume that a customer may make a payment to 6 different stations which is $payment\_per\_EV$ over the planning period.

For the baseline optimization scenario, the unit flow cost between stations, $c^v_{jj'}$, is set to 1. The channel establishment fee cost multiplier is set to be equal for every link, i.e. $c^f_{jj'} = c^f$ and it varies between 50 and 650. γ varies from 20 to 80. For each customer, the payee stations set is generated randomly. A customer is registered to another random station, which is different from the stations in the payee set. For ease of following the transactions and result comparisons, each transaction is selected to be 10 units, i.e., each customer supplies 10 times $payment\_per\_EV$ to the network. Since stations are created randomly, the demand of a station may be different from another one. $number\_of\_stations$, $number\_of\_EV$ and $payment\_per\_EV$ are set to be 10, 80 and 6 respectively. The topology with these variables is created once, and is used throughout the tests unless otherwise stated. [1]

4.3.2 Metrics and Benchmarks

The solution to the optimization problem is evaluated based on the following metrics:

[1]These cost multipliers do not necessarily correspond to one-to-one currency replacement. They can more be defined as scores or willingness that the network partners think that may have a high impact or a low impact in network establishment from their perspective.

- **Betweenness centrality of a node:** Betweenness centrality of a node is a measure for the node which shows, in a network, how many times a node is visited while traveling between other nodes in the possible shortest distance routing. For example, for a fully connected network, it is expected that the betweenness centralities for all nodes are to be small. However, for a hub-and-spoke network the central nodes will have high betweenness centrality value.
- **Total Capacity of the Network:** This metric basically represents total required investment amount in the escrow accounts to form a PCN network among participants.
- **Number of Edges:** This metric shows the resulting number of payment channels created among the network.

We compared our approach against certain benchmarks. Specifically, we first consider benchmarks for creation of the network topology among stores as follows:

- **Random network topology** In this approach, we assumed that the topology of the network is random. The edges are created by assuming that each station has more than 2 connections to other stations [2] Our optimization model is applied on this randomly generated network.
- **Fixed Hub and Spoke topology:** In this approach, 2 stations are picked to be the most central stations and remaining stations are divided into 2 groups. Each of these groups are directly connected to those main stations. Again, optimization is carried out for that network.

[2]The reason for selecting to form a 3-connected network is that, the best network in terms of betweenness centrality is observed to be a 3-connected network.

4.3.3 Experiment Results and Discussion

Betweenness Centrality Comparison

Betweenness comparison of the networks is shown in Fig. 4.2. In the figure, note that, x-axis shows the node number re-named according to descending centrality score, y-axis is the centrality score. Hub-and-spoke network shows high betweenness for 2 nodes (center hubs) and the betweenness centrality value of the other stations is 0. Compared to hub and spoke network, there is a big change in betweenness centrality for random connected network. However, the optimized networks give better centrality results. Especially the network optimized with $c^f = 650$ and $\gamma = 20$ gives a flat betweenness centrality graph (0.083 centrality value), which yields a flat distribution of connections. As expected, optimized networks yield a more balanced network topology.

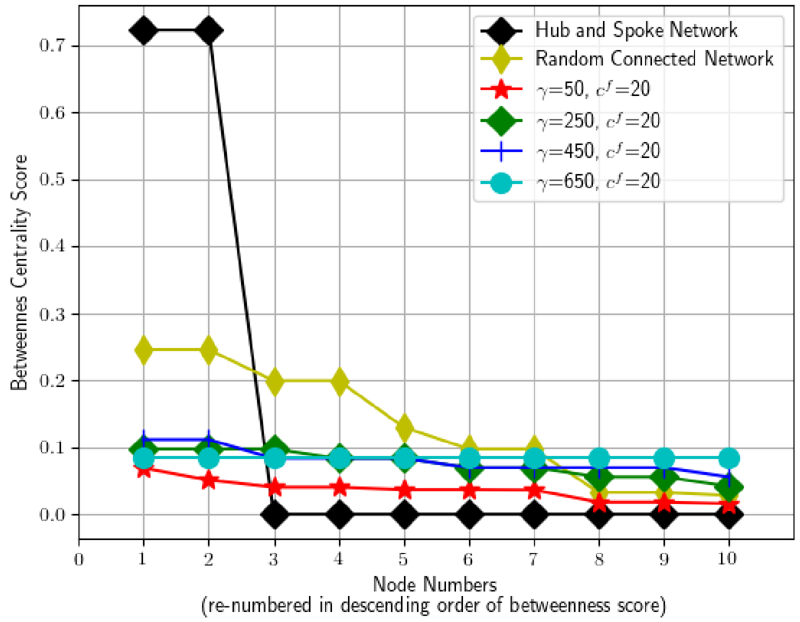


Figure 4.2: Betweenness centrality comparison between the networks with respect to γ and c^f

Total Capacity of the Networks

In this experiment, for our approach we fixed γ while varying c^f. Fig. 4.3 shows the resulting total capacity for different network topologies. As can be seen, our approach helps the participants invest less money into the escrow accounts compared to hub-and-spoke and random connected topologies. When c^f decreases, our approach strives to reduce the total capacity as will be justified shortly. Note that in the experiments, total amount of money supplied by all EV owners is fixed which is 4800 units. From a business perspective, it can be argued that not so many people want to invest much more than the amount they will earn. Our approach ensures this. For instance, for our approach, the total capacity is always less than 7000 units while for hub-and-spoke the total capacity is over 10,000 units.

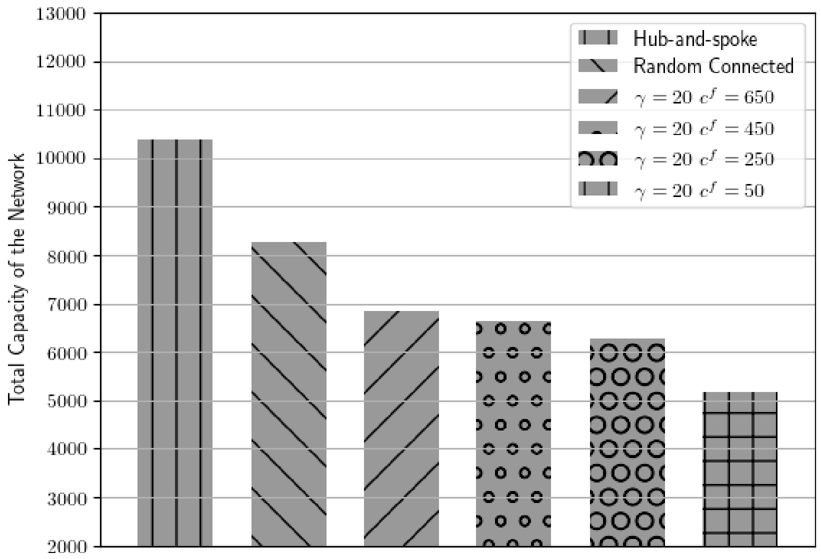


Figure 4.3: Capacity of the networks

Total Edges in the Network

Fig. 4.4 depicts the number of edges established in resultant network topologies. Picked cost parameters plays an important role in the number of established edges.

Decreasing c^f causes an increase in number of edges as connection fee cost starts to become less dominant in total cost calculation. Although fewer number of connection seems as a good choice, e.g., ring topology, relay stations for a particular transaction will have to invest more during the network establishment. Additionally, some of the stations might experience single point of failure problem in case of link terminations.

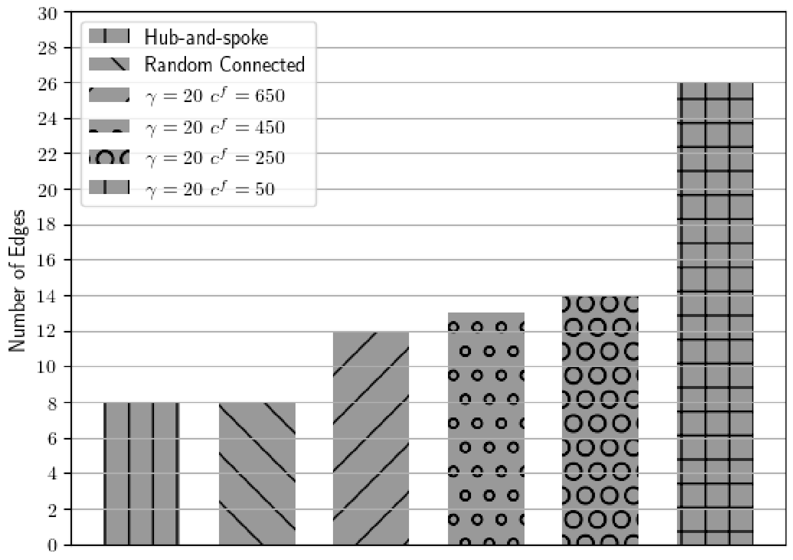


Figure 4.4: Number of edges in networks

Overall, we can speculate that when there are fewer edges in the network, the transactions have less options to travel and thus they follow the available paths that are limited. In other words, there is not much liberty for a transaction. It has to follow longer routes in many cases. However, when there are more edges in the topology (i.e., when $c^f = 50$ as shown in Fig. 4.4), this provides more options in terms of shortcuts for their travel paths. This means, instead of using multiple hops for a transaction, a single hop can be used which will reduce total capacity cost (see Fig. 4.3. As a result, when there is increased number of edges, our approach is able to reduce the total capacity of the network which is not the case in hub-and-spoke model.

Effect of Parameter Selection on Performance

The total cost is affected mainly by 3 parameters. c^f, γ and c^v have direct impact on the cost, reader is encouraged to refer to Equations 4.3 and 4.14. Bi-directional channel establishment requirement given in Equation 8 has an intrinsic effect in network establishment that was discussed.

Effect of γ Parameter Selection. Noting the random distribution in payments for the network, the increase in γ causes an increase in number of edges in the final network, as shown in Fig. 4.5. From the experiments we observe that as the unfairness cost increases the solver opens new channels in order to create a more balanced flow in the network. Making the number of edges high comes with the advantage of a decrease in the total flow in the network. Increase in γ makes a decrease in total flow in the network as shown in Fig. 4.6.

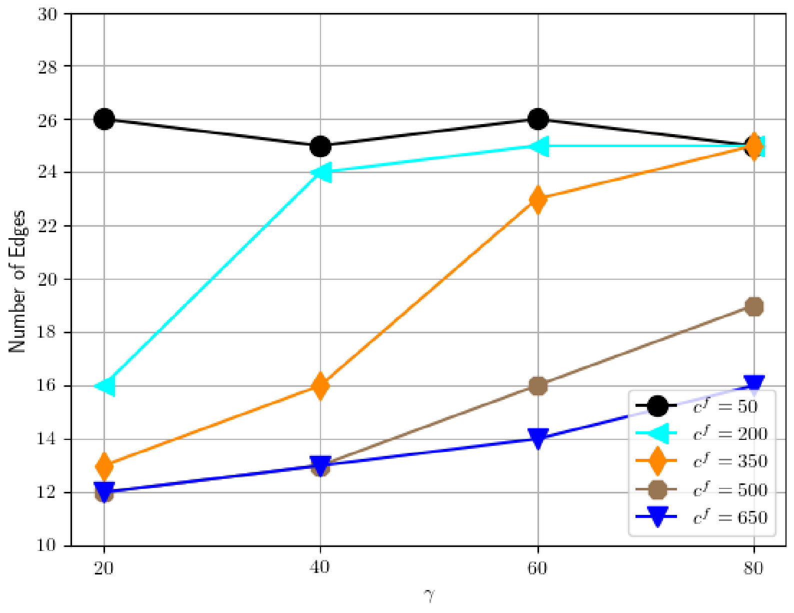


Figure 4.5: Number of edges in the network vs. γ for different c^f

Effect of c^f Parameter Selection. In order to minimize the total fee paid by the stations, number of established links should be minimized as much as possible.

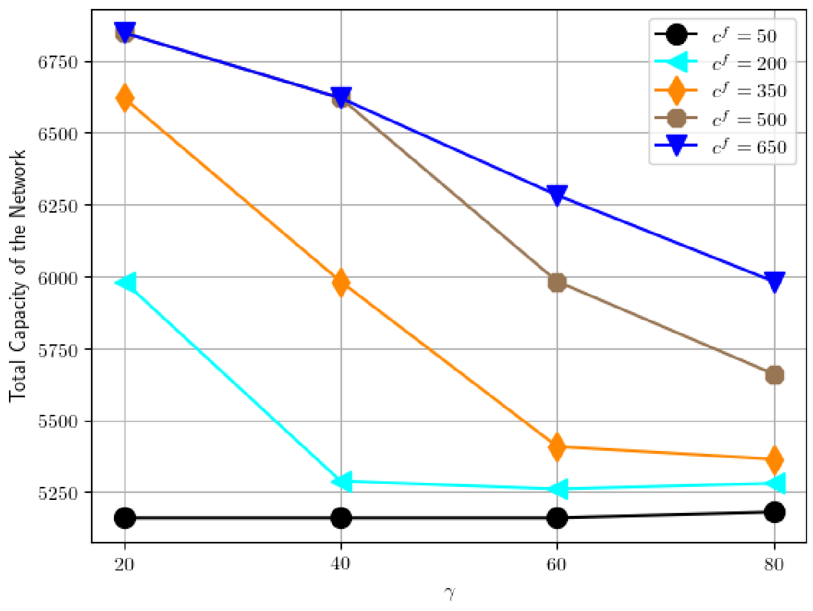


Figure 4.6: Total Capacity of the network vs. γ for different c^f

Established links are designated by binary variables and cost of a link is calculated by multiplication of the binary variable by c^f. The effect c^f is shown in Figures 4.7 and 4.8. Fig. 4.7 shows that for a constant γ, an increase in c^f causes a decrease in the number of edges as expected. As explained previously, the decrease in the number of edges causes an increase in the total capacity. That is transactions start to traverse either more hops or in a fragmented fashion.

Scalability of the Model

For comparison of the scalability of the model, we included the time required for the solver to solve the problem in Fig. 4.9. More acceptable network topologies are obtained with solutions which take longer times. Increasing the number of stations by one will almost double the number of equations in the model definition. Hence, required time for the optimization increases exponentially with the number of stations. A change in the EV count or the number of payments per EV user also

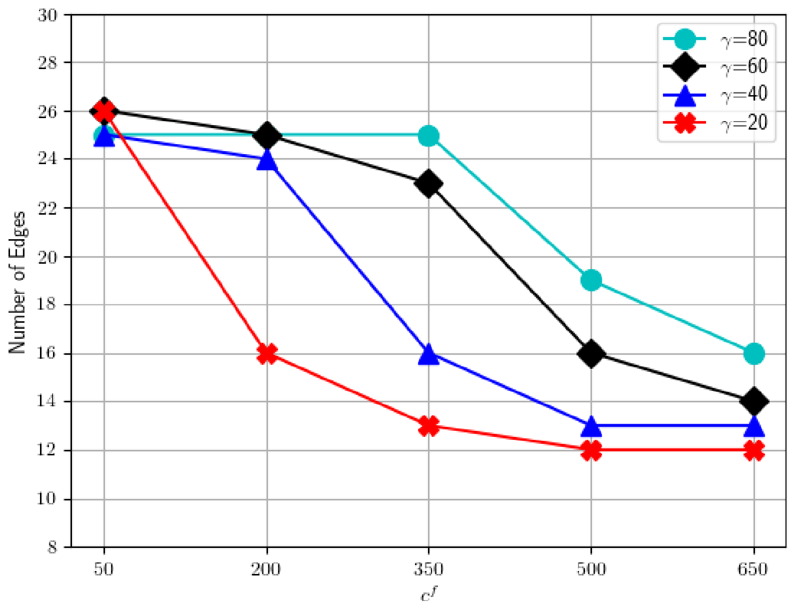


Figure 4.7: Number of edges in the network vs. c^f for different γ

adds up to the number of equations. Hence, for a large number of stations optimum solution time will increase tremendously.

4.4 Conclusion

In this chapter, we designed a private payment network for Bitcoin transactions in a network of customers and stores. The objectives were to eliminate the high transaction fees and verification times by using the off-chain concept of Bitcoin and establishing a privacy-aware payment network thanks to anonymous behavior of public blockchain based cryptocurrency. We formed an overlay payment network using an optimization model based on a multi-commodity flow problem structure.

The resultant topology ensured that it minimizes the costs for establishing channels among stations. The topology also favors privacy since it prevents formation of hub nodes that can potentially monitor the source and destination of all payments.

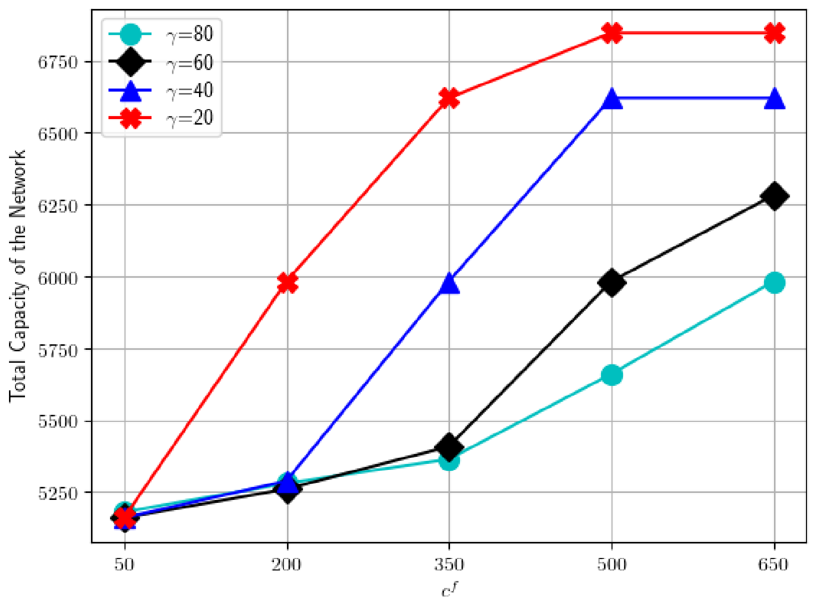


Figure 4.8: Total Capacity of the network vs. c^f for different γ

The results indicate that the topology can significantly save transaction fees for customers. As the number of stations increase the problem becomes harder as this mixed integer problem optimization type is an NP-complete problem [Pap81].

In the next chapter, we will investigate heuristics for large-scale formation of the payment network topology.

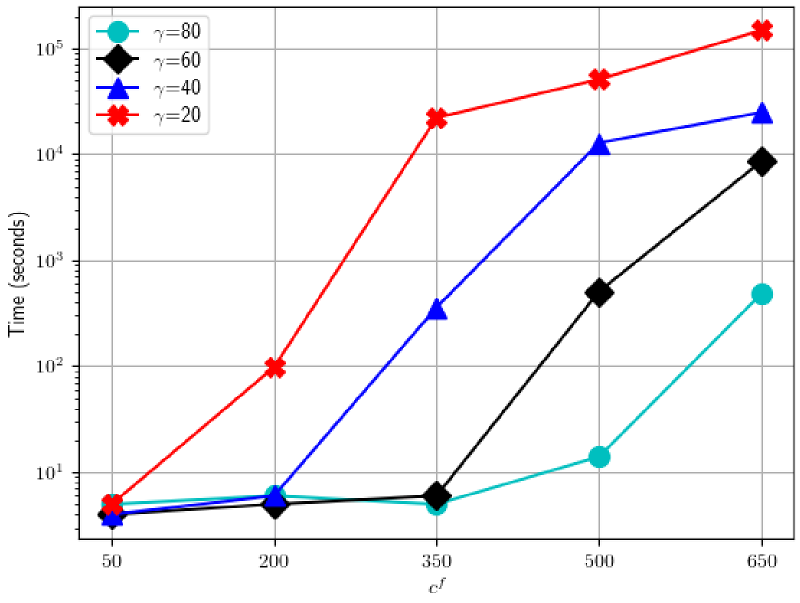


Figure 4.9: Time elapsed vs. c^f for different γ

CHAPTER 5

A PAYMENT NETWORK FORMATION HEURISTIC BY LINEAR PROGRAMMING

In Chapter 4 we have developed an optimization model for a decentralized payment channel network formation. However, due to the computational complexity of the method, the model can not scale more than 10 nodes. In this chapter, we propose to build such a private payment channel network topology from scratch via linear programming by pruning network's links so that we can pick the right network topology by monitoring certain metrics instead of following a brute force approach. In addition, we also minimize the number of channels to reduce network formation costs. However, pruning needs to be done carefully as it may favor some of the channels or paths that will create an unfair distribution (i.e., create hub nodes and put a burden on these hubs). Therefore, we consider several criteria in pruning and continue such a process until we achieve a certain standard deviation among the capacities of the channels. We also consider the trade-offs between the capacity and standard deviation.

The evaluations using *Python* and *Gurobi solver* indicated that our proposed heuristics can provide comparable performance to that of the optimal solution while allowing scalability. In addition, the heuristic creates purely distributed topologies that are resilient towards DDoS attacks and potential privacy leakages.

5.1 Suitability of LN for Real-time Payments

As LN is geared for micropayments, the urgency of payments could sometime be important to be used in real life applications such as buying a cup of coffee. Therefore, before moving to the problem definition we instantiated real LN nodes on the test network to measure the round trip time (RTT) for a payment. We measured the RTT in seconds for various number of hops and showed the results in Table 5.1. 2-hop payments are done to *htlc.me*, 3-hop payments are done to *starblocks.acinq.co*. Entrance node is connected to the LN via Tor network so that it introduces additional delays to the communication. The length of an LN payment packet is 1366 Bytes and the permissible number of hops is 20 [Lig16]. The connection between the nodes can be of any type, i.e., wired, wireless, optical or combination of any of those. The RTT measurements suggest that even though the number of hops grows, the delay only increases slightly in seconds and thus LN is a viable network architecture proposal for instant cryptocurrency payments. We would like to also note that a longer RTT does not constitute to double spending problem as it does in the Bitcoin network. This is because the transactions are not immediately written to the blockchain but the state of the channel is always tracked by the peers, whenever one peer tries to forge the channel the other can open a dispute on Bitcoin network and get all of the money in the channel with a valid proof which is a disincentive against forging.

Table 5.1: RTT measurements for various number of hops.

Number of Hops	Min (s)	Avg (s)	Max (s)
2	2.3	2.7	4.3
3	2.3	2.9	5.4

5.2 Problem Motivation and Definition

Even though the concept of LN is very attractive to introduce Bitcoin in micropayment market, its current structure requires the deployment of relay nodes which will act as bridges between customers and retailers. This brings two major issues: First, these relay nodes charge forwarding fees and thus the goal of minimizing transaction fees through the payment network will be violated. Second, eventually, these relay nodes will become major hubs in the network creating the risk of having DDoS attacks against these nodes to stop the payments in the network at any time. The third risk here is regarding customer privacy. If these relay nodes are compromised, they can easily analyze the payments passing through them which will expose the privacy of the customers using them as relays.

These risks may deter any business to adapt the current LN for its payments. Hence we argue that a planned approach for creating a network topology from scratch is needed for forming a **private payment network** because of the following observations/issues on the current LN topology:

- **Network connectivity:** LN had a basic assumption that a payment network can be formed by mostly random connections without a topology plan. This assumption is not valid since there will be a certain probability of connectivity success which means that the final payment network may not be connected or some of the nodes can be loosely connected to the network. As a result, payment options for a customer will be limited. To solve this issue the developers of `lnd` (Lightning Network Daemon) introduced the optional '`Autopilot`' feature to help a node to initialize connections to other nodes in the network. `Autopilot` uses constrained Barabasi-Albert (BA) method which is being used to model connections in social networks [Sta18]. However, we believe this approach, which relies on "network influencers" (i.e., hubs).
- **Investment need for each channel:** Forming a connected network will not be free. Each channel establishment results in two mandatory on-chain transactions. Hence, the number of channels established by a node should be kept optimum, namely, high enough to keep the needed transactions flowing but low enough to decrease the on-chain fees.

- **Partial usage of available payment capacity:** A node may assume that it needs 100 Bitcoins worth of total transaction volume for its own business. However, most of the times that capacity will be used by other nodes which use this node as a relay. Thus, at a given time, only a portion of the capacity will be available for the node itself to accept transactions from its own customers. This implies that one should invest much more than its planned transaction volume when a consortium of business come together to form a private payment network.
- **Diminishing channel capacity over time:** An important and unique property of LN is that the capacity of channels diminishes over time. At the initial creation of a channel, it will be set to have a maximum forwarding capacity. When nodes use this channel for payment forwarding, its capacity degrades each time. For resolving this issue, either more investment should be done for the channel or there should be a reverse payment forwarding to balance the capacity. This feature needs to be incorporated into the design of a new topology.

We propose to utilize the idea of creating payment channels among the retailers of a consortium assuming that every node in this network will create in-advance payment channels (i.e., links with certain transaction capacities) with some other nodes as shown in Fig. 5.1.

Based on these discussions, our problem can be formally defined as follows: "*Let us assume M retailers and N customers. Let us also assume that a payment channel network (PCN) can be represented as a graph $G = (V, E)$, where V represents all Bitcoin accounts (of M retailers) and E represents all payment channels among M retailers. Every edge between retailers has a capacity that denotes the amount of depositable Bitcoins (i.e., the actual investment put in the escrow account by a*

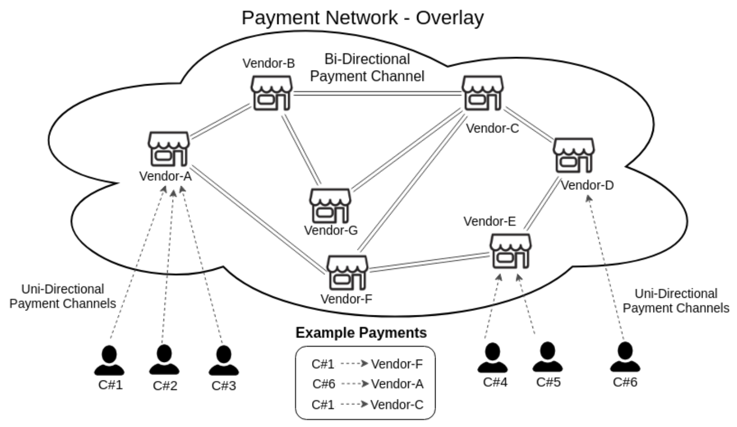


Figure 5.1: An overview of the envisioned Payment Network among retailers.

retailer). We assume that every vertex (retailer) $v \in V$ will make an initial total investment that represents the maximum Bitcoins that can be transmitted or forwarded over it. In other words, we are considering the maximum possible instantaneous payments that can be made from a retailer or forwarded by it. This can also be described as the maximum possible business capacity of a retailer within a certain time. Based on these inputs, how can we create a virtual topology PCN among the retailers in such a way that 1) the average transaction fee for a customer will be minimized; 2) the total investment made by a retailer for creating channels with its neighbors will be minimized; 3) the standard deviation of total investment costs among the retailers will be minimized."

5.3 Flow and Network Optimization Model

While designing the network, the payment channels between retailers should have enough capacity for both routing the payments of others and for receiving the payments intended for themselves. Moreover, the channel capacity between the retailers should be created in such a way that retailers with similar intent and opportunity

Table 5.2: Notations and their explanations

Symbol	Meaning
$u_{jj'}$	Channel capacity from node j to j'
$z_{jj'}$	Binary decision variable, equals 1 if $u_{jj'} > 0$, equals 0 else
$c^{v}_{jj'}$	Cost multiplier for capacity from node j to j'
$c^{f}_{jj'}$	Cost multiplier for channel establishment from node j to j'
γ	Parameter to control unfairness between the nodes
x^{k}_{ij}	Flow originated from customer i, flowing from j destined to k
$y^{ik}_{jj'}$	Flow originated from customer i destined to node k, flowing from j to j'
a_{ik}	Total amount of intended payment originating from customer i destined to node k
$totFlows_j$	Sum of all flows originating from node j
Γ	Total cost of unfairness among nodes
$IUBpN$	Investment upper bound per node

(i.e., in terms of business capacity) should contribute to the network in a similar way.

Although mixed-integer programming approach is discusses in Chapter 4, for the completeness of the discussion it will be visited again in this chapter. The notations and their explanations are shown in Table 5.2.

5.3.1 Formulation of the Network Optimization Model

Let us assume that there are N users in the network with I defining the set of the customers where $I = \{Cu_1, Cu_2, \ldots, Cu_N\}$ and $N = |I|$. Similarly, let us assume that there are M retailers (denoted as "RT") in the payment system and P is the set of the retailers and $P = \{RT_1, RT_2, \ldots, RT_M\}$ and $M = |P|$. From now on, the term "node" and "retailer" will be used interchangeably. Each customer will be registered to a retailer. The payment of a customer will be initiated from the registered retailer to the payee retailer. Since there will be a money traversal the problem can be formulated with graph representation where vertices are the retailers and edges are the links between retailers. Let the graph $G(V, E)$ be defined such that $V = I \cup P$ and $E = \{[(i,j) : i \in I, j \in P] \cup [(j,j') : j,j' \in P]\}$. Each customer, Cu_i, is assumed to be making payments to a set of retailers, J_i, during the payment period. This time period can be of any length. Hence, $|J_i|$ is the expected total number of payments that will be made by Cu_i. Let a_{ij} be the expected payment amount by $Cu_i \in I$ to the retailer $j \in J_i$ during the planning period.

Let decision variable $u_{jj'}$ denote the capacity of the payment channel (i.e., Bitcoins put on the escrow account) on edge (j, j') to be established between $j, j' \in P$, where each unit of capacity incurs a variable cost of $c^v_{jj'}$ (i.e., the opportunity cost for keeping Bitcoins in reserve since this money will stay there untouched and thus there will be no way to use it in other profitable investments). In our model, to represent a real-time scenario, we assume that setting up a payment channel will carry a fixed "channel establishment fee (i.e., on-chain transaction fee)". So, a fixed cost of $c^f_{jj'}$ is assumed if a payment channel is established between $j, j' \in P$.

Suppose that the optimization objective involves the minimization of a function of the total cost of establishing payment channels across the entire network ensuring that all payments by customers are processed following some path on the network where the termination node is the recipient of a given payment.

To allow for a multi-commodity flow type integer programming formulation, which is known to be NP-Complete [Pap81], we further define the following decision variables. We let x^k_{ij} define the payment flow on arc (i, j) for $i \in I, j \in P$ intended for retailer $k \in J_i$. Moreover, $y^{ik}_{jj'}$ refers to the flow on arc (j, j') for $j, j' \in P$ which has originated from customer $i \in I$ with destination $k \in J_i$. In order to indicate the channel 'opening' decision between any two retailers, we define the binary variable $z_{jj'}$ such that $z_{jj'} = 1$, if there is positive flow on arc (j, j') for $j, j' \in P$, and $z_{jj'} = 0$, otherwise.

We then can formulate the optimization problem through equations (5.1)-(5.10) as follows:

$$z_{jj'} = \begin{cases} 1, & u_{jj'} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

$$\min \sum_{j \in P} \sum_{j' \in P} c^{v}_{jj'} u_{jj'} + c^{f}_{jj'} z_{jj'} \tag{5.2}$$

$$\text{s.t.} \sum_{j \in P} x^{k}_{ij} = a_{ik} \qquad \forall i \in I, k \in J_i \tag{5.3}$$

and Eq. (5.1) can be modeled as:

$$u_{jj'} \leq \epsilon z_{jj'} \tag{5.4}$$

$$u_{jj'} \geq \delta z_{jj'} \tag{5.5}$$

where ϵ is the upper bound for the decision variable $u_{jj'}$ and δ is a positive small number close to the lower bound (e.g. 0.001). That constraints will force the solver to find a suitable z either 0 or 1 as z is defined to be a binary number which takes the possible solutions for z from linear range to integer range. Eq. 2 is the objective function and Eq. 3 states that for Cu_i sum of all money supplied to the network through any node for RT_k equals to a_{ik}. It basically means that a customer can register to more than one retailer.

We need to have the node transaction conservation equations as:

$$\sum_{i \in I} \sum_{k \in J_i} \sum_{j' \in P} y^{ik}_{j'j} + \sum_{i \in I} \sum_{k \in J_i} x^{k}_{ij} - \sum_{i \in I} \sum_{k \in J_i} \sum_{j' \in P} y^{ik}_{jj'} = d_j \qquad \forall j \in P \tag{5.6}$$

where d_j defines the total '*demand*' of retailer j as $d_j = \sum_{i \in I} a_{ij}$ for all $j \in P$. Eq. 4 states that for a node j, i.e. RT_j, sum of all of the flows coming to RT_j minus sum of all of the flows leaving RT_j must be equal to the demand of RT_j.

In addition, the capacity of the links should be large enough to accommodate the flows on the arcs and the fixed cost structure should be defined. As the established channel can be bidirectional, we further assume that "channel capacity" from one node to the other is symmetric. Note that the amount that will be put into the escrow accounts can be decided by the peers at the time of agreement:

$$\sum_{i\in I}\sum_{k\in J_i} y_{jj'}^{ik} \leq u_{jj'} \qquad \forall j,j' \in P \quad (5.7)$$

$$\sum_{i\in I}\sum_{k\in J_i} y_{jj'}^{ik} \leq C' z_{jj'} \qquad \forall j,j' \in P \quad (5.8)$$

$$u(j,j') = u(j',j) \qquad \forall j,j' \in P \quad (5.9)$$

$$x, y, u \in R^+, z \in \{0,1\}$$

where C' is some upper bound for the capacity on a given channel. This C' can also be chosen different for each possible arc. For this study C' is set to a constant high value for every edge.

Finally, in order to assign a customer to a single predefined retailer, we include the following equation:

$$x_{ij}^k = a_{ik} \qquad \forall i \in I, k \in J_i \text{ and } j = RT_{ij}^k \quad (5.10)$$

For that equation to hold RT_{ij}^k is a retailer chosen randomly for delivering a payment from customer i to retailer k starting at retailer j.

5.3.2 Fair Distribution of Network Design Costs Among Nodes

A member retailer of the PCN should open payment channels with other peers more than its own demand so that it can relay the payments for others. However, these payment channels have an associated cost due to keeping Bitcoin in escrow and related transaction fees. Therefore, the members of PCN strive to keep their investment costs at a minimum while participating in the formation of a PCN.

Thus, in the optimization, a new cost, namely, *unfairness cost* represented by Γ, is introduced in order to provide a mechanism for sharing network formation costs fairly. For that purpose, capacity difference between nodes with maximum and minimum outbound flows are multiplied by the parameter γ as follows:

$$totFlows_j = \sum_{i \in I} \sum_{k \in J_i} \sum_{j' \in P} y_{jj'}^{ik} \qquad \forall j \in P \quad (5.11)$$

$$maxFlow \geq totFlows_j, \qquad \forall j \in P \quad (5.12)$$

$$minFlow \leq totFlows_j, \qquad \forall j \in P \quad (5.13)$$

$$\Gamma = \gamma(maxFlow - minFlow) \qquad (5.14)$$

Note that $maxFlow$, $minFlow$, and $totFlows_j$ are non-negative decision variables. Given the minimization objective, the optimization should ensure that the $maxFlow$ is set equal to the highest of $totFlows_j$ values, and $minFlow$ is set equal to the lowest of $totFlows_j$ values.

The final objective function becomes

$$\min(\sum_{j \in P} \sum_{j' \in P} c_{jj'}^{v} u_{jj'} + c_{jj'}^{f} z_{jj'} + \gamma(maxFlow - minFlow)) \qquad (5.15)$$

Throughout the rest of the chapter, the model described in Sections 5.3.1 and 5.3.2, combined, will be referred to as mixed-integer optimization (MIOP) model.

5.4 Pruning-based Heuristic

As the presented MIOP model falls under the category of NP-hard problems in general, the computational complexity would be high and thus the model will not scale. Therefore, in this section, we present a heuristic approach to improve the scalability of the MIOP model.

5.4.1 Simplifying the MIOP Model

In our heuristic approach, MIOP model is modified in such a way that the solver will not decide on the links to be established and the unfairness among nodes. Specifically, $z_{jj'}$ and Γ variables are removed from the model. Hence, the model turns into linear programming (LP) one. Throughout the rest of the chapter, this modified model will be called the *LP model.* LP model basically only optimizes the flows in the network.

Furthermore, in our heuristic, the standard deviation among flows (i.e., unfairness) is controlled by assigning an investment upper bound per node (IUBpN) for all the flows passing through a channel as shown in Eq. 5.17. This approach not only removes the computational overhead in the model but also decreases the possible solution domain of the model and intrinsically helps in shortening the computational time.

$$totFlows_j = \sum_{j' \in P} u_{jj'} \qquad \forall j \in P \quad (5.16)$$

$$totFlows_j \leq IUBpN \qquad \forall j \in P \quad (5.17)$$

5.4.2 Algorithm for Building the Topology

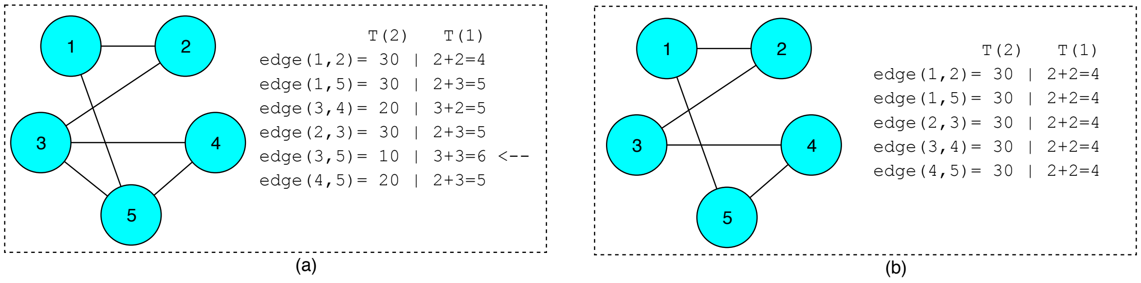
In order to keep the problem in the LP domain (with a polynomial time algorithm [Kha80]), we build the network manually and let the LP model calculate the flows in the network in polynomial time. Note that for network flow calculations there are many other alternative algorithms like Ford-Fulkerson [FF56] or Dinic's algorithms [Din70]. However, in these greedy algorithms, the result at each step varies based on the nodes selected initially. In other words, the algorithm becomes order oriented and thus finding the best order becomes a challenge. To prevent this challenge, we opted to stick with the LP model to get rid of the order dependency.

Once the flows are determined by the LP, we start pruning certain edges of the initial network topology that carry the created flows in an iterative fashion by applying the following procedure:

Algorithm 1 Network Establishment

```
1: Input: $G(V,E)$=Fully connected undirected graph, $V$ representing set of nodes,
   $E$ representing set of edges, $E_{jj'}$ representing edge between $j$ and $j'$
2: Input: Anticipated payment plan
3: Using LP create flow model $\boldsymbol{m}$ using $\boldsymbol{Edges}$ and payment assumptions
4: Create a 2-dimensional Table: $T[\,][\,]$
5: Solve $\boldsymbol{m}$
6: while m is feasible do
7:     for $E_{jj'}$ in $E$ do
8:         $T[E_{jj'}][1]$ = total number of edges $j$ and $j$' have
9:         $T[E_{jj'}][2]$ = capacity of $E_{jj'}$
10:    end for
11:    Sort $T$ in descending order w.r.t. $T[*][2]$
12:    Sort $T$ in ascending order w.r.t. $T[*][1]$
13:    Remove the last edge in $T$
14:    Update $\boldsymbol{m}$
15:    Solve $\boldsymbol{m}$
16:    Reset $T$
17: end while
18: Output: $G$
```

The algorithm, pseudo-code of which is shown in Algorithm 1, starts with a fully connected network topology. Customer registration information, payment plan, and edge connections are fed to the solver. The solver calculates the optimal transaction flow by considering the availability of the connections between nodes and upper bounds for the decision variables. When the flows are obtained, the edge capacities and edge counts of the nodes are temporarily saved. In order to start pruning, we create a table where every edge in the final flow diagram is tracked. Let us assume we track an arbitrary edge (j, j'). The first column of the table would represent



```
                T(2)    T(1)
edge(1,2)=  30  |  2+2=4
edge(1,5)=  30  |  2+3=5
edge(3,4)=  20  |  3+2=5
edge(2,3)=  30  |  2+3=5
edge(3,5)=  10  |  3+3=6  <--
edge(4,5)=  20  |  2+3=5
```

```
                T(2)    T(1)
edge(1,2)=  30  |  2+2=4
edge(1,5)=  30  |  2+2=4
edge(2,3)=  30  |  2+2=4
edge(3,4)=  30  |  2+2=4
edge(4,5)=  30  |  2+2=4
```

Figure 5.2: A scene from the flow of the pruning heuristic. a) Initial table entries before pruning. b) Revised table entries and edges after pruning.

the total number of connections that nodes j and j' has. The second column of the table represents the established capacity on edge (j, j'), i.e., $u_{jj'}$. This is shown in an example in Fig. 5.2. Fig. 5.2a is the resulting representation of a 5-node network after 5th iteration. $T[1]$ is the sum of connections of the peers of the corresponding edges whereas $T[2]$ represents capacities of the edges. The edge which has the highest in $T[1]$ and lowest in $T[2]$ is pruned. For instance, the 5th row in Fig. 5.2 shows that edge(3,5) has an edge capacity of 10, node 3 has 3 connections and node 5 has 3 connections. Hence, peers of edge(3,5) have 6 connections in total.

Thus, the edge(3,5) will be pruned. If there was another edge with the same total edge count, our algorithm would pick the edge with the minimum capacity. The idea behind aiming the edge with the lower capacity is avoiding unnecessary on-chain transactions for a channel that will not be used frequently as establishing a channel between two nodes will incur on-chain transaction fees. It is not desirable to have an edge for a small capacity, hence choosing the edge with the minimum capacity is important. And similarly, the nodes with more edges would have the potential to become hubs and thus create unfairness. Therefore, some of their edges are also pruned. Additionally, observing the total number of connections of the nodes ensures that we do not end with a disconnected network. Specifically, if a node has lower number of connections, it is highly probable that its edges will not be pruned. Fig. 5.2b shows the resulting representation after the 6th iteration.

This procedure is an iterative and greedy process. When an iteration ends, the network graph is updated with the pruned edge and procedure restarts, until topology becomes infeasible for the solver. We note that there are two reasons for a network to become infeasible, one reason is that the parameters are selected in such a way that it is impossible for the solver to come up with a flow solution. Another reason is that the network becomes disconnected after pruning so that flows cannot be forwarded to the intended destination.

5.5 Experimental Evaluation

This section presents the experimental details in terms of setup, assumptions, metrics, benchmarks and performance analysis.

5.5.1 Experiment Setup and Assumptions

The LP and MIOP optimizations were solved using Gurobi solver version 7.2 with Intel Xeon E5-2630 v4 @ 2.20GHz CPU and 64 GB of RAM PC running Ubuntu Linux Kernel version 4.15.0.

We make the following assumptions: for the optimization implementation, the available number of nodes may vary and it is explained in every figure and is denoted as N. For every node, there is one customer registered to it. Single customer per node is chosen in order to decrease the number of equations in flow optimization. Every customer sends transactions to every other node. This assumption is made in order to not end with a disconnected graph after pruning operations. Each transaction from each customer is 10 units. γ and $c^{f}_{jj'}(linkcost)$ parameters should be thought as adjustment parameters, they don't directly imply monetary values. $c^{v}_{jj'}$ in Eq. 3 is assumed to be 1.

5.5.2 Performance Metrics and Benchmarks

The evaluation of the performance of the heuristic approach is done according to the metrics below:

- *Total Investment Cost for the Network*: This metric is the amount of total investment put by the nodes (in escrow accounts) to create a network composed of nodes establishing off-chain payment channels.
- *Total Number of Edges in the Network*: This metric indicates the total number of channels (i.e., edges or links) created in the network.
- *Standard Deviation among the Nodes' Actual Investment Costs*: This metric is to assess the fairness among the nodes in the resultant network. It measures the standard deviation among the investment costs for each node in the network.
- *Computational Time*: This metric measures the computational time elapsed for finding out the resultant payment network.
- *Betweenness Centrality of the network*: Betweenness centrality is one of the measures that show how the nodes are dominant in a network. It simply defines how many times in total a node is visited while traversing from one node to other with the shortest path. This metric gives hints about a node in a network if it becomes a hub.
- *Percentage of Cut Nodes in the network*: Cut nodes set is the set of nodes that break the connectivity of the network when they are removed.

Using those metrics, we compare the performance of these approaches:

- **MIOP approach**: This solution is based on the MIOP formulation discussed previously. The initial topology network assumed to be a fully connected one.

- **Heuristic approach**: This approach involves the pruning of the edges. It is an iterative approach, and the flows are calculated by the LP model, and edges are pruned until an infeasible setup is reached. Note that for our heuristic there are two versions: one without privacy guarantees (shown as “Our Heuristic” in the figures) and one with the *privacy guarantees* (shown as “Privacy” in the figures). For privacy guarantee, we enforced all transactions make at least 3-hops. For this heuristic, we considered various initial topologies as follows:

  – *Fully connected Topology*: The heuristic starts with a fully connected topology where each node has a link to every other node in the network.

  – *Peer-to-Peer (P2P) Torus Topology*: The heuristic starts with a purely P2P topology that creates an equal number of edges among the nodes by following the idea of a torus. Torus topology is one of the most popular topologies in parallel computing [A+06] so we wanted to adapt it in our study. Fig. 5.3 shows an example 16 node torus topology.

  – *Barabasi-Albert (BA) Model Topology*: Currently, a restricted version of the Barabasi-Albert model is utilized in LN under `Autopilot` tool. In our tests, the same heuristic is applied to a network with an initial topology created by the Barabasi-Albert model.

  – *P2P Hypercube Topology*: Hypercube topology is a multidimensional network which has 2 nodes in each dimension. In every 2 dimension squares are generated. Due to this nature number of nodes in a hypercube is 2^m, m being an integer.

  – *P2P Star Topology*: In star topology all of the nodes are connected to a central node. This type of topology is very efficient in low latency multi-hop communication.

- **Random Topology**: In this approach, we pick a random topology which has a number of edges comparable to the results of the heuristic that starts with the fully connected network. Only the flows are calculated on this random topology without applying pruning.

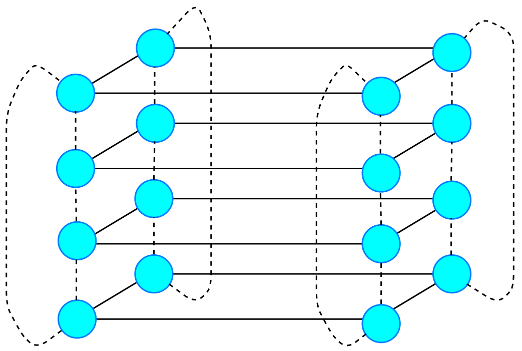


Figure 5.3: 16 node torus topology

5.5.3 Experiment Results

Comparison of Heuristic with MIOP Model Solution - Experiment 1

In this experiment series, we compared the performance of our heuristic starting with an initial fully connected network with the performance of the MIOP model solution [ECA+18].

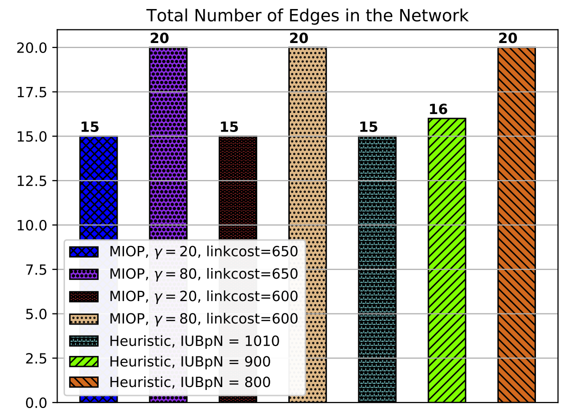
We utilized the same payment schedule used for the MIOP model solution and for the heuristic experiments. We created 10 retailers (nodes), a fully-connected network. We varied the $IUBpN$ for our heuristic experiments and varied γ and $c^f_{jj'}$ (shown as *linkcost* in the figures) for the MIOP experiments. Each retailer has 48 registered customers, so in total there are 480 customers. For the payment scenario, we assume that each customer initiates a single transaction to a retailer other than its registerer retailer which is selected randomly with uniform distribution. Each transaction has a monetary value of 10 units. Note that, in this setup, every node -via customers- supplies an equal amount of money to the network but demands of the nodes are not guaranteed to be equal. Hence, total supply to the network by all of the customers is of 4800 units. MIOP finds the optimum network topology based on this scenario and it is being controlled by the γ and *linkcost* parameters.

Our heuristic algorithm stops execution when the LP model becomes infeasible after iterative edge pruning. The LP model can become infeasible mainly for 2 reasons. The first reason for infeasibility is that the connectivity in the network can be broken because of excessive pruning. The second reason for the infeasibility is that the IUBpN can be tight that LP can not distribute the flow with the obtained connectivity of the network, hence a solution does not exist in that case.

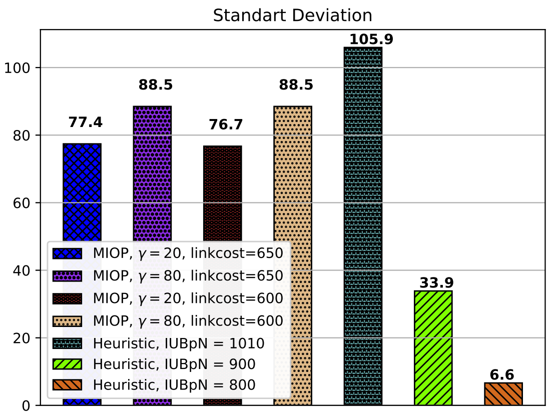
The results are given in Figures 5.4a, 5.4b, 5.4c, and 5.4d for total number of edges, standard deviation among the nodes' actual investments, total actual investment of the nodes in the network and total computational time to run the experiments respectively. In the experiments, we tried to adjust IUBpN in a way that the results would be comparable in terms of number of edges.

It is seen that the MIOP model gives only slightly better results in terms of total investment. However, this comes with a drawback of huge computational time requirement. Because of the previously mentioned complexity of the MIOP model, the required running time increases significantly.

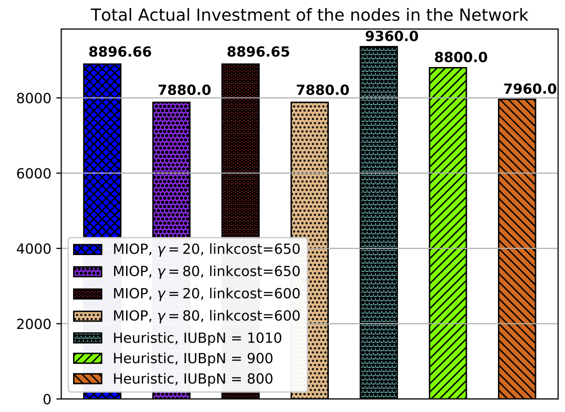
There are several other observations: First of all, the heuristic produces less number of edges compared to the MIOP solution when IUBpN increases. Because with a lower IUBpN the heuristic terminates earlier, so that, it leaves a network with a higher number of connections. Hence, the heuristic can be adjusted for savings from channel creations.



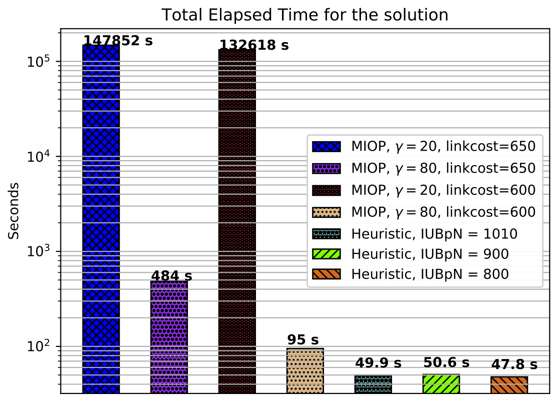
(a) Number Of Edges



(b) Standard Deviation



(c) Total Investment



(d) Total Run Time

Figure 5.4: Heuristic approach compared to MIOP solution

Standard deviation results are interesting since we observe the high impact of increasing the IUBpN. However, when the right IUBpN is selected, such as 800 in our case, the flows become much fairly distributed among the nodes and surpasses the MIOP (optimal) solution. This behavior makes sense because after IUBpN is made tighter, the node capacities are kept the highest possible to fulfill the constraint, yielding a close outbound flow among nodes. Similarly, when we look at the total actual investment cost in the network, we see that the heuristic almost matches that of the MIOP model. For instance, compared to the $\gamma = 20$ case, there is only about 3% cost increase and yet the number of edges is equal to the total number of edges for the same case which is matching the heuristic. This means the heuristic can get very comparable results in terms of standard deviation and total actual investment costs while achieving scalability. Also, the IUBpN parameter works efficiently to have control over the standard deviation among the nodes.

Comparison of Full Connected Topology with others: Torus topology, Barabasi-Albert Model Topology, and Random Topology - Experiment 2

In these experiments, our goal is to show the scalability properties of our heuristic by trying it on a P2P Torus topology, on a topology created by Barabasi-Albert (BA) method and also compare the results with a randomly connected network. To this end, we created topologies of sizes 25, 30, and 35 nodes. The topology created by BA model has 4 edges per node on average. We then compared the performance of our heuristic starting with a fully connected network to that of a Torus network, and to that of a BA model network. We also created a random network topology with the same number of nodes and same "final" number of edges, so there is no heuristic applied to random topology, only the flows are calculated with the same parameters.

Figure 5.5a shows the resulting total number of edges in the networks. Figure 5.5b shows the total computational time for the experiments. Figure 5.5c shows the total actual investment of the nodes. Figure 5.5d shows the results of the standard deviations among the investment costs of the nodes. Legend for the figures is tabulated in Table 5.3.

Our first observation after starting the experiments was that Torus and Barabasi-Albert model topologies were becoming infeasible in much earlier iterations when the same IUBpN is applied compared to the fully connected network. For instance, heuristic for a fully connected network could be applied up to an acceptable number

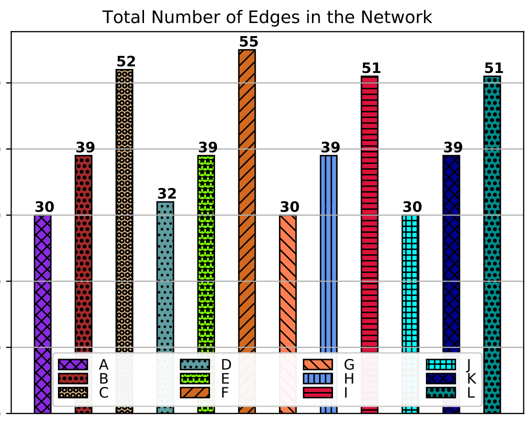
Table 5.3: Legend for Figure 5.5

Abbreviation	Legend
A	25 Nodes, 1200 IUBpN, full connected
B	30 Nodes, 1200 IUBpN, full connected
C	35 Nodes, 1200 IUBpN, full connected
D	25 Nodes, 1200 IUBpN, torus
E	30 Nodes, 1200 IUBpN, torus
F	35 Nodes, 1200 IUBpN, torus
G	25 Nodes, 1200 IUBpN, BA Model
H	30 Nodes, 1200 IUBpN, BA Model
I	35 Nodes, 1200 IUBpN, BA Model
J	25 Nodes, 10000 IUBpN, Random
K	30 Nodes, 10000 IUBpN, Random
L	35 Nodes, 10000 IUBpN, Random

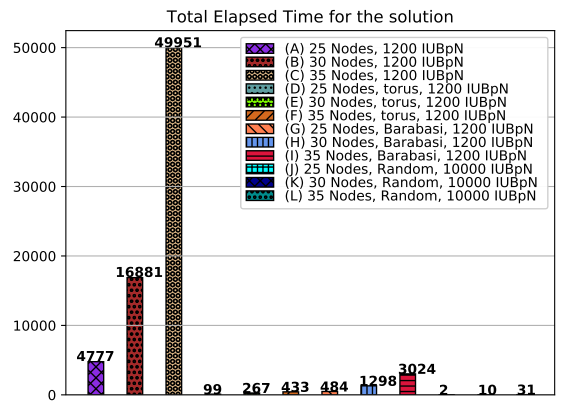
of iterations when $IUBpN = 800$ but the others could not be solved, namely, with those parameters and flow requirements, the networks seem infeasible to the solver. Keeping that in mind, we decided to go with an increased IUBpN value for all cases which is set to 1200 in the experiments. The random connected graph was the worst among all in feasibility, so, for getting a result we had to set IUBpN 10000 for the random topology.

Since the heuristic steps take place in an iterative fashion, Torus and the BA model topologies have the advantage of starting from a pre-pruned state with much less number of edges. This property is well seen in total time comparisons (Figure 5.5b). Additionally, with lower number of edges, the LP model deals with less number of equations which makes the equations solvable in much less time. For the random topology, as the network has less number of edges and no further pruning is done, the solver only calculates flows which is much faster compared to others. These results suggest that Torus and BA can scale much better.

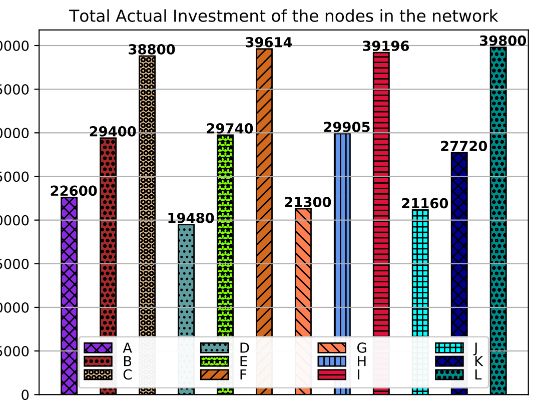
An interesting observation from the results is that both BA generated topologies and Torus topologies can perform as good as a fully connected network for the total number of edges and total actual investment. For instance, for 35 nodes



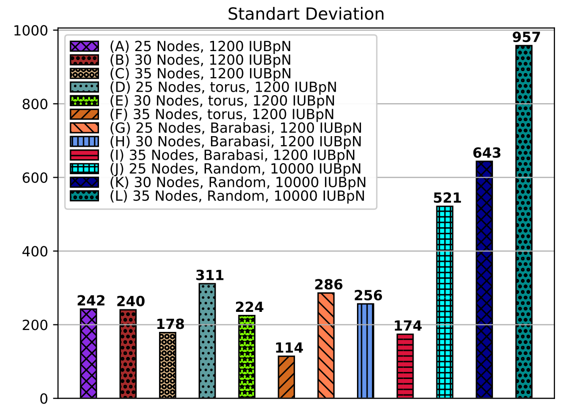
(a) Number Of Edges



(b) Total Run Time (s)



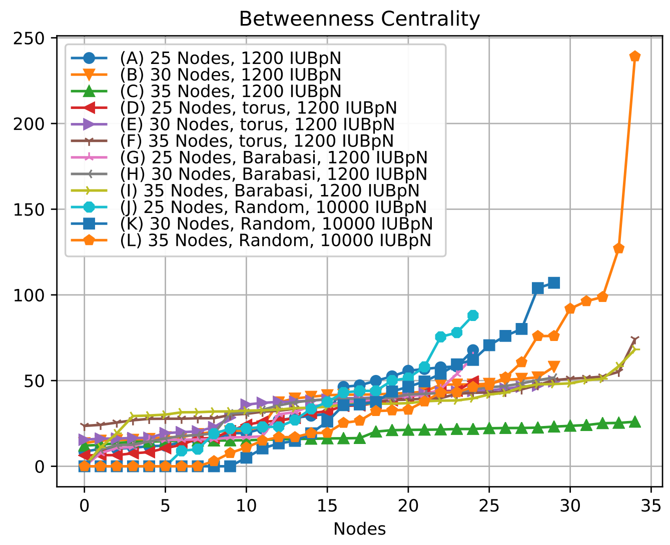
(c) Total Investment



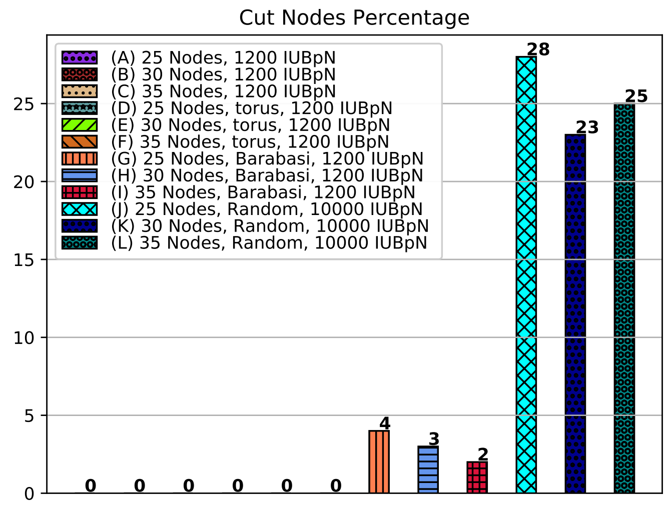
(d) Nodes' Standard Deviation

Figure 5.5: Comparison of results related to heuristic for Full Connected network, Barabasi-Albert model, Torus Topologies vs. Random Topologies

case, it is interesting to observe that although Torus has more edges than the fully connected topology, the total capacity of Torus network exceeds that of the fully connected one (Figure 5.5c). In general, a network with a higher number of edges should have lower total capacity. This is because, if the number of edges decreases, the transactions tend to follow paths with more number of hops. High number of hops incurs additional investment requirement for other nodes, in turn, increasing the total network capacity. Because of the strict connection in Torus, the flexibility of finding shorter paths decreases resulting in higher investments.



(a) Betweenness Centrality



(b) Percentage of cut nodes

Figure 5.6: Betweenness centrality and Cut Node Results of full connected, Torus, BA model and random topologies after applying heuristic

The standard deviations of nodes' actual investments in Torus and BA are comparable to a fully connected network as seen in Fig. 5.5d. For the random network, however, the standard deviations are high since it has a very high IUBpN parameter. This hints that some of the nodes have higher loads, resulting in topologies more similar to a hub-and-spoke. The betweenness centralities of the networks shown in Fig. 5.6a support this finding as random topologies have the highest scores.

On top of statistical and computational calculations made on edges and nodes, visiting the topological properties of the networks gives us a better perspective on advantages and disadvantages of resulting topologies. Fig. 5.6a shows the betweenness centrality of the networks and Fig. 5.6b shows the percentage ratio of cut nodes (articulation points) to all of the nodes in the network.

With total capacity, total number of edges and total investment costs of the topologies being close to each other, heuristic applied to the initially fully connected network gives the best results in terms of betweenness. Percentage of cut nodes shown in Fig. 5.6b shows that randomly connected network is very weak against attacks as there are many nodes to attack which will highly impact the operational success. BA model has similar vulnerabilities but significantly lower than that of the randomly connected network. Fully connected and Torus networks seem to be more robust.

Consequently, we can conclude that given the much more efficient computational time for Torus, it can be a more viable option in real-life as long as the right IUBpN is selected. Additionally, if BA model is created with a higher average number of edges, the results may come closer to the fully connected network results. Because there will be more room to fine tune the final topology of the network, but, with an increase in computation time.

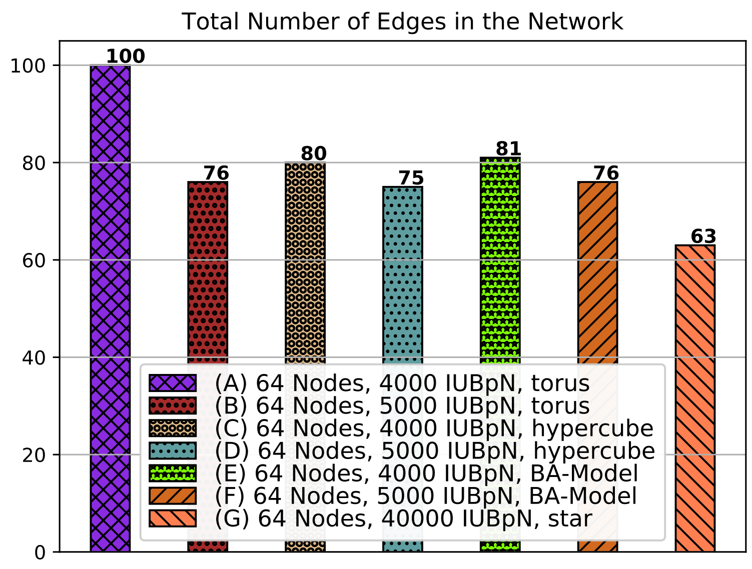
Comparison of Others: Torus, Hypercube, BA, and Star topologies - Experiment 3

In this experiment, we investigate how our approaches perform compared to other related topologies. We conducted experiments to with an initially torus connected network, hypercube connected network, BA model connected network, and star connected network topologies. As hypercube being combinations of squares in ndimensional space, for a hypercube topology, number of nodes in the network is 2^n where n is an integer. Hence for this experiment the number of nodes in the networks is selected to be 64. Figures 5.7a, 5.7b, 5.8a, and 5.8b show total number of edges, standard deviation, total investment cost, and total time spent in the final topologies.

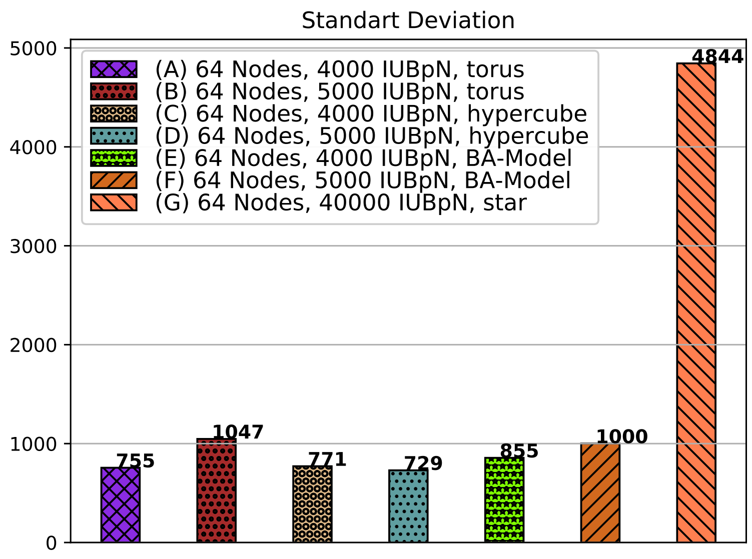
In star topology the central node has to forward all of the payment requests to corresponding nodes so $IUBpN$ for central node is extremely high, 40,000. Otherwise, with a lower $IUBpN$ the solver can not optimize. Torus, hypercube and BA-model topologies give comparable results in terms of total capacity and standard deviation for the same $IUBpN$s. Hypercube topology starts with a higher number of nodes compared to Torus. That causes more edges to be pruned and a higher total time for the method to finish for the same $IUBpN$. Additionally, initial higher number of edges means more equations and that makes the solver to take longer times to calculate the flows. For the torus' 4000 $IUBpN$ case it can be said that because torus has a larger diameter, (8) than the hypercube (6), the average number of hops is higher. Thus, for a small $IUBpN$ in torus, the solver will stop earlier because average load on the nodes increases. That explains why the solver stopped when the number of edges was 100. Although hypercube topology gives comparable results to the torus topology, for the remaining experiment we will continue with the torus network since it is much more efficient in terms of total computation time and scales better.

Scalability Experiments

In order to further evaluate the scalability property of our heuristic, we made several experiments with networks of 100 nodes. As a benchmark, we compared the Torus and BA model networks with different IUBpN values. Figures 5.9a, 5.9b, 5.10a



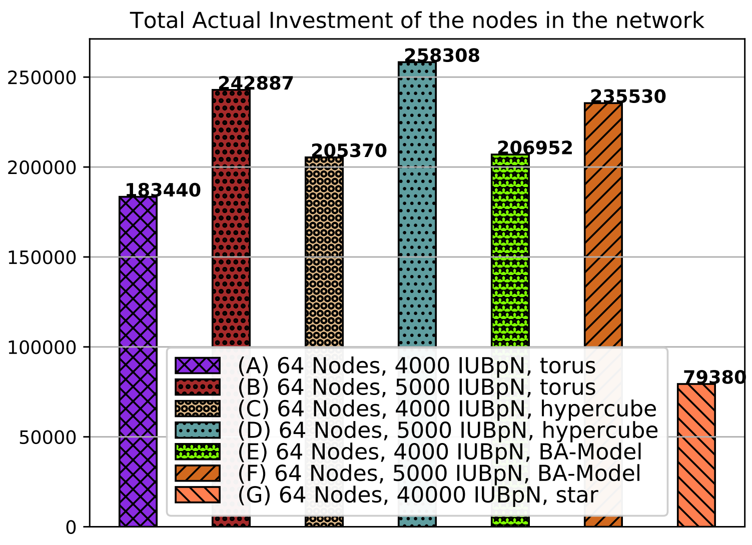
(a) Number Of Edges



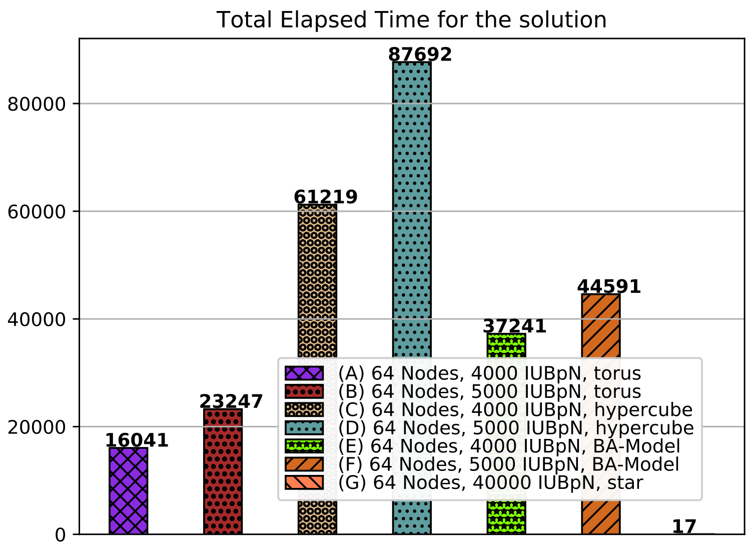
(b) Standard Deviation among nodes

Figure 5.7: Comparison of Torus, BA model, Hypercube and Star topologies. Number of edges and standard deviation.

and 5.10b depicts the total number of edges, standard deviation, total investment cost and total time in the final topologies. As discussed in the previous sections, the more ordinate connection arrangement in the Torus topology results in a higher number of edges and higher investment cost but a better control on the standard



(a) Total Investment



(b) Total Run Time (s)

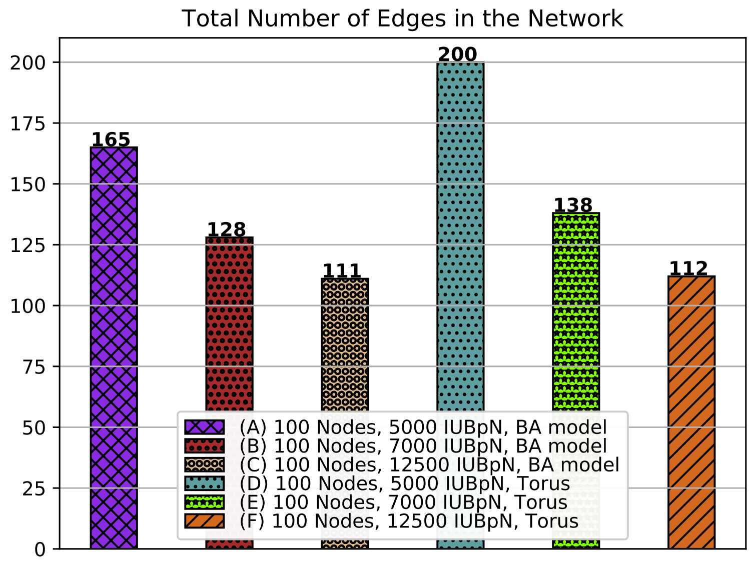
Figure 5.8: Comparison of Torus, BA model, Hypercube and Star topologies. Total network capacity and total time.

deviation among node capacities. However, in BA model topology there is more room to find shorter paths yielding in lower number of hops and consequently lower investment cost throughout the network although the control on the standard deviation is harder. For the very same reason, Torus topologies terminate sooner than the BA model based topologies. Consequently, the computational time required in the Torus topology experiments is significantly lower than that of the BA model topologies. Considering the final number of edges, final total capacity, and comparable standard deviation control BA model gave better solutions compared to Torus. However, when computation time is important Torus topology has an undeniable advantage.

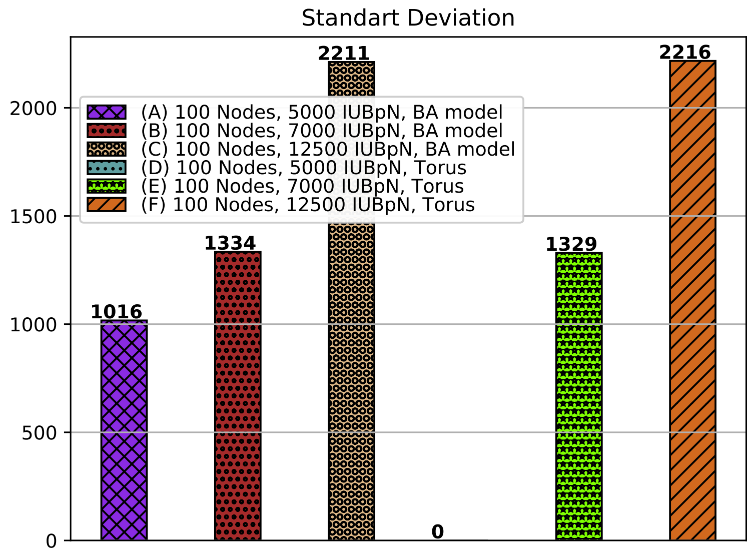
5.6 Conclusion

In this chapter, we designed a private payment network from scratch for Bitcoin by exploiting LN technology for a marketplace where retailers and customers are available. In developing this payment network, we achieved several objectives: First, we eliminated the high transaction fees and confirmation times by using the off-chain concept of LN. Second, we ensure forming a connected payment network which is capable of transferring any payments between customers and retailers while establishing fairness between cooperating retailers to share the associated costs. Finally, we both reduced the success of DDoS attacks on the network and the possibility of privacy leaks by creating a pure P2P topology. The development included an optimization model for flow maximization while performing pruning for the edges in order to reduce the number of channels to be opened.

The evaluation with Python and Gurobi indicated that the performance of the heuristic approach is very close to the MIOP solution while allowing a certain scalability. The results also suggested that the topologies generated by Barabasi-Albert came as a favorable initial topology that can scale much faster and still provides comparable results as long as the upper bound for the channel capacities are picked rightly.

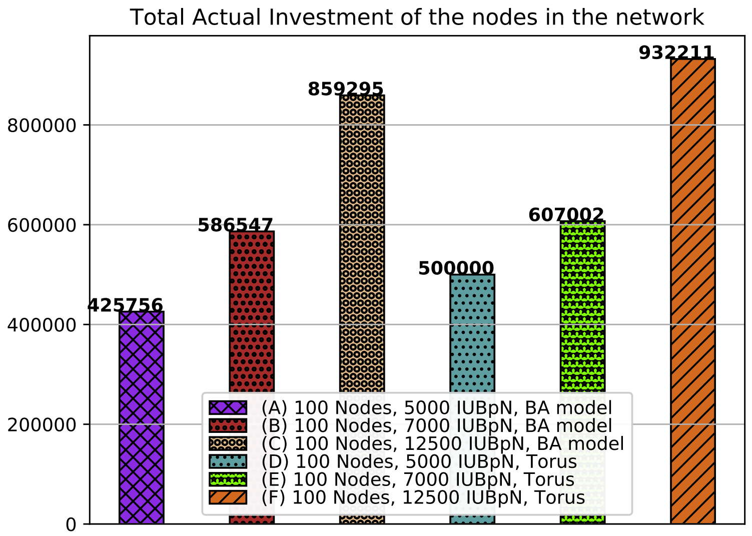


(a) Number Of Edges

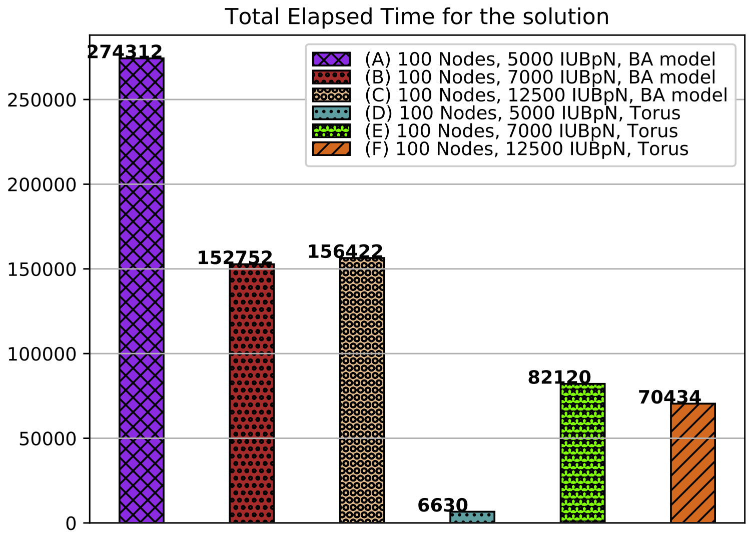


(b) Standard Deviation among nodes

Figure 5.9: Further scalability tests for Torus and BA model generated topologies. Number of edges and standard deviation.



(a) Total Investment



(b) Total Run Time (s)

Figure 5.10: Further scalability tests for Torus and BA model generated topologies. Total network capacity and total time.

CHAPTER 6

A SCALABLE PAYMENT NETWORK FORMATION HEURISTIC USING DIJKSTRA'S SHORTEST PATH ALGORITHM

In Chapter 5 we have developed a sub-optimal network formation model in order to further scale the model obtained in Chapter 4. Although it gives comparable results, its scalability performance still lacks as scaling beyond 100 nodes seems impractical. LN currently serves with more than 10 thousand nodes and in its well established form, a few hundred thousand nodes are thought to be participating. Hence in this chapter, we propose a network-formation heuristic which will form a network topology by relying on the transaction intents between nodes using a shortest path algorithm. In the network we are trying to form, nodes start to transfer money to each other, weights (or interchangeably referred to as costs) on the edges will be updated so that the shortest path formations can be influenced in such a way that existing channels are favored to a certain extent. There are three components in the weight of an edge, namely, link-establishment cost, transaction cost, and the new channel forcing cost. When all of the transactions are completed, we obtain a final topology by creating off-chain links on the used paths. We consider several criteria while initializing and changing the weights of the edges that will enable a highly decentralized topology (i.e., a pure peer-to-peer (P2P) topology). We also aim to achieve a certain standard deviation among the outbound flows of the nodes within a certain channel capacity to ensure fairness.

The evaluations using *Python* and *Gurobi solver* indicate that our proposed heuristics can provide comparable performance to that of the optimal solution while allowing scalability and fairness. We also achieve 3-hops payments with similar topology features with a slight increase in the computational time.

6.1 Proposed Heuristic Algorithm

In this section, we describe our proposed heuristic in more detail.

6.1.1 Approach Overview

Our heuristic of PCN formation is based on the idea of in-advance planning of payments and flows. As every retailer has an idea of their business capacity and expectation, we use it to plan payment flows among the customers and retailers. We first start distributing the flows in advance from various retailers to others in the best way we can (i.e., fair load and P2P distribution) assuming that there are already available channels among them at the beginning. We then look at the final used channels among retailers, set up the actual off-chain links and remove any other channels that are never used in the iterations of our heuristic approach.

In this heuristic, finding the path between a source and a destination retailer is crucial. When we look at today's LN, if there is a path between the payer and the payee, the payer can use that path if it is convenient to use, meaning, if there is enough capacity on the planned path. Otherwise, the other alternative is to establish a direct channel with the payee. However, in that case, there will be on-chain transaction fees for opening and closing channels. Therefore, one needs to weigh these two options when finding a path.

We follow a similar rationale for our heuristic. Specifically, during the iterations in the heuristic, for a particular payment request a convenient path is searched. If there exists a path from one retailer to another one, our heuristic uses that path by relying on a shortest path algorithm, namely Dijkstra's shortest path algorithm. If there is none, new channels are opened suitably. Additionally, if the total cost on the path starts to create inconveniences for intermediary nodes (i.e., adding a burden for the retailer), then we force our approach to open a new channel by adjusting the edge weights in the Dijkstra's shortest path algorithm. Namely, we strive to find a sub-optimal approach for opening channels so that the participants of the network neither suffer from unfair load distribution nor pay excessive on-chain transaction fees. Next, we describe our heuristic details.

6.1.2 Finding Paths

In order to find the best possible routes, Dijkstra's shortest path algorithm is used [Dij59]. In Dijkstra's algorithm, the path with the lowest total weight is found between a source and a destination node. In our case, we have an a priori payment list. From the payment list, transactions are read one by one. At each reading, meaning iteration, a shortest (i.e., lowest weight) path from the source to the destination is found. After a path is found, the weights on the edges are updated according to the flow (i.e., payment amount) which will be detailed in the next subsection. The algorithm is shown in Algorithm 2 which utilizes the notation in Table 6.1.

Note that in this Algorithm the payments are picked in a round-robin fashion from the payment list, P (i.e., finish a particular retailer's payments and move on to the next) which may greatly influence the resultant topology as we followed a certain order. This may create unfair load distributions and undesirable topologies. In order to come up with a topology in which the loads are more evenly distributed, the randomly selected customers execute their transactions in a random round-robin fashion. Specifically, in order to minimize the impact of dependence on the order, at each round, the order of the retailers is renewed with a new distribution. This is achieved with Algorithm 3.

Table 6.1: Notations and their explanations

Symbol	Meaning
H	Directed Graph
H_e	Edge e in graph H
L_c	Link establishment cost
W_i	New connection forcing cost
γ	Parameter to control unfairness between the nodes
T_a	Transaction amount
T_{AB}	Transaction amount on edge (A, B)
$H_e.weight$	Weight of edge e
$H_e.flow$	Flow on edge e
U_{AB}	Binary var. represents existence of flow on edge (A, B)
E	Initial number of edges in the experiments

Algorithm 2 Network Establishment

```
1: Input: $P$=$Payment\ List$, $H$=$fully\ connected\ directed\ graph$, $L_c$=Link establishment cost, $W_i$=New connection forcing cost
2: **for** every edge, $e$, in $H$ **do**
3:     $H_e.weight$=$W_i$+$L_c$
4:     $H_e.flow$=0
5: **end for**// *Initial assignments are done*
6: **for** every *payment* in $P$ **do** // *A payment is defined by a source, a destination and the transfer amount* $T_a$
7:     $Path$=$ShortestPath$($H$, from=$a$, to=$b$)
8:     **for** *each edge*, $e$, in $Path$ **do**
9:         $H_e.flow$ +=$T_a$
10:        $H_e.weight$=$W_i$+$H_e.flow$
11:    **end for**
12: **end for**
13: **for** all edges in $H$ **do**
14:     **if** $H_e.flow$=0 **then**
15:         Remove edge from $H$
16:     **end if**
17: **end for**
18: Output: $H$
```

According to this approach, first, two nodes are selected randomly, one of which is the source, a, and the other is the destination, b. If there is an intended transaction

Algorithm 3 List Establishment

```
1: Input: S=Set of Retailers
2: while All required payments are not fulfilled do
3:     TempS=S
4:     while TempS is not Empty do
5:         Pick 2 random retailers (a,b) from TempS
6:         if Transaction from $a$ to $b$ was not fulfilled then
7:             Add $a$ as source, $b$ as target in $P$
8:             Remove (a,b) from TempS
9:         end if
10:    end while
11: end while
12: Output: P=Payment List
```

from a to b, and if it is not fulfilled yet, a transaction from a to b with the transaction amount is added to a payment list, P. Afterward, a, b pair is removed from the list of retailers. This removal is important because we want every node to be visited equally either as a source or as a destination. Whenever every retailer visit is complete, meaning the list of retailers is an empty set, the procedure is repeated. This new random list of payments is then fed to Algorithm 2.

6.1.3 Defining Edge Weights

As mentioned, after finding a path the edge weights need to be updated to inject the desired influence to topology formation. To achieve this, we define a sophisticated **weight function**, on an edge, e.g., the weight between A and B, W_{AB}. Specifically, three components of the W_{AB} are defined: *link establishment cost*, *transaction cost*, and *new connection forcing cost.* Below, we explain them next in more detail.

Link Establishment Cost (L_c): In LN, establishing a channel means doing at least two on-chain transactions on Bitcoin blockchain, which incur on-chain transaction fees. Hence, opening many channels will be costly for a node. Furthermore, if all of the nodes open many channels, the cumulative fee paid by the network participants will be discouraging for the consortium. For example, for a fully connected mesh network of N nodes, there will be $N \times (N-1)/2$ edges. With increasing N, the total fee paid by the network participants will be tremendously high. Instead of full connection in the network, a lower number of edges will be more acceptable as it lowers the total on-chain transaction fee. The edges should be reused cleverly to distribute the transactions among nodes in an acceptable way so that a flood in the number of channels in the network will be prevented.

In order to encourage the reuse of the edges rather than inserting new channels in the network, a parameter called $LinkCost$, denoted as L_c is introduced. L_c mainly relates to the on-chain transaction fee. In the proposed heuristic, all edges in the network have a non-negative L_c set to some value. Whenever an edge is used (i.e., there is a flow on the edge), the L_c on that edge is nullified (set to 0 to indicate that the channel is already open). So further transactions can use that edge on their paths. Nullifying the L_c encourages other transactions in such a way that other transactions will prefer low-cost edges instead of opening a new one.

Transaction Cost: In the algorithm, When an edge is used (or channel is established), L_c will be nullified, that is, a channel will be opened, and thus all later transactions will tend to use that channel since other yet-never-used channels will have a higher weight due to higher L_c. As some edges will have lower weight due to nullified L_c, these channels will be used for the transaction flows. This high usage of the channel contradicts with the aim of establishing a flat network to execute all of the transactions, because, using a channel for a transaction requires enough collateral in that channel. In a financial system, ideally, the users will not want to put a lot of money aside just to spend a little of it. So again, putting a lot of money is discouraging. In addition to that, if a cost is not assigned to the unit flow, we will see that frequently used channels will need more collateral than others. That will be extremely unfair if all of the participants expect the same benefit from the network. For that reason, whenever there is a transaction through an edge, the amount transferred incurs a weight on that edge which is basically a *transaction cost* induced by channel usage. So, when there are edges with heavier loads (high transaction costs), the transactions will start to look for new routes or open new channels. This helps to distribute the loads more evenly. Hence the weight, W_{AB}, is revised as follows to accommodate this transaction cost:

$$W_{AB} = L_c(1 - U_{AB}) + \sum T_{AB} \tag{6.1}$$

where U_{AB} is a binary variable and equals to 1 if there happens to be a flow on edge AB anytime during the procedure, and 0 otherwise, and T_{AB} is the amount of all transactions (from all nodes in the network) passing on edge (A, B).

New Connection Forcing Cost: In some cases, when the links are established, during the algorithm run, the future transactions in the list tend to use those links which will increase the investment need to be made by intermediate nodes for maintaining these links. In such cases, we need an additional force to further increase these links' weights so that the Dijkstra's algorithm will not choose these links anymore.

As an example consider an initially fully connected mesh network topology shown in Fig. 6.1(a) where all of the edge weights are initialized accordingly, with $L_c = 500$. If we look at the established links after the first run of payments, we see that half of the nodes initiate transactions to the remaining half of the nodes randomly in one hop as shown in Fig. 6.1(b), and the effect of L_c is nullified and the weights are updated with the flows on the edges. Note that, for simplicity, not all of the $L_c = 500$ weights are shown in the figures. In the second round of transactions, new randomly selected half of the nodes will initiate new transactions to other nodes.

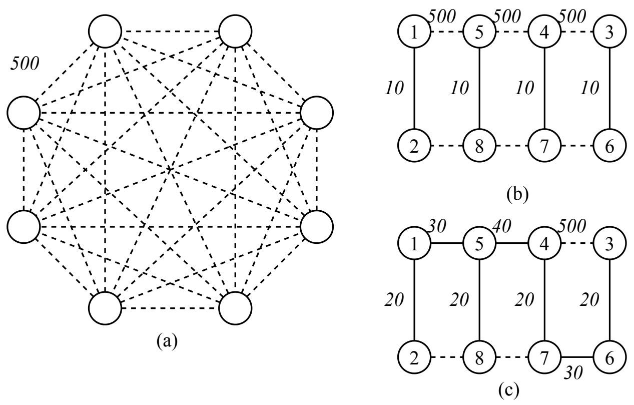


Figure 6.1: Initially fully connected mesh network topology.

This will form new connections as shown in Fig. 6.1(c) but still the opened links will be in use. However, now unused edges in the topology will still have a higher weight of L_c (500 in this example), while other edges will have the weights only created by the transaction amounts. In the later rounds, no matter how random the nodes are picked, all of the transactions will follow the already established edges since they will have lower weights due to their L_c being set to 0. Thus, the topology will continue to be as shown in Fig. 6.1(c), resulting in no significant change. For a larger N, we will observe longer paths in the network and the topology will stay unchanged although more transactions are added. In order to prevent transactions traversing always through the same paths, we introduce a constant weight, W_i, for each edge to increase the total weight on the path and thus surpass the edges with L_c ((i.e., forcing new connections). In this way, the longer paths will be hampered. Thus, the weight on edge (A, B) will be updated as follows:

$$W_{AB} = L_c(1 - U_{AB}) + \sum T_{AB} + W_i \tag{6.2}$$

The effect of W_i can be better seen with an example shown in Fig. 6.2. In this sample experiment L_c is set to 3000, and the number of nodes, N, is 20. The payment lists are the same for both of the resulting topologies. Basically, without

W_i, we will get a topology in Fig. 6.2(a). Introducing W_i and setting it to 1000 causes the topology to change to the one in Fig. 6.2(b). We argue that Fig. 6.2(a) is not a desired topology because it is weak against node failures, and some nodes are highly centralized.

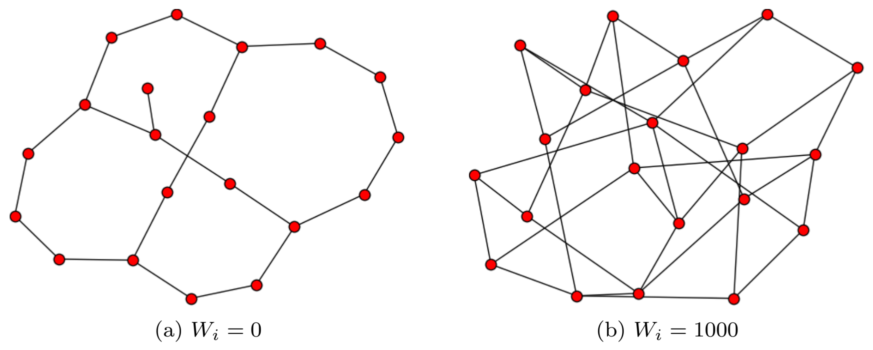


Figure 6.2: Effect of W_i for a network of 20 nodes, $L_c = 3000$

6.2 Evaluation

In this section, we analyze relationship anonymity, describe the experiment setup, performance metrics, and discuss the evaluation results.

6.2.1 Experimental Setup and Implementation

N nodes (retailers) are assumed in the network. A single customer is assumed to be attached to a single node and it will create 10 unit worth transactions to every other node. So, the supply from a single customer to the network is $(N - 1) \times 10$. Total money traversing in the network is $N \times (N - 1) \times 10$. In LN channel formation, the peers can independently decide on the amount they want to put in the channel. However, for the completeness of the study, we assume that peers of a channel put the same amount in the channel they created. The proposed approach is implemented in Python and its performance is assessed extensively through various experiments.

All the experiments are carried out on a computer with an Intel Xeon E5-2630 v4 @ 2.20GHz CPU and 64 GB of RAM.

6.2.2 Metrics and Benchmarks

The results of the experiments are assessed based on the following metrics:

- ***Betweenness Centrality of nodes:*** Betweenness centrality of a node in a network is a measurement showing how many times a node is visited while traveling between other nodes using the shortest path traversal. In a hub-and-spoke network model, hubs will have the highest betweenness score.
- ***Total Capacity of the Network:*** This metric shows the total amount of investment to be put by the vendors to the channels for the formation of the network.
- ***Number of Edges:*** This metric shows the number of edges established in the resultant topology.
- ***Standard deviation among the nodes:*** This metric shows the standard deviation among the outbound flows of the nodes. A high standard deviation hints that some of the nodes are used more like a relay compared to the other nodes. A zero standard deviation means all of the loads on the nodes are equal.
- ***Total Computation time:*** This metric is the measure to show how long it takes, in seconds, to finish all necessary computations for the final results.
- ***Utilization:*** This metric is the ratio of the total flow in the network to the total capacity of the network. It is calculated by dividing the sum of all transactions to sum of all established capacity in the network.

- ***Histogram of Number of Hops:*** This metric shows the histogram of the transactions in terms of the number of hops they follow calculated in percentage.
- ***Cut Nodes:*** Cut nodes are the nodes whose removal entirely makes the network disconnected. The higher the better for a topology since this means more nodes need to be removed/failed to disconnect the network.

We compared our approach against certain benchmarks and methods as listed below:

- **MIOP model**: The results of the heuristic are compared with an optimization model in [ECA$^+$18].
- **Random network topology**: The results of the heuristic are also compared with the results of a randomly connected network. The heuristic is run on the random network to get the flows in the network.

6.2.3 Experiment Results and Discussion

Comparison of Heuristic with the MIOP model

In this section, the results of the proposed heuristic approach are presented and compared with that of the MIOP model studied in [ECA$^+$18]. The objective is to assess our approach's performance with respect to the ideal one. The optimization model was solved by Gurobi Solver. However, in the setup of this experiment, only 10 nodes are used since the MIOP model does not scale beyond 10 and thus in practice is not usable. Only for this experiment, different than the general scenario assumption, we assumed that these 10 nodes are serving to 80 customers which are distributed to these nodes randomly. Each customer sends money to 6 different nodes and each is of a value of 10 units. Hence, the total supply by the customers to the network is 4800 units. From the experiment results of the MIOP model, best ones are used in regards to betweenness centrality, standard deviation and number of edges. For the results of the heuristic approach, the same scenario is inherited. All the related results are shown in Figures 6.3, 6.4, 6.5, 6.6, 6.7, and 6.8. In those figures, γ is a control parameter for the unfairness among node outbound flows, and *linkcost* is the link establishment cost, L_c in MIOP.

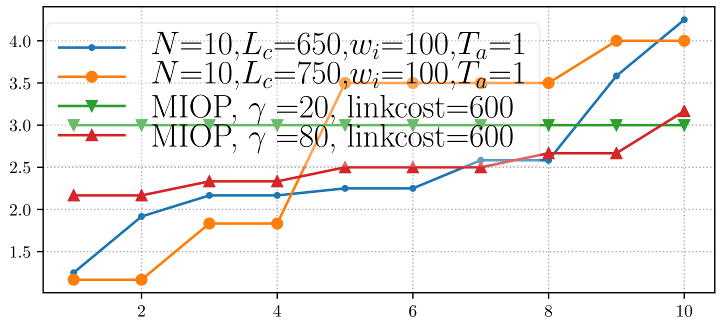


Figure 6.3: Optimal vs. Heuristic Comparisons - Betweenness Centrality.

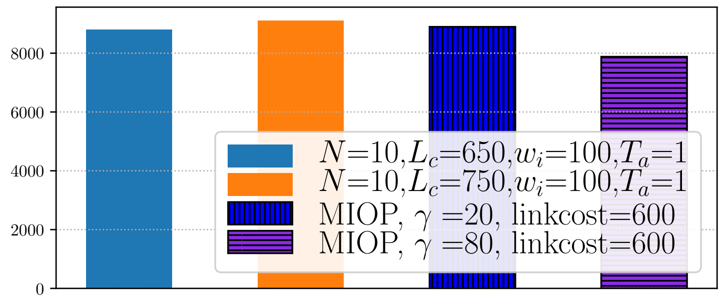


Figure 6.4: Optimal vs. Heuristic Comparisons - Network capacity.

As can be seen from Fig. 6.4 and 6.5, our heuristic's performance almost matches the performance of the MIOP solution in terms of total capacity and edges. It is also only 20% short of the utilization of MIOP (Fig. 6.6). For the standard deviation metric, as MIOP has a significant control on unfairness, the standard deviation

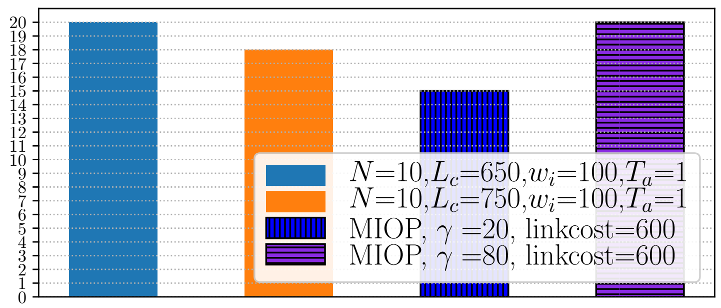


Figure 6.5: Optimal vs. Heuristic Comparisons - Total edges.

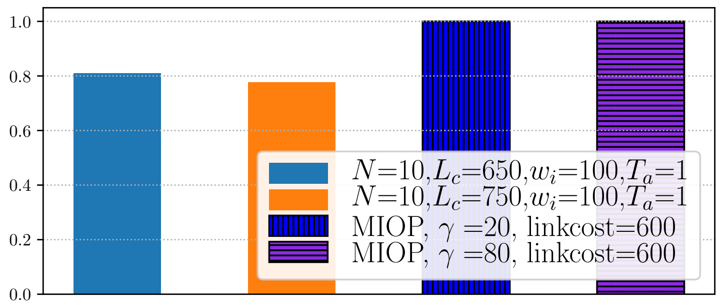


Figure 6.6: Optimal vs. Heuristic Comparisons - Utilization.

in MIOP solutions is lower than that of the heuristic approach as seen in Fig. 6.7. However, when W_i is 100 and L_c is 650 in the heuristic approach, standard deviation comes to a more comparable level, where the number of edges has a significant effect on this. This is because as the number of edges increases, the flows tend to be distributed more evenly since the flows can find shorter routes compared to a network with a lower number of edges. Finally, compared to MIOP solutions the betweenness centrality for our approach in Fig. 6.3 is slightly increasing but still maintains a topology close to P2P.

The obvious advantage of our approach is computational overhead. It reduces the computational time 100 to thousands folds (i.e., it scales much better) while still getting very close to the MIOP's overall performance (Fig. 6.8). In summary, the proposed approach provides the same features as MIOP in a much faster/scalable manner but with some slight deviation from an ideal P2P topology.

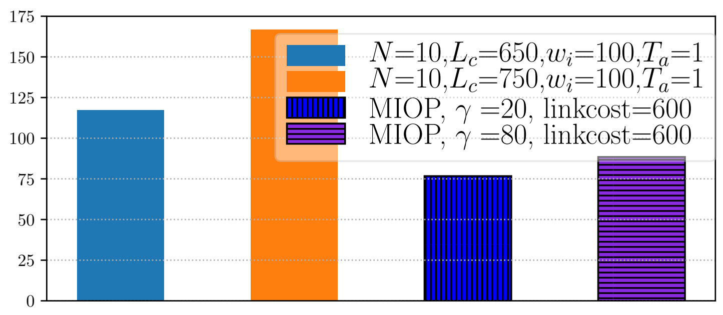


Figure 6.7: Optimal vs. Heuristic Comparisons - Standard Deviation.

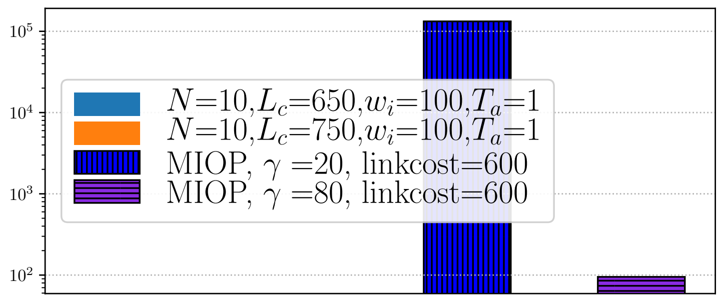


Figure 6.8: Optimal vs. Heuristic Comparisons - Total Time (s).

Ideal Parameter Selection for the Heuristic

Apparently, picking different parameters highly affects the resulting network topology for the heuristic approach. In this section, we conducted a series of experiments to determine the ideal parameters for our heuristic to run. The experiments are evaluated for different L_c and W_i cases and a fixed number of nodes, $N = 100$, which yields traversal of 99000 units of money in the network, with an exact amount of 990 units per node. The results are shown in Figures 6.9, 6.10, 6.11, 6.12, 6.13, and 6.14.

Considering all of the different parameters visited in the course of this experiment, with the payment scenario assumption and under 100 nodes, we obtain a good topology when L_c is 4000 and W_i is 700. We call the topology good because, the standard deviation is around 600, with an average load per node around 3000. The total number of edges in the network is close to 300 implying on average every node has 6 connections. Additionally, the maximum number of hops does not exceed 6 and resides around 3. These parameters are used in the remaining experiments.

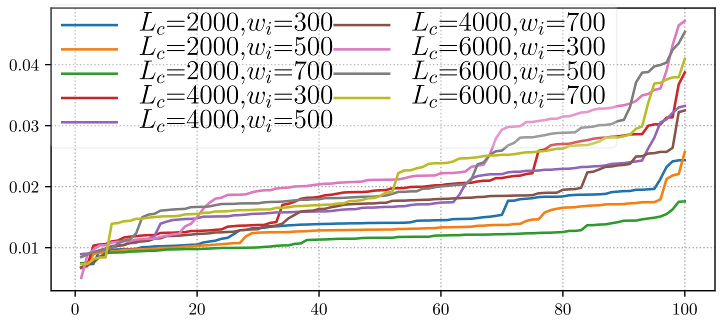


Figure 6.9: Ideal Parameter Selection - Betweenness Centrality

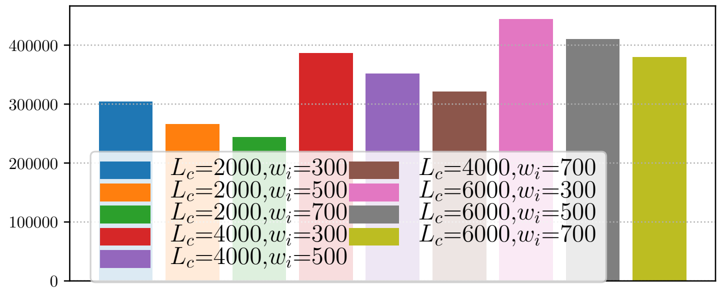


Figure 6.10: Ideal Parameter Selection - Total Flow Capacity of the Network

The Impact of T_a on the Performance

During the calculations in the previous experiments, the transfer amount from one source to a destination was picked to be 10, and it was atomic. In this experiment, we want to observe whether breaking up a transaction amount into smaller chunks will change the behavior of the network formation. The transaction, 10 units, is divided into smaller chunks (i.e. T_a) and at every iteration, in the algorithm one of those chunks is assumed to be transferred. In order to see the effect of the T_a parameter, in the light of the results obtained in the previous experiments, L_c and W_i are fixed to 4000 and 700 respectively.

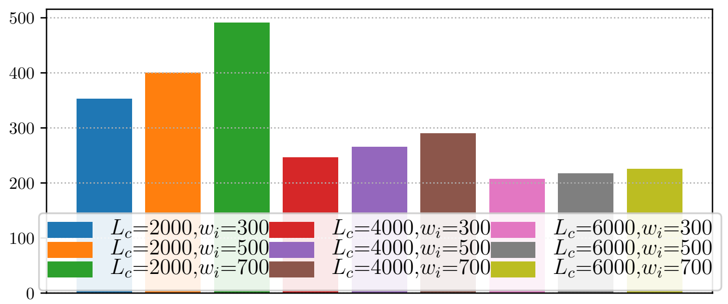


Figure 6.11: Ideal Parameter Selection - Total Edges

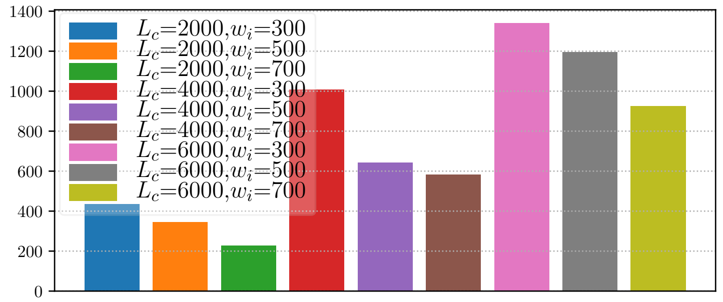


Figure 6.12: Ideal Parameter Selection - Standard Deviation

The results are shown in Figures 6.15, 6.16, 6.17, 6.18, 6.19, and 6.20. As can be seen, there is not a dramatic improvement in terms of the total number of edges, hop percentage, and total network capacity. However, partitioning the flow into smaller units create a significant negative impact on the total time needed for the procedure to end. From the standard deviation results, we observe that $T_a = 5$ and lower values give a lower/better result. With the liberty of sending a transaction through different paths, we can deduce that dividing payment into chunks improve the fairness as loads will be distributed to many different nodes. Due to the same reason, the betweenness centrality of the network seems to be slightly getting better too. The number of edges and the capacity are not changing significantly. We deduce that the exploration of the paths is heavily dependent on L_c and W_i. T_a shows its impact only on even distribution of the flows.

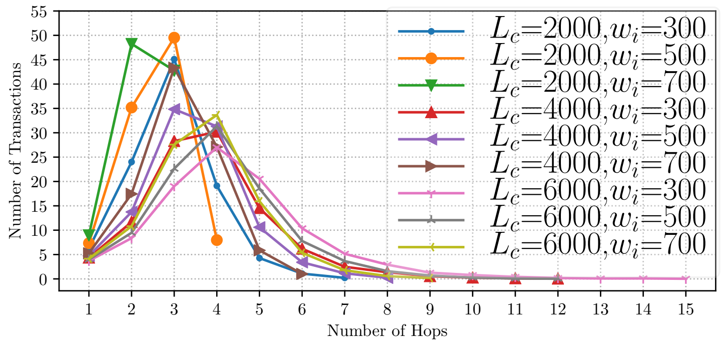


Figure 6.13: Ideal Parameter Selection - Histogram of the number of hops for transactions (Normalized- percentage)

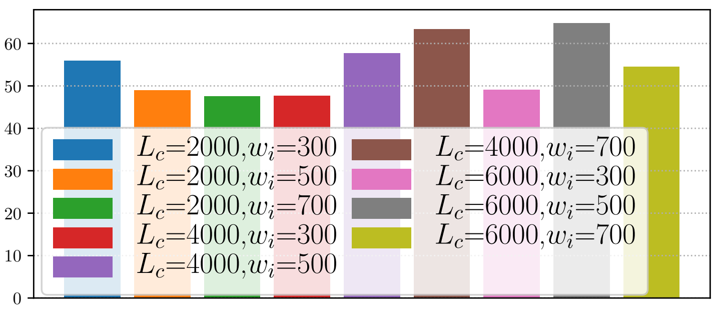


Figure 6.14: Ideal Parameter Selection - Total Time (s)

Scalability of the Heuristic

In this experiment, we assessed the scalability features of the proposed heuristic. Specifically, the heuristic approach is run with different numbers of nodes, namely 250 and 500 nodes. However, as the number of nodes increases, the computation time required to finish the calculations increases drastically due to the time complexity

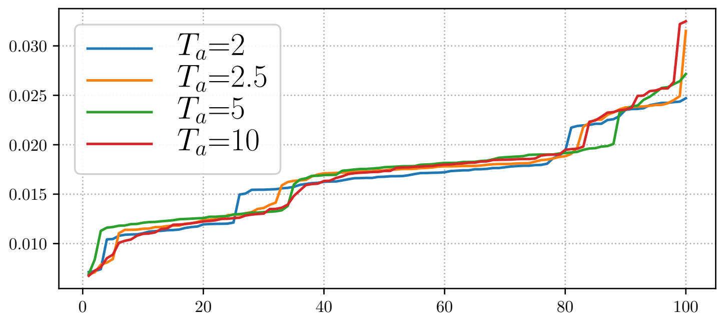


Figure 6.15: Effect of T_a - Betweennes Centrality

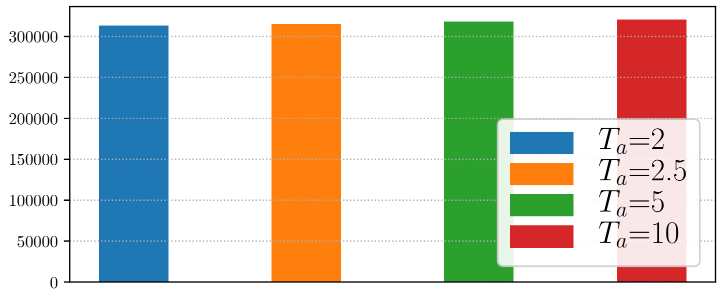


Figure 6.16: Effect of T_a - Network Capacity

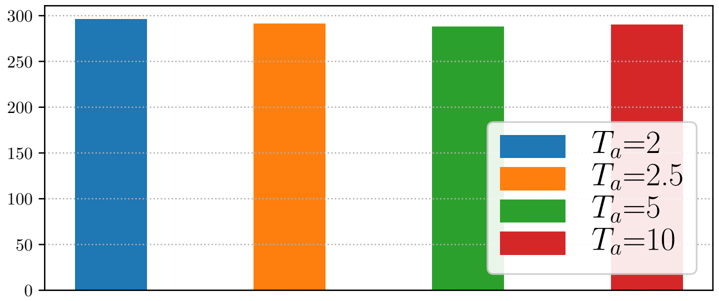


Figure 6.17: Effect of T_a - Total number of Edges

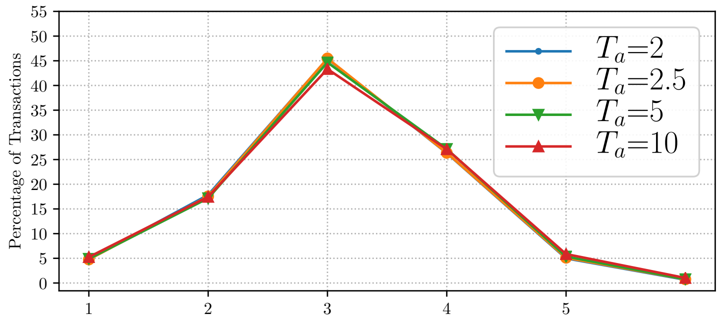


Figure 6.18: Effect of T_a - Histogram

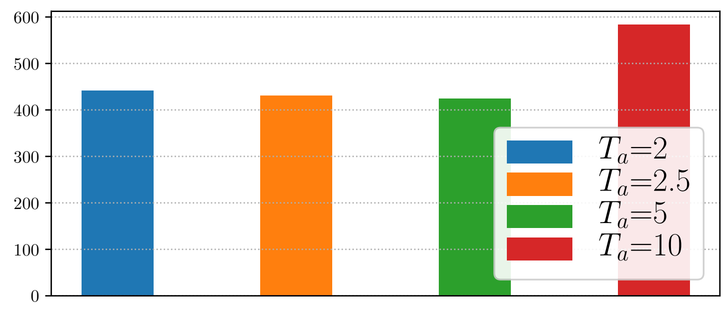


Figure 6.19: Effect of T_a - Standard Deviation

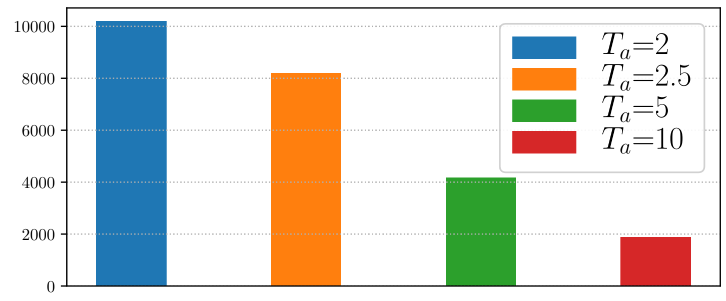


Figure 6.20: Effect of T_a - Total time (s)

of Dijkstra algorithm which is, if implemented in simple form, $O(|E|log|N|)$, where E is the number of edges and N is the number of nodes. Since our heuristic starts with the assumption that all nodes are connected to each other, the number of edges becomes $E = N^2$. So the time complexity of the heuristic translates into $O(N^2 logN)$. In order to decrease the effect of the assumption of an initially fully connected mesh network, we also created networks with random initial connections as an alternative approach for comparison. To differentiate these two, in the figures, the initial number of edges are depicted with the E parameter where $E = All$ indicates our approach with a fully connected network.

The results are shown in Figures 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, and 6.27. Pre-pruned topologies (i.e., randomly connected) give an advantage in terms of total computation time, as expected, especially with 500 nodes. However, other results are generally slightly better for the initially fully connected mesh network setup.

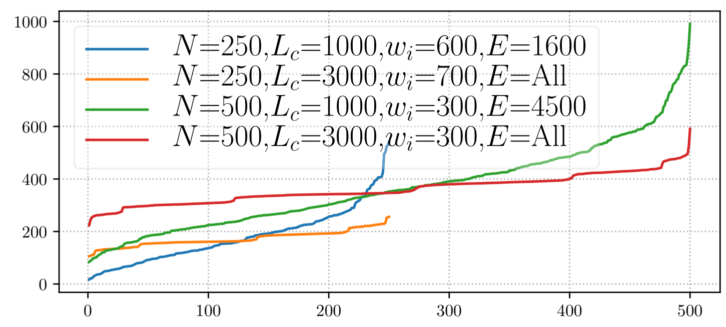


Figure 6.21: Scalability Test Results - Betweenness Centrality

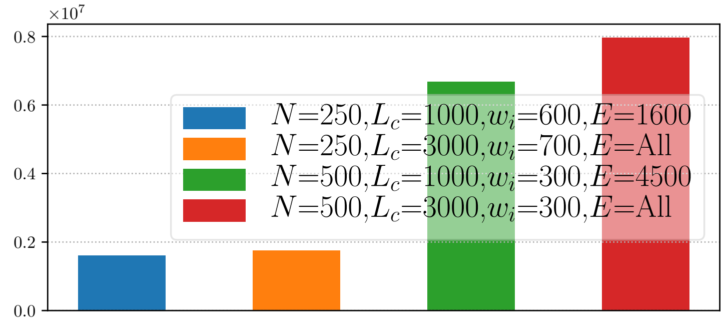


Figure 6.22: Scalability Test Results - Network Capacity

In particular, standard deviation of our approach with a fully connected topology is significantly reduced. Additionally, based on the results, we argue that making random connections may not degrade the total investment capacity in the network but comes with unfairness among the nodes as standard deviation among nodes changes too much.

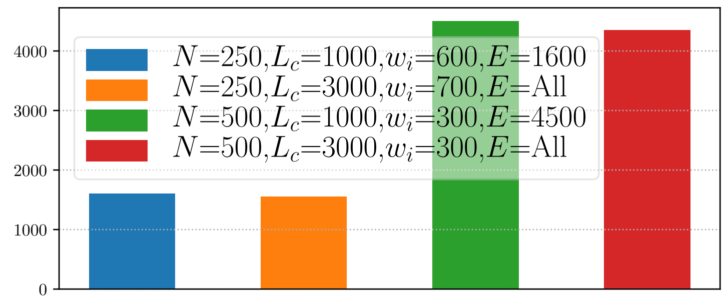


Figure 6.23: Scalability Test Results - Edges

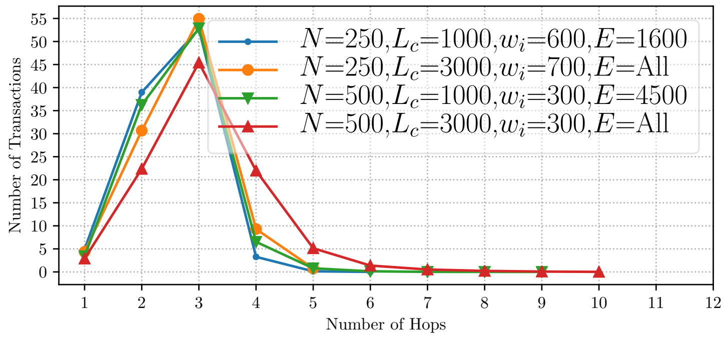


Figure 6.24: Scalability Test Results - Histogram

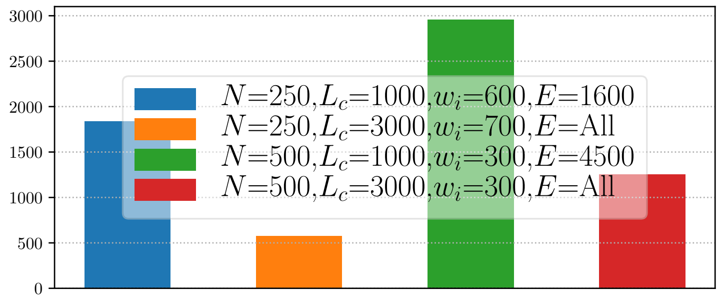


Figure 6.25: Scalability Test Results - Standard Deviation

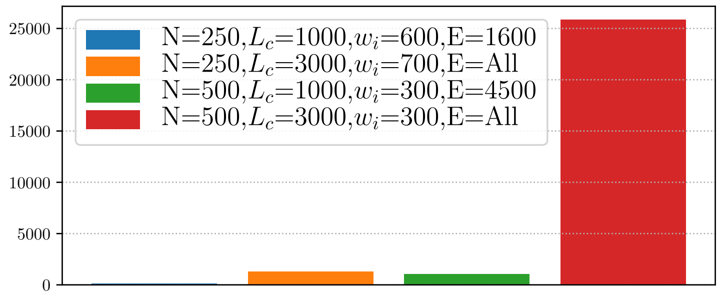


Figure 6.26: Scalability Test Results - Total Time (s)

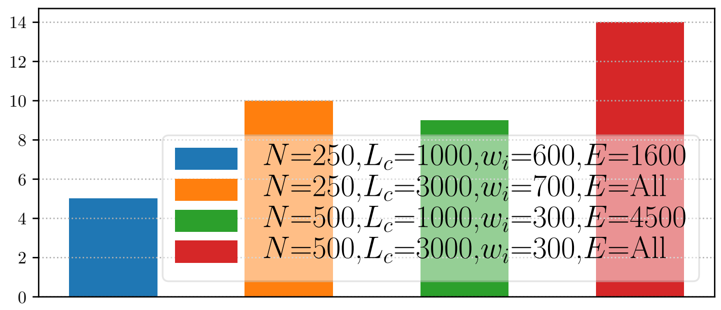


Figure 6.27: Scalability Test Results - Number of cut nodes

As part of this experiment, we also looked at the number of cut nodes. Fig. 6.27 represents the results of the cut nodes for different parameters. As can be seen, when the network is randomly connected, we observe a lower number of cut nodes compared to our heuristic topology. That means, for a randomly connected network, the possibility of taking down the network is easier because attacking fewer nodes will be enough. This is not the case in our heuristic with the fully connected mesh network as its betweenness centrality is more stable and thus more nodes need to be taken down in order to disconnect the network. As the network size doubles, this number also increases linearly indicating that our heuristic maintains a similar behavior as new nodes are added. This is one of the main strengths of our approach in terms of producing a good topology against DDoS attacks. We can conclude that up to a certain number fully connected topology might be a better choice. However, when we move beyond a certain number of nodes, for time savings, pre-pruned topologies may be preferred based on their cut node performance.

6.3 Conclusion

Cryptocurrency based payment channel networks using the idea of off-chain payments has have been emerging recently. This is not only because they reduce confirmation times but they also let users send micro-payments in a very affordable way. Therefore, forming a reliable and scalable P2P payment network is an open question assuming a private consortium of retailers (nodes). In this study, based on some scenarios and assumptions, we developed a heuristic approach to form such a payment network topology using Bitcoin's off-chain concept and Dijkstra's shortest path routing and compared the results with the results of an optimal solution.

Compared to the optimal solution, the heuristic reduces the computational time significantly. Additionally, the fair distribution of the load among nodes, centrality measures and the total number of edges obtained in the networks are satisfying to ensure a truly P2P network topology features.

CHAPTER 7

PRIVACY IN PAYMENT CHANNEL NETWORKS

In Chapters 4, 5, and 6 we have developed 3 different methods for a PCN formation. In this chapter we investigate ways to extend these works for privacy consideration. This chapter defines attack models, privacy requirements in payment channel networks, and finally compares our approach to the state-of-the-art PCNs.

In Bitcoin, privacy plays a vital role. Although, attacks related to de-anonymization in Bitcoin is out of scope in this dissertation, by utilizing pseudonyms, the real identities of the users were aimed to be kept private. It is seen that, inherited from this philosophy, designers pay attention to privacy features in the design of PCNs at various levels. Nevertheless, strengthening security of the proposal comes with weaker privacy or strengthening privacy makes the network less practical. In this chapter, we seek answers for the following question; if PCNs are used for daily transactions, from web purchases to retailing, what kind of privacy do they provide, and to what extent?

In its simplest form the definition of *data privacy* or *information privacy* is defining how storing and disclosing of the collected data take place. For centrally managed systems the central node (or company) is the responsible party for keep-

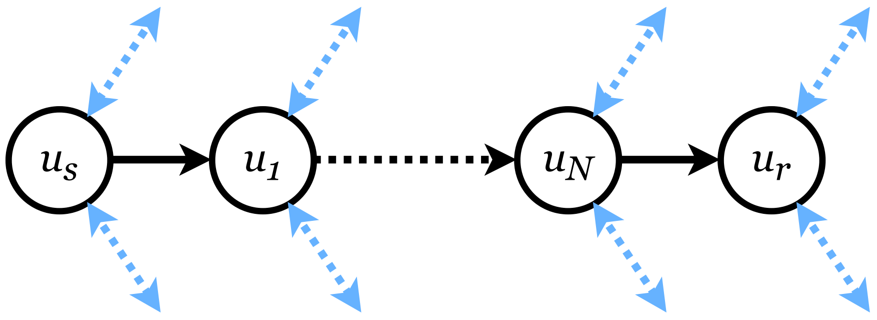


Figure 7.1: For a typical example the sender u_s initiates a transaction to u_r through the intermediary nodes $u_{1 \ldots N}$. A node can open as many channels as she wishes. The channels are assumed to be bidirectional.

ing the privacy of the users. However, when the system shifts towards a decentralized/distributed fashion, the privacy of the users should be taken care by the network protocol. A payment in a PCN is assumed to take place as shown in Fig. 7.1 where the sender, u_s, initiates a transaction destined to the recipient, u_r. Her payment passes through other nodes $u_{1...N}$ where $N \geq 1$. As we will see in the coming sections, an intermediary node either can be defined as a normal node like others or it may be equipped with special functionalities, e.g., a node can be a landmark which decides on possible routes for a payment. A node may optionally have established extra channels (shown with blue dashed arrows). Unless otherwise stated a payment channel is assumed to be bidirectional.

In the state-of-the-art papers, the attacker models and their capabilities vary. Some studies aim to hide the sender (u_s) identity whereas some concentrate on strengthening the relationship anonymity. Throughout this chapter we will consider different kinds of attacks based on the privacy metrics. Nonetheless, privacy notion is totally depends on needs of the users of the network. Before proceeding to the attack model, we want to state our assumptions about the attacker. The attacker is clever, attacker fully or partially knows the topology of the network, and the information the attacker knows does not decrease but may increase in time. That is, with every action taking place in the network attacker may learn from it and she does not forget them.

There are two types of attackers considered. The first attacker is the honest but curious (HBC) type. HBC attacker acts honestly while running the protocol. However, at the end of an operation the attacker knows more than what she knew before. The second attacker of interest is the malicious attacker. The malicious attacker controls more than one node in the network to deviate the protocol. Hence, she can act based on her own rules, e.g. denial of service or colluding with other

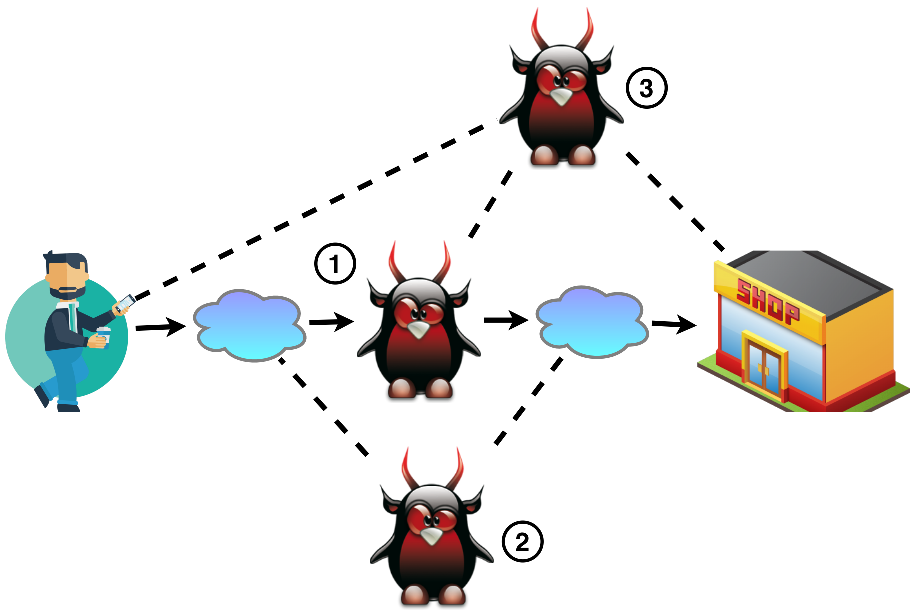


Figure 7.2: Attackers can appear in the network in different places. ① The attacker is on the path of a payment. ② The attacker is not on the path of a particular payment but she can partially observe the changes in the network. ③ The attacker colludes with the other nodes.

nodes in order to infer user/payment information. Attacker types and how they can appear in the network are shown in Fig. 7.2. ① The attacker is on the path of a payment. She already knows the atomic payment amount. ② The attacker is not on the path of a particular payment but she can partially observe the changes in the network. ③ The attacker colludes with other nodes, for example, to make packet timing analysis. This attacker needs sophisticated methods. Attacks on privacy will eventually result in attacks on the security (e.g., denial of service attacks, censorship, hijacking) of the network but security attacks are out of scope within the context of this study.

- **Sender (Payer) Anonymity**: The identity (or pseudonym) of u_s should not be known to the others after a payment finishes. In the extreme case even u_r may not know the identity of the user. However, there may be cases where an adversary may successfully guess the pseudonym or the real identity of the sender. For example, for case ① in Fig. 7.2 the sender can have only single connection to the network and u_1 can be the attacker, hence, attacker is sure that u_s is the sender. For case ② the attacker may guess the sender by observing the changes in the channel balances in the network. For case ③ the attacker will learn the sender if she can carry out a payment timing analysis within the partial network formed by the colluded nodes.

- **Recipient (Payee) Anonymity**: The definitions and examples for sender anonymity and possible attacks hold for recipient anonymity as well.
- **Channel balance privacy.** To protect the investment information of a node, the channel capacities should be kept private. Moreover, if the capacity fluctuations on the channels are known, following the start and end of changes cause indirect privacy leakages about the payers and payees.
- **Relationship Anonymity.** In some cases identities of either u_s or u_r may be known to the others, this is a very valid case for retailers because they have to advertise their identities to receive payments. However, the privacy of the trade can be satisfied by hiding the information of who sends a payment to whom. It may be known that a payment is already destined to a retailer, but, keeping the sender identity unidentifiable helps protecting privacy. If the attacker can relate the payer and the payee to each other, not only the spending habits of the sender but also the income pattern and the business model of the recipient will be known to the attacker. Hence, to satisfy the relationship anonymity either the recipient or the sender anonymity should be satisfied.
- **Business Volume Privacy.** For a retailer, publicly known income will yield the trade secrets of her business. In that sense privacy of each of the payments is important. In a scenario where two or more nodes collude (③), or an

attacker controls more than one node in the network, the amount of a transaction can be known to the attacker. In another scenario, if the recipient is connected to the network via a single channel (①) and u_N is the attacker, she will be able to track all of the flows towards the recipient.

7.1 Privacy in Lightning Network

LN, as in Bitcoin network, operates with the pseudonym addresses. Although the addresses are pseudonyms, there are still privacy concerns and solutions are being introduced. First of all, when the nodes join LN, with a public advertisement, they advertise some properties of the channels they establish. The IP address is one of those properties. The nodes need to know the IP addresses of other nodes in order to route the payments. The LN community offers using Tor Network in order to further anonymize the identities. The second one is the channel capacity. For a successful routing, the sender needs to know the capacities of the channels that the payment will be routed through so that the transaction can be successful. However, in LN in order to keep the directional capacities of the nodes private, the channel capacity of a channel is expressed as sum of all fundings provided by the peers. Third is the possible privacy leakage in the transactions, namely, analyzing the source node, destination node, and the amount being transferred. In LN, source-routing is utilized which makes a payer node to decide on the route of transaction. Source-routing idea comes with additional extensions where onion based routing is utilized in addition to a mix-net approach inspired from `Sphinx` protocol. With these properties, in a multi-hop payment, an intermediary node only knows the predecessor and the successor nodes.

LN, instead of connecting retailers and customers directly, relies on *relay nodes* which act as bridges between parties. Eventually, these relay nodes become powerful hubs which forward most of the payments and hence may charge high forwarding fees. Obviously, this is against the very idea of reducing the transaction fees for creating such a payment network. Furthermore, allowing the relay nodes to become monopolies in forwarding poses vulnerabilities for denial of service (DoS) attacks and privacy analysis of customers' transactions. For instance, there was a recent DDoS attack on LN which took 20% of the nodes down and gave hard times to the transactions [Tru18]. In addition to these issues with the structure, there are also issues with routing of payments. Any payment made in the current LN has a chance of less than 1% to make it to its destination if the transaction amount is greater than $200 [DIA18]. This would obviously be a big problem for a retailer that needs to collect payments without having any issues on time.

7.2 Privacy Implications of MIOP Solution

In Chapter 4 we have focused on overall cost reduction as our key optimization objective along with multiple cost-sharing scenarios, our constructions and security definitions do not directly aim to ensure privacy.

However, the resultant PCN indirectly ensures privacy by forming a decentralized topology. This is particularly relevant since the existing studies that focus on routing privacy in PCN require PCN to be designed in a decentralized manner [RMSKG17, MMSK+17]. Lightning Network offers source routing mechanism together with Tor-like onion payment forwarding. The routing mechanism provides a strong user privacy by not letting intermediary nodes to find out source and destination of the payment. Nevertheless, this will not be possible in a hub-and-spoke model since the payments will pass through the same set of nodes. This will only be possible in a fully decentralized/distributed topology as provided by our optimization model.

7.2.1 Privacy Analysis of MIOP Solution

Our study is complementary to other studies which provide privacy preserving routing. We argue that our network formation helps to achieve the privacy goals as discussed below:

Payment Privacy: We assert that payment privacy is preserved if an adversary who is not on the payment path can not determine the payment value. The adversary can not determine payment value without compromising all the intermediary nodes since the point-to-point communications are encrypted in tor-like routing. Moreover, our PCN formation allows to perform payments in fractions over different paths. Thus, in case even some of the intermediary nodes on the path are compromised, the privacy of the whole payment value can be preserved due to the availability of alternative paths.

Customer Privacy: The privacy of a customer is preserved if an adversary can not determine customer identity while forwarding the payment. If the adversary is placed on an intermediary node other than the customer's initial connection node, the adversary can not identify the customer since the payment was forwarded by a previous node. If adversary compromised the first node (which forwarded the payment) or the receiving node, the privacy is still protected. Note that, in public PCNs, a customer participates in the network using its pseudonym Bitcoin address which does not carry any information about the identity of the customer. The privacy might be deteriorated by only performing a long-term analysis along with other side information such as payment amount which is encrypted (i.e., first node). However, the customer can change its address by creating another payment channel when its payment channel is exhausted or expired.

Retailer Privacy: The retailer privacy is protected in the same manner as the customer privacy.

7.3 Privacy Implications of Linear Programming Heuristic

This section discusses how privacy considerations can be embedded to the solution offered in Chapter 5.

7.3.1 Incorporation of Privacy in the Heuristic

In order to increase the privacy on the flow of transactions in the network, the best practice is forcing the transactions to make multiple hops. For doing so, we can introduce binary decision variables which will be set to one if there is a positive $y_{jj'}^{ik}$ flow on an edge. So, putting a constraint on these binary decision variables, such that sum of them will be greater than some number will successfully force the flow to make hops. However, introducing binary decision variables will push the flow optimization model from linear polynomial time domain to NP-complete domain as explained previously.

To avoind this situation, we opted for another approach for satisfying privacy by introducing Eq. 7.1 as a constraint in the flow optimization model:

$$\sum_{j \in P} \sum_{j' \in P} y_{jj'}^{ik} \geq Y^{ik} \qquad \forall i, k \in J_i \tag{7.1}$$

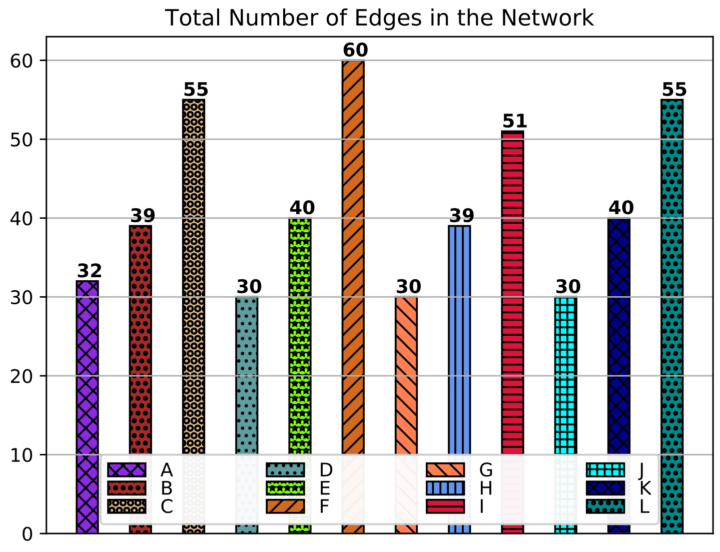
where Y^{ik} is a lower bound for satisfying privacy. Eq. 7.1 states that, for a particular source and destination, the sum of all of the flows carried out in the network particularly for these peers should be larger than a lower bound. For example, if Y^{ik} is set to be 3 times of a_{ik} the flow optimization solver will need to find additional routes in order to satisfy this constraint.

7.3.2 Privacy-Guaranteed Heuristic Experiments

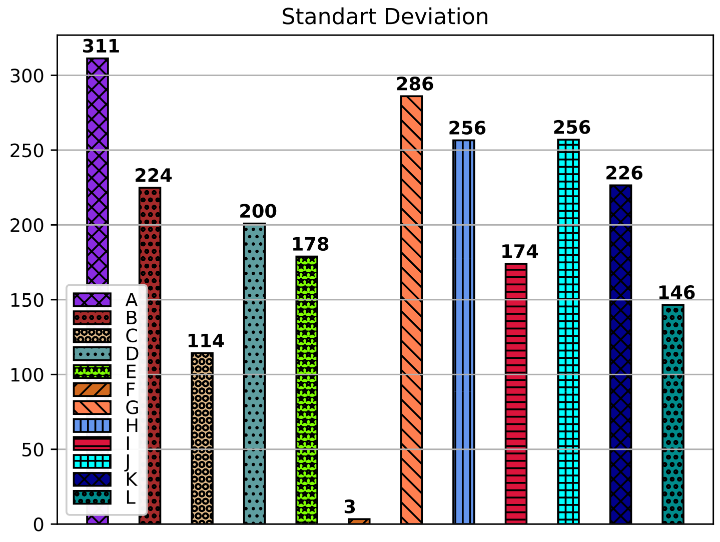
In these experiments, we looked at the impact of guaranteeing privacy by revising our heuristic and compared it with the case when privacy is not necessarily guaranteed (i.e., normal heuristic). The initial topologies for the experiments are chosen as Torus and BA model generated topologies as they provide a close performance to a fully connected network. For the privacy case, we set the constraint (i.e. Y^{ik}) in Eq. 7.1 as 30 units. Since a transaction flow is worth 10 units, we would like to observe 3 hops for a successful transaction in an ideal case. For this particular experiment, the number of nodes in the networks is set to be 25, 30, and 35. The IUBpN in the experiments is fixed and is 1200. As applied before, stopping criteria for the algorithm is reaching to an unfeasible flow distribution solution.

Figures 7.3a, 7.3b, 7.4a, 7.4b show the resulting final number of edges, the standard deviation among the final actual investment of the nodes, final total actual investment in the network and the total computational time for the experiments, respectively where the legend for the figures are is in Table 7.1.

One of the observations in these experiments was that when the stress on the solver is increased by increasing the limit in Eq. 7.1, the solver starts to create



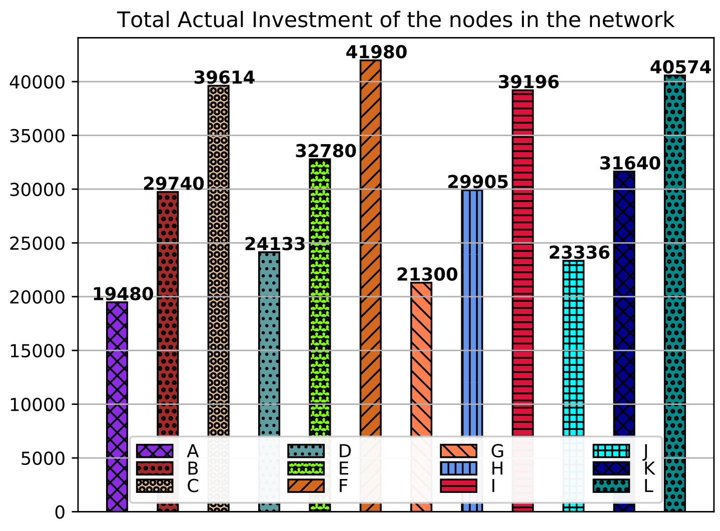
(a) Number Of Edges



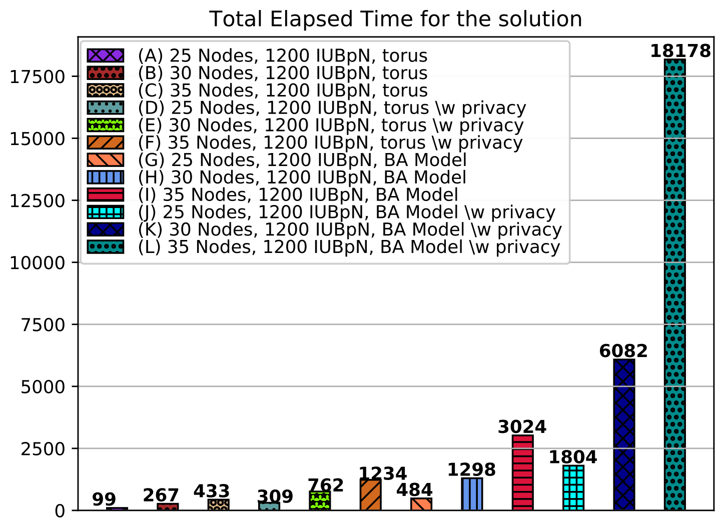
(b) Standard Deviation among nodes

Figure 7.3: Heuristic applied to topologies of BA and torus with and without privacy - Number of Edges and Standard Deviation

bogus payments. For example, in a case, we observed that the receiving node was making transactions to other nodes just to fulfill the requirements of the constraint. We solved that problem by modifying the model such that a node will not accept a transaction it makes. We either observe multi-hop flows or flows similar to the ones shown in Fig. 7.5. This forwarding scheme is acceptable as an attacker with



(a) Total Investment



(b) Total Run Time (s)

Figure 7.4: Heuristic applied to topologies of BA and torus with and without privacy - Total Network Capacity and Total Time

malicious intent will not be able to learn full details about transactions.

For this experimental setup, as expected, in order to satisfy the privacy constraint the solver consumes more time, 3 to 4 folds, compared to the normal heuristic for both topologies. Another observation is that for Torus topology, the solver converges to a solution faster compared to the BA model topology, thanks to its more ordinate structure. Likewise, the tidier arrangement in Torus topology results in a relatively

Table 7.1: Legend for Figure 7.3

Abbreviation	Legend
A	25 Nodes, 1200 IUBpN, torus
B	30 Nodes, 1200 IUBpN, torus
C	35 Nodes, 1200 IUBpN, torus
D	25 Nodes, 1200 IUBpN, torus, with privacy
E	30 Nodes, 1200 IUBpN, torus, with privacy
F	35 Nodes, 1200 IUBpN, torus, with privacy
G	25 Nodes, 1200 IUBpN, BA Model
H	30 Nodes, 1200 IUBpN, BA Model
I	35 Nodes, 1200 IUBpN, BA Model
J	25 Nodes, 1200 IUBpN, BA Model, with privacy
K	30 Nodes, 1200 IUBpN, BA Model, with privacy
L	35 Nodes, 1200 IUBpN, BA Model, with privacy

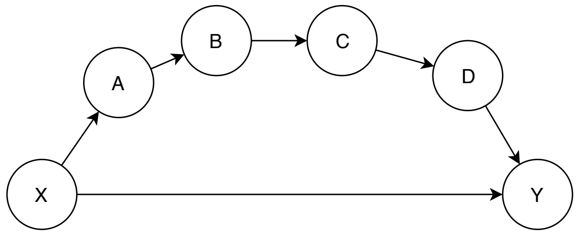


Figure 7.5: Privacy-preserving flow from X to Y.

higher number of edges. Since the edges are connected in a strict array type, the possible solutions are consumed rapidly. That's why the standard deviation tends to decrease with given IUBpN. However, BA model topologies tend to have some degree of liberty in connections.

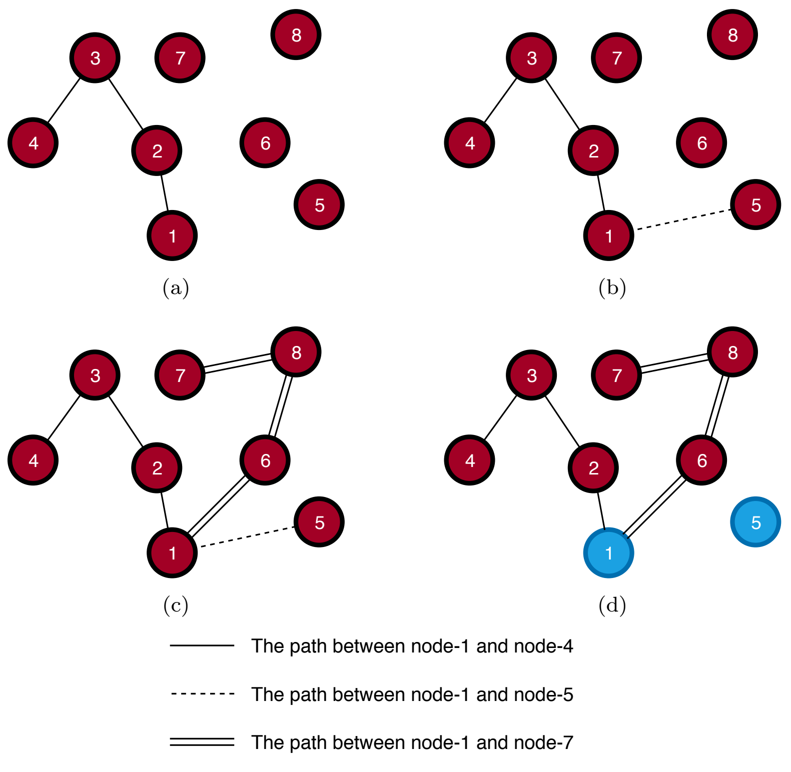
We can see that the total number of edges and total investment are only slightly higher in the privacy case while standard deviation among the investment of the nodes is smaller. This can be attributed to the fact that IUBpN upper limit in the experiment still gives enough room for successful distribution of the transactions. Additionally, in the case without privacy, some transactions are already satisfying the privacy requirement. So we do not observe a drastic increase in total capacity. However, as the node flows are closer to the IUBpN limit, we observe a lower standard deviation indicating a fair distribution of the flows which is good.

7.4 Extending the Dijktra's Shortest Path Algorithm based Heuristic for Privacy Guarantees

In Chapter 6 we have designed a new network formation method using Dijkstra's shortest path algorithm. As we have discussed for onion-routing the higher the number of hops a payment is traversing through, the better the privacy is. If at least 3-hops transaction mechanism is utilized or enforced by the users, a node on the route of the payment will not be able to get a clear view of the source and the destination about the payment [DMS04].

While the Dijkstra's shortest path algorithm is highly efficient in finding the shortest path from a source to a target in weighted directed graphs, it cannot guarantee a minimum number of hops for a path. In order to increase the privacy of payments, we propose an extension to it by relying on re-routing. Specifically, the proposed extension method is comprised of two steps. In the first step, Dijkstra's algorithm is run with the default configuration as defined Algorithm 4, and all the shortest paths for payments are found. Note that, in this configuration, there is no dictated number of hops requirement. Among all of those paths, the ones with lower than R hops are saved in a temporary list, which will be used in the second step of the method.

In the second step, the shortest path algorithm is run again for the payments in the temporary list. However, in this case, instead of finding the shortest path, k-shortest paths algorithm is utilized. k-shortest path algorithm basically lists top k shortest paths from a source to destination which requires more computation. The computational complexity of the k-shortest path algorithm is shown to be $O(E + kNlogN)$ [BELR07] where E represents the number of edges and N is the number of vertices in the graph, and k is the number of hops required. The goal here is to



3
7
8
4
2
6
5
1
(a)
(b)
(c)
(d)
The path between node-1 and node-4
The path between node-1 and node-5
The path between node-1 and node-7

Figure 7.6: Explanation of the rerouting mechanism

look for paths with at least R-hops, essentially forcing the paths that are in our list to perform a *re-routing*. Among all of the paths with R or more hops, the path with the minimum weight is picked as the solution. This process is hypothetically shown in Fig. 7.6. Based on this figure, during the Dijkstra's algorithm run, a 3-hop path between node-1 and node-4 is found as shown in Fig. 7.6(a). As the iterations carry on, a 1-hop algorithm is found between node-1 and node-5 as shown in Fig. 7.6(b) and in the next iteration another 3-hop path is found between node-1 and node-7 as illustrated in Fig. 7.6(c). Thus, when the first step is over, the re-routing with the k-shortest path algorithm starts for the 1-hop path found in Fig. 7.6(b). This 1-hop path should be forced for a re-route via more than R hops between node-1 and node-5 as shown in Fig. 7.6(d).

One might question why the first step of the proposed method is not directly utilizing k-shortest paths algorithm instead of following a two-step approach. This choice is due to the computational complexity that comes with k-shortest path algorithm with the initial topology. It takes a lot of time to find a path for the payments essentially turning the approach into a brute force search. The proposed privacy guaranteed method is shown in Algorithm 4.

7.4.1 Privacy Analysis of Dijkstra based Heuristic

In this section, we analyze how our proposed 3-hop multi-hop route enforcement along with topology can ensure relationship anonymity among payments and payees.

Thanks to onion-routing in LN, in a multi-hop payment scheme, the addresses of all of the nodes in a path are encapsulated which helps hiding the identity of the source and the destination nodes from the intermediary nodes. However, if the transactions are sent through only one relay (the source node in no relay case), the relay node will be aware of the business habits, spending and earnings of the users because the preceding node is the payer node and the next node will be the payee node. Assuming at least one of the intermediary nodes is honest, using at least 2 intermediary nodes will successfully hide the identities of the source and the destination nodes.

We elaborate this justification with some potential scenarios. In scenario 1, the payments are direct, meaning, 1-hop. In 1-hop scenario the source will have a guess on the earnings of the destination. If the topology of the network is highly centralized, that guess will be highly accurate. In another scenario, scenario-2, the payments are forwarded using 2-hops, the source will not be aware of the destination but this time the intermediary node has a chance to gather information about the earnings of the destination. In the scenario which is also depicted in Fig. 7.7, scenario-3, neither the source nor the “Curious Retailer” will have sufficient infor-

Algorithm 4 Network Establishment for Privacy Guarantees

```
1: Input: $P$=Payment List, $H$=fully connected directed graph, $L_c$=Link establishment cost, $W_i$=New connection forcing cost
2: for every edge, $e$, in $H$ do
3:     $H_e.weight$=$W_i$+$L_c$
4:     $H_e.flow$=0
5:     $H_e.Tempflow$=0
6: end for// Initial assignments are done
7: for every payment in $P$ do // A payment is defined by a source, a destination and the transfer amount $T_a$
8:     $Path$=$ShortestPath$($H$, from=$a$, to=$b$)
9:     if length($Path$)<$R$ then
10:        Add payment to Temp list
11:        for Each edge, $e$, in $Path$ do
12:            $H_e.Tempflow$ +=$T_a$
13:            $H_e.weight$=$W_i$+$H_e.Tempflow$+$H_e.flow$
14:        end for
15:    else
16:        for Each edge, $e$, in $Path$ do
17:            $H_e.flow$ +=$T_a$
18:            $H_e.weight$=$W_i$+$H_e.flow$+$H_e.Tempflow$
19:        end for
20:    end if
21: end for
22: //Remove effect of $H_e.Tempflow$ in $H$
23: //Nullify Tempflow and make edge weights $L_c$+$W_i$ of edges with only positive Tempflow
24: for every payment in Temp do
25:    $AllPaths$=$AllSimplePaths$($H$, from=$a$, to=$b$)
26:    $Path$ = $x$ where $x \in AllPaths$ if length($x$)$\geq R$ and $weight_x$ is the minimum
27:    for Each edge, $e$, in $Path$ do
28:        $H_e.flow$ +=$T_a$
29:        $H_e.weight$=$W_i$+$H_e.flow$
30:    end for
31: end for
32: for All edges in $H$ do
33:    if $H_e.flow$=0 then
34:        Remove edge from $H$
35:    end if
36: end for
37: Output: $H$
```

mation about the destination. On the other hand, if Curious Retailer starts analysis, s/he still will not be sure if s/he is the node just before the destination. Because in onion-routing Curious Retailer has to make 2 guesses. Either Node-Destination is the real destination or he is another intermediary node. As Tor uses 3 hops as a practical heuristic to increase the privacy, it turns out that it is a sufficient figure to pick.

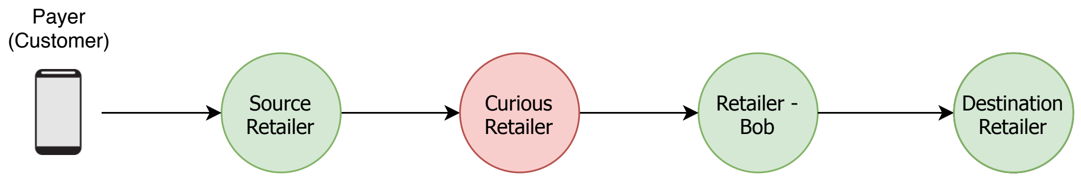


Figure 7.7: Relationship anonymity through multi-hop payments

It is worth noting that the minimum hop limit can be increased but a line should be drawn somewhere to watch for trade-offs in costs and delays. In Tor network, the probability of selecting a connection is proportional to the available bandwidth in the link. Independent from our study and similar to the Tor network's approach, a retailer can establish links to every other node with a uniformly distributed channel capacity. For sure, that approach will very much improve the mentioned privacy in the network. However, there are some drawbacks in this procedure. The first drawback, e.g., in a network with size of 500, is a user willing to open 499 links and invest in all of these links with a collateral? Most probably he will not. Second, with all these links established, if they are forced to make 3-hops payments to increase their privacy, they will have to put much more collateral than the anticipated amount otherwise frequent channel capacity depletion cases will occur. This is again a huge drawback. Therefore, in this study we picked 3-hop to achieve our objectives.

7.4.2 Evaluation of Dijkstra based Heuristic with Privacy Considerations

In this subsection, we present results related to privacy extension of our approach presented in Section 7.4 which guarantees payment privacy with at least 3-hop payments. In these experiments, we compared two other baseline approaches with ours: The first baseline approach utilizes our Dijkstra heuristic without any privacy guarantees starting with a fully connected initial topology shown with $E = All$ in the figures. The second baseline is the same as the first except that it starts with a randomly connected initial topology with 4500 edges (shown as $E = 4500$). Our approach which guarantees privacy is shown as $E = All, Reflow$. We conducted the experiments with 500 nodes. The results are shown in Figures 7.8, 7.9, 7.10, 7.11, 7.12, and 7.13 for the following metrics: betweenness centrality, total investment capacity, the total number of edges, the number of hops, the standard deviation and total processing time.

As seen in Fig. 7.8, with a randomly connected initial topology, the betweenness centralities of the nodes change drastically, which is expected. This is because with the random connection some of the nodes will be apparently dominant compared to others. However, betweenness centrality of the initially fully connected mesh network topologies gives a more flat measure. This behavior assures that the dominance of some of the nodes is decreased significantly. The initial fully connected mesh network topology and pseudo-random payment distribution is effective in getting that result. However, our approach with privacy guarantees is slightly better than the one without privacy for betweenness centrality. As shown in 7.10, while the number of edges did not change, the betweenness centrality measure is maintained. This is because some of the channels opened in the first step of the privacy approach are discarded in the second step which eventually helped in establishing a more balanced network.

The other results for the experiment with the privacy guarantee are promising too. Although the number of edges does not change, the main factor for the slight increase in the total capacity with respect to the approach where privacy is not guaranteed is the forcing of minimum 3 hops in the transfers which causes each node to invest more, meaning an additional investment for the others' payments too. Nonetheless, our privacy-guaranteed approach comes with the best standard deviation, even surpassing the non-privacy one as shown in Fig. 7.12. The one-time cost for these improvements comes with some time overhead as seen in Fig. 7.13. The total calculation time for the setup with the random connection network is the lowest because the complexity of the Dijkstra is directly related to the number of edges. For the privacy guaranteed approach, the time is the highest because all of the 1 and 2 hops payments were converted to at least 3 hops transfers by the k-shortest path algorithm which in turn brings an additional overhead to the total computation time.

We hence further extended this heuristic to guarantee privacy-aware routing in the network with at least 3 hops and compared its performance with the non-privacy case. When privacy is to be guaranteed with at least 3 hops, the results show that the network topology becomes even better in terms of the considered topological metrics with some additional computation time.

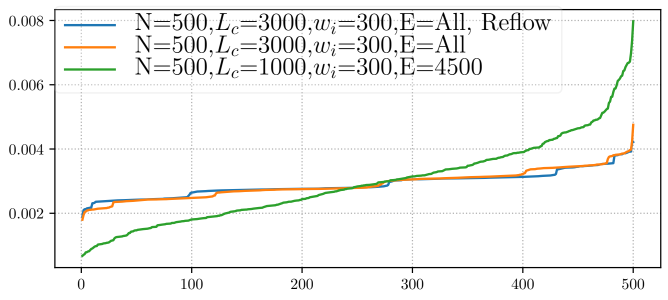


Figure 7.8: Privacy aware method for 500 nodes - Betweenness Centrality

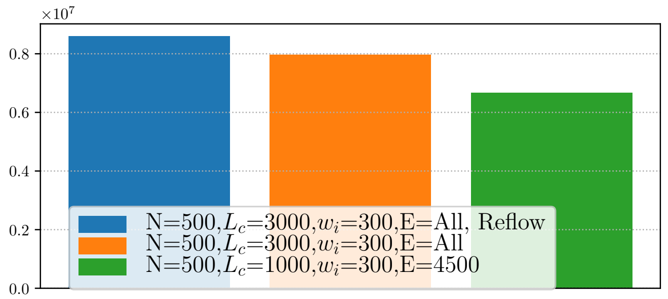


Figure 7.9: Privacy aware method for 500 nodes - Network Total Flow Capacity

7.5 A Comparative Analysis of the State of the Art PCN Implementations

In this section we first give a background on network architectures and types of Blockchain networks. We then compare state-of-the-art PCN proposals with our PCN formation proposal in terms of the privacy.

7.5.1 Network Architectures

In this section we present a brief background on the types of network architectures.

Centralized Networks

In this type of network there is a central node and users communicate with each other either over that central node or based on the information received from the central

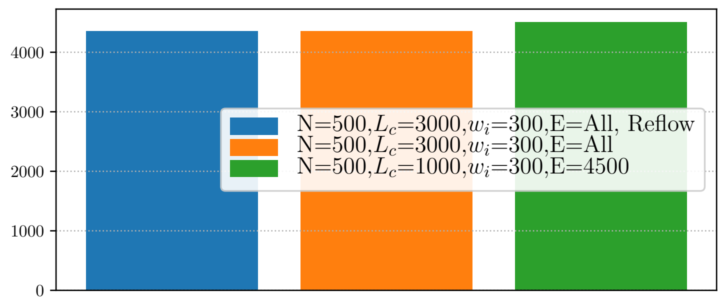


Figure 7.10: Results for the Privacy aware method for 500 nodes - Total Edges

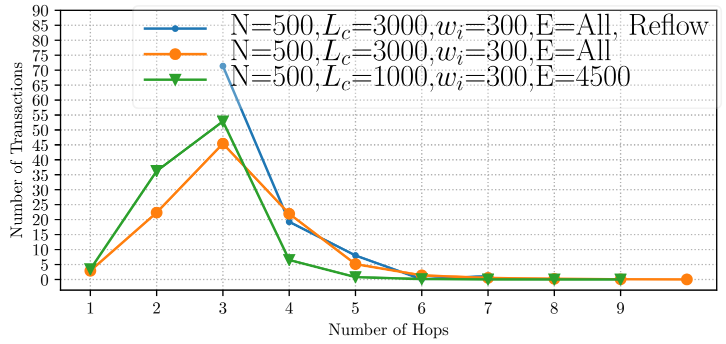


Figure 7.11: Privacy aware method for 500 nodes - Number of Hops

node. From the governing point of view, informally defining, if an organization or a company can solely decide on the connections, capacity changes, and flows in the network then this network is called to be a centralized network.

Distributed Networks

In distributed networks there is no central node. As opposed to the centralized network, each user has the same connectivity, right to connect, and voice in the network, namely, the nodes and right to connect are distributed equally.

Decentralized Networks

This type of network is the combination of the previous two types. In this network there is no singular central node, but there are independent central nodes. When the child nodes are removed, central nodes' connections look very much like a distributed

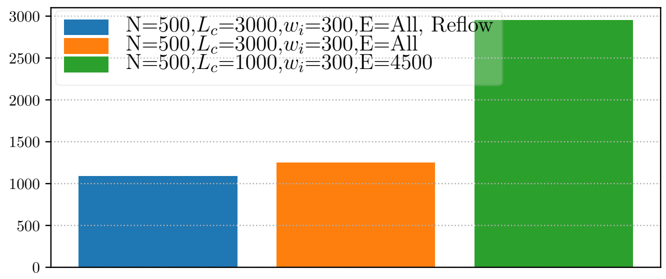


Figure 7.12: Privacy aware method for 500 nodes - Standard Deviation

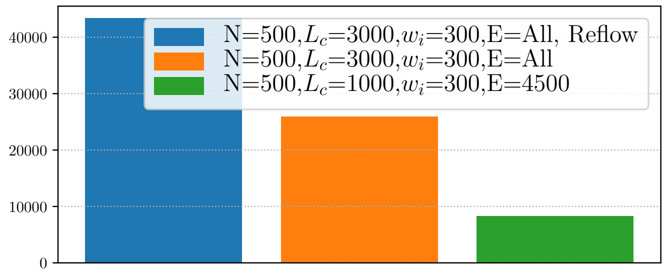


Figure 7.13: Results for the Privacy aware method for 500 nodes - Total Time (s)

network. However, when the view is concentrated around one of the central nodes it is seen that in the small scale there is a centralized network. Today's PCNs can be classified in this type. Although the users are free to connect to any other node, we see that (e.g., Lightning Network) some nodes are popular (highly influential) and they establish more channels.

Federated Networks

Federated networks sound very much like the federation of the states in the real world, and arguably lies somewhere between centralized and decentralized networks. In federated networks there are many central nodes where they are connected to each other in a P2P fashion. Then the remaining nodes (the children nodes) communicate with each other over these central nodes which very much looks like a federation of centralized networks. Moreover, each federation can come up with their local rules in addition to the protocol being used.

7.5.2 Types of Blockchain Networks

In this section we present a brief background on the types of blockchains. This will help better understand the privacy approaches of the studies, their strength, and their weakness. There are mainly three types of Blockchain networks.

Public Blockchain

Public blockchain is, as the name suggests, the blockchain where no binding contract or registration is needed in order to be a part of the network. Users can join or leave the network whenever they want.

Permissioned Blockchain

Permissioned (i.e., Private) blockchain lays at the opposite side of the public blockchain. In permissioned blockchain the ledger is managed by a company or an organization. Besides, the roles of the nodes within the network are assigned by the central authority. Not everybody can participate or reach to the resources in the permissioned blockchain network. Blockchains used for supply chain management are good examples for this type of blockchain.

Consortium Blockchain

Contrary to the permissioned blockchain, in consortium blockchain the blockchain is governed by more than one organization. From the centralization point of view, this approach seems more liberal but the governance model of the Blockchain slides it to the permissioned side. Facebook's Libra cryptocurrency is a good example for this type of blockchain.

7.5.3 State-of-the-art PCN Proposals

In this section we tabulate the state of the art PCN proposals and frameworks with respect to the privacy they satisfy in Table 7.2. We compare those studies with our results and the privacy implications of our study.

Table 7.2: Comparative Table

	Network Type	Blockchain Type	Sender Anonymity	Recipient Anonymity	Channel Balance Privacy	Relationship Anonymity	Business Volume Privacy	Remarks
Lightning Network (HTLC) [PD16]	Decentralized/ Distributed	Public	●	●	◑	●	●	Onion routing, network topology is important
Spider [SVR+18]	Decentralized/ Centralized	All	◑	◑	○	◑	◑	Packet-switching discloses identities. Attacks on spider routers is a threat to privacy. Centralization is highly possible
SilentWhispers [MMSKM17]	Decentralized/ Centralized	All	◑	◑	◑	◑	◑	Privacy depends on the security of landmarks. Colludoing malicious landmarks will learn everything
Speedy-Murmurs [RMSKG17]	Decentralized/ Centralized	Public	◑	◑	◑	◑	◑	Better than SilentWhispers but Landmarks are still threat to the privacy.
PrivPay [MSKMP15]	Decentralized/ Centralized	Permissioned	◑	◑	◑	◑	◑	Privacy depends on trusted hardware platforms. Easily slides towards centralization
Flash [WXJW19]	Decentralized/ Distributed	Public	●	◑	○	●	○	The protocol lets users to probe the channel capacities. If an honest user excessively probes the network anonymity can be broken.
Flare [PZS+16]	Distributed	Public	●	●	◑	●	●	Each node designates her own landmarks. Attack on a node will reveal partil information about the network.
Bolt [GM17]	Centralized	Public	●	●	●	●	●	It is a payment hub-network and centralization is inevitable. Attacks on the central node is a threat to privacy.
Rayo and Fulgor [MMSK+17]	Decentralized/ Distributed	Public	◑(Rayo) ●(Fulgor)	◑(Rayo) ●(Fulgor)	◑	◑(Rayo) ●(Fulgor)	●	Sutiable for Bitcoin network. Fulgor introduces extra overhead in communication which hinders applicability. Rayo concentrates on concurrency issue and anonymity can not be protected.
Erdin et al. (Our Studies)	Distributed/ Federated	All	◑	●	◑	◑	●	A consortium is proposed. Orthogonal to routing algorithms. Relationship anonymity will be protected.
Anonymous Multi-Hop Locks (AMHL) [MMSS+19]	Decentralized/ Distributed	Public	○	●	◑	●	●	A modification to LN is proposed to prevent wormhole attack. Sender anonymity is under threat.

CHAPTER 8

CONCLUSION AND FUTURE WORK

Payment channel networks mark a new epoch for the use of cryptocurrencies in daily transactions like today's credit cards. Payment networks are formed by the gathering of the payment channels that are established between the peers in the network and by that means, the owner of the payment system is the users themselves. Lightning Network is the mainstream and the most attractive payment channel network being actively developed as of today. LN is built on top of Bitcoin network in order to remove the drawbacks native to the Bitcoin that are, slow confirmation times and disproportional fees asked by the Bitcoin miners. Although LN is seem to be the most mature PCN it is still in its beta phases and without any doubt further development and capability enhancements are on the way.

In this dissertation we concentrated on the methods for obtaining P2P decentralized payment channel network topologies. Inspiring from the multi-commodity flow problem, we first developed an optimal solution to establish the perfect payment channel network topology by utilizing mixed integer programming. As multi-commodity flow problem is known to be NP-complete in computational complexity, we developed a sub-optimal optimization heuristic utilizing linear programming. Although, that approach gave remarkable solutions in terms of scalability we felt the need for further scaling at least comparable to the current LN user set. So, we developed a heuristic algorithm to come up with a PCN formation using Dijkstra's shortest path algorithm. We finally, investigated state of the art PCN algorithms with respect to the privacy they provide. We compared our solution to them. Although our studies are orthogonal to the offered routing algorithms we show that in order to increase the security and the privacy in the payment channel networks a distributed network has to be formed. Until now none of the studies mentioned this problem. We show that by forming a federated network, the retailers can create a payment channel network and the customers can privately connect to these retailer nodes which removes the frictions in the current PCN formations.

PCNs are investigated in many aspects from security to privacy of the users to fixing channel depletion problem. However, there is a big gap in explaining how the topology should be established in order for the healthy continuation of the operations. We believe, we are the first investigating this research topic.

The resultant topology for the mixed-integer optimization (MIOP) ensured that the costs for establishing channels among stores are minimized. The topology also favors privacy since it prevents formation of hub nodes that can potentially monitor the source and destination of all payments.

The evaluation of the linear programming heuristic with Python and Gurobi indicated that the performance of the heuristic approach is very close to the MIOP solution while allowing a certain scalability. The results also suggested that the topologies generated by Barabasi-Albert came as a favorable initial topology that can scale much faster and still provides comparable results as long as the upper bound for the channel capacities are picked rightly.

For the results of the Dijkstra based heuristic, compared to the optimal solution, the heuristic reduces the computational time significantly. Additionally, the fair distribution of the load among nodes, centrality measures and the total number of edges obtained in the networks are satisfying to ensure a truly P2P network topology features. When privacy is to be guaranteed with at least 3 hops, the results show that the network topology becomes even better in terms of the considered topological metrics with some additional computation time overhead.

Thanks to PCNs being in their early phases, there are many open areas worth studying.

- In our study the nodes were static. All of the nodes were assumed to attend to the network at the beginning. It is worth investigating how the network and the capacities should change in a dynamic environment where some nodes can leave the network or new nodes attend.
- In our algorithms in order to be able discuss the unfairness between the investment needs of the nodes we assumed an equal income for the retailers. However, in real world this will not be the case as some of the stores will be more favorable than the others. Our results will be a basis for investigating nonuniform income expectations for the retailers. Observing the final topology and developing methods for keeping the P2P behavior of it will be interesting.
- In real life the nodes in the network will try to benefit from the network as much as they can. Strategical moves of the participants are very aligned with the game theoretical approach for network formation. Unfairness perception can be better explained with a game theoretical approach.
- Atomic multi-path payment is recently evolved from theory to practice in LN. Although our MIOP solution supports atomic multi-path payments a scalable version of it should be established. It is worth investigating how small should the payments be and how it affects the required channel capacities.
- A formal analysis of privacy needs to be made for PCNs. As we see in the state of the art solutions when privacy is protected, usability of the network decays (e.g., payment hubs). If usability increases privacy concerns arise (e.g., Rayo and Fulgor). Hence, the relation between privacy, security and operational usability is worth investigating.

[1ML20] 1ML. Lightning network search and analysis engine. https://1ml.com, May 2020.

[A$^+$06] Krste Asanovic et al. The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[ACI] ACINQ. A scala implementation of the lightning network. https://github.com/ACINQ/eclair.

[BAN17] Alican Bozyiğit, Gazihan Alankuş, and Efendi Nasiboğlu. Public transport route planning: Modified dijkstra's algorithm. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 502–505. IEEE, 2017.

[BELR07] Eric Bouillet, Georgios Ellinas, Jean-Francois Labourdette, and Ramu Ramamurthy. *Path Routing–Part 2: Heuristics*. Wiley Telecom, 2007.

[Bita] Bitcoin wiki. Bitcoin contract. en.bitcoin.it/wiki/Contract.

[Bitb] Hash Time Locked Contracts. en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.

[BKP14] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29, 2014.

[Bro17] Ryan Browne. Big transaction fees are a problem for bitcoin but there could be a solution. cnbc.com/2017/12/19/big-transactions-fees-are-a-problem-for-bitcoin.html, 2017.

[DIA18] DIAR. Lightning Strikes, But Select Hubs Dominate Network Funds. diar.co/volume-2-issue-25, June 2018.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[Din70] Efim A Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Math. Doklady*, volume 11, pages 1277–1280, 1970.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[ECA$^+$18] Enes Erdin, Mumin Cebe, Kemal Akkaya, Senay Solak, Eyuphan Bulut, and Selcuk Uluagac. Building a private bitcoin-based payment network among electric vehicles and charging stations. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1609–1615. IEEE, 2018.

[EIS75] Shimon Even, Alon Itai, and Adi Shamir. On the complexity of time table and multi-commodity flow problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 184–193. IEEE, 1975.

[Ele] Elements Project. c-lightning — a lightning network implementation in c. https://github.com/ElementsProject/lightning.

[FF56] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[Fou16] Ethereum Foundation. Raiden network. https://raiden.network, 2016. Online; accessed 18 May 2020.

[GM17] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489, 2017.

[GMSR$^+$19] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Off the chain transactions. *IACR Cryptology ePrint Archive*, 2019:360, 2019.

[HAB$^+$17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoincompatible anonymous payment hub. In *Network and Distributed System Security Symposium*, 2017.

[HJNAP$^+$19] Jordi Herrera-Joancomarti, Guillermo Navarro-Arribas, Alejandro Ranchal Pedrosa, Perez-Sola Cristina, and Joaquin Garcia-Alfaro. On the difficulty of hiding the balance of lightning network channels. *AsiaCCS*, 2019.

[JKLT19] Maxim Jourenko, Kanta Kurazumi, Mario Larangeira, and Keisuke Tanaka. Sok: A taxonomy for layer-2 scalability related protocols for cryptocurrencies. *IACR Cryptology ePrint Archive*, 2019:352, 2019.

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 20(1), 1980.

[Lig16] Lightning RFC. Bolt 4: Onion routing protocol. `github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md`, 2016.

[Mar19] Stefano Martinazzi. The evolution of lightning network's topology during its first year and the influence over its core values. *arXiv preprint arXiv:1902.07307*, 2019.

[MBB$^+$19] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In *International Conference on Financial Cryptography and Data Security*, pages 508–526. Springer, 2019.

[MMSK$^+$17] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 455–471. ACM, 2017.

[MMSKM17] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Silentwhispers: Enforcing security and privacy in decentralized credit networks. In *NDSS*, 2017.

[MMSS$^+$19] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *NDSS*, 2019.

[MSKMP15] Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Kim Pecina. Privacy preserving payments in credit networks. In *Network and Distributed Security Symposium*, 2015.

[Opt17] Gurobi Optimization. Gurobi optimizer 7.2. gurobi.com, 2017.

[P$^+$15] Rafael Pass et al. Micropayments for decentralized currencies. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 207–218. ACM, 2015.

[Pap81] Christos H Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

[PD16] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[PZS$^+$16] Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. Flare: An approach to routing in lightning network. *White Paper*, 2016.

[Rai17] Raiden. https://raiden.network/, 2017.

[RMSKG17] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748*, 2017.

[RMT19] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks. *2019 IEEE European Symposium on Security and Privacy Workshops*, 2019.

[Sat09] Satoshi Nakamoto. Bitcoin v0.1 released. https://www.metzdowd.com/pipermail/cryptography/2009-January/014994.html, January 2009.

[SGNB20] István András Seres, László Gulyás, Dániel A Nagy, and Péter Burcsi. Topological analysis of bitcoin's lightning network. In *Mathematical Research for Blockchain Economy*, pages 1–12. Springer, 2020.

[Sta18] StackExchange. How does lnd's autopilot feature work? `https://bitcoin.stackexchange.com/questions/75097/how-does-lnds-autopilot-feature-work`, 2018.

[Sto18] StopAndDecrypt. Them lightning network nodes sure do look centralized to me! what gives? https://hackernoon.com/them-lightning-network-nodes-sure-do-look-centralized-to-me-what-gives-ee39c9b12ac0, May 2018.

[Sub17] Hemang Subramanian. Decentralized blockchain-based electronic marketplaces. *Communications of the ACM*, 61(1):78–84, 2017.

[SVA$^+$18] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia Fanti, and Pramod Viswanath. Routing cryptocurrency with the spider network. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 29–35, 2018.

[SVR$^+$18] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathy Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Mohammad Alizadeh, and Giulia Fanti. High throughput cryptocurrency routing in payment channel networks, 2018.

[Tea] Lightning Network Team. Atomic cross-chain trading. https://en.bitcoin.it/wiki/Atomic cross-chain trading.

[Tec] Lightning Technologies. Lightning network daemon. https://github.com/lightningnetwork/lnd.

[TMSM20] Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. A quantitative analysis of security, anonymity and scalability for the lightning network. *IACR Cryptology ePrint Archive*, 2020:303, 2020.

[TPA12] Gill R Tsouri, Alvaro Prieto, and Nikhil Argade. A modified dijkstra's routing algorithm for increasing network lifetime in wireless body area networks. In *BODYNETS*, pages 166–169, 2012.

[Tru18] TrustNodes. Lightning network ddos sends 20% of nodes down. *trustnodes.com/2018/03/21/lightning-network-ddos-sends-20-nodes*, 2018.

[TS15] Stefan Thomas and Evan Schwartz. A protocol for interledger payments. https://interledger. org/interledger. pdf, 2015.

[Wik] Bitcoin Wiki. Contracts: Example 7: Rapidly-adjusted (micro) payments to a pre-determined party.

[WXJW19] Peng Wang, Hong Xu, Xin Jin, and Tao Wang. Flash: efficient dynamic routing for offchain networks. *arXiv preprint arXiv:1902.05260*, 2019.

VITA

ENES ERDIN

June 17, 1984	Born, Turkey
2006	B.S., Electrical and Electronics Engineering Middle East Technical University Ankara, Turkey
2009	M.S., Electrical and Electronics Engineering Middle East Technical University Ankara, Turkey
2006–2016	Senior Digital Design Engineer TUBITAK-SAGE Ankara, Turkey
2019	Administrative Intern Iowa Office of Chief Information Officer Des Moines, Iowa
2016–present	PhD Candidate Florida International University Miami, Florida

PUBLICATIONS AND PRESENTATIONS

Mercan, S., Erdin, E., Akkaya, K., (2020) *Improving Transaction Success Rate via Smart Gateway Selection in Cryptocurrency Payment Channel Networks.* arXiv preprint arXiv:2003.10877.

Mercan, S., Erdin, E., Akkaya, K., (2020) *Improving Sustainability of Cryptocurrency Payment Networks for IoT Applications.* arXiv preprint arXiv:2003.00294.

Erdin, E., Cebe, M., Akkaya, K., Solak, S., Bulut, E., Uluagac, S., (2020) *A Bitcoin Payment Network with Reduced Transaction Fees and Confirmation Times.* Computer Networks, Volume 172, 8 May 2020, 107098, 1-13.

Kurt, A., Erdin, E., Cebe, M., Akkaya, K., Uluagac, S., (2019) *LNBot: A Covert Hybrid Botnet on Bitcoin Lightning Network.* arXiv preprint arXiv:1912.10617.

Erdin, E., Cebe, M., Akkaya, K., Bulut, E., Uluagac, S., (2019) *A Heuristic-Based Private Bitcoin Payment Network Formation Using Off-Chain Links.* 2019 IEEE International Conference on Blockchain (Blockchain), 294-301.

Denney, K., Erdin, E., Babun, L., Vai, M., Uluagac, S., (2019) *USB-Watch: A Dynamic Hardware-Assisted USB Threat Detection Framework.* International Conference on Security and Privacy in Communication Systems, 126-146.

Denney, K., Erdin, E., Babun, L., Uluagac, S., (2019) *Dynamically detecting USB attacks in hardware: poster.* Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks.

Erdin, E., H Aksu, Uluagac, S., Vai, M., Akkaya, K., (2018) *OS Independent and Hardware-Assisted Insider Threat Detection and Prevention Framework.* MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM), 926-932.

Cebe, M., Erdin, E., Akkaya, K., H Aksu, Uluagac, S., (2018) *Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles.* IEEE Communications Magazine 56 (10), 50-57.

Erdin, E., Cebe, M., Akkaya, K., Solak, S., Bulut, E., Uluagac, S., (2018) *Building a private bitcoin-based payment network among electric vehicles and charging stations.* 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 1609-1615.

Erdin, E., Kılcıoğlu, Ç., Yılmaz, A., (2011) *Implementation and performance of parallellised turbo decoders.* IET communications 5 (1), 39-50.