Designing and Evaluating Rebalancing Algorithms for Payment Channel Networks

Yuup R.J.M. van Engelshoven

Designing and Evaluating Rebalancing Algorithms for Payment Channel Networks

by

Yuup R.J.M. van Engelshoven

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday December 10, 2019 at 15:00.

Student number:	4076583	
Project duration:	January 1, 2019 – December 10, 2019	
Thesis committee:	Prof. dr. ir. D. Epema,	TU Delft
	Dr. S. Roos,	TU Delft, supervisor
	DR. P. Pawelczak,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Abstract

Payment Channel Networks(PCN) utilize payment channels with an established link capacity between two nodes to route transactions over multiple links to carry out transactions. Such transactions can support a blockchain due to the transactions happening off-chain, i.e., not requiring any information to be published to a ledger. PCNs can help aid in the scalability of blockchains, by moving transactions off-chain not all transactions need to be stored on the blockchain, reducing the amount of data that needs to be stored on the blockchain. Lightning is the PCN implementation that makes use of Bitcoins blockchain.

As transactions occur over the network the capacity of the link may vary over time between two nodes. This change may lead to the link being only available from one side. If enough links become unavailable then processing transactions may take longer or in the worst-case scenario transactions may no longer be feasible in the network. To help avoid these short-comings in PCN strategies can be designed in path-based transaction algorithms to help keep links capable of handling transactions bidirectionally.

This work presents two such algorithms, the Passive Merchant and Active Merchant. Additionally, two synthetic data-set models are proposed to help evaluate the effectiveness of the Merchant algorithms, due to a lack of data-sets in this field. In the evaluation two different topologies are examined to evaluate the impact a graph has on the success rate of transactions within a PCN.

The evaluation of the The Merchant algorithms is simulation based, experiments evaluated how different algorithms effected the success rate of transactions. The simulation did indicate that the algorithms were able to help increase the success rate of transactions, up to 8%. As these algorithms are embedded in the transaction process of a payment the algorithms are a first of there kind, other solutions have been proposed for rebalancing as a separate protocol. In addition to being the first to propose transaction embedded rebalancing algorithms, no other synthetic data-set models for PCN have been proposed. The synthetic data-set models may allow this area of research to be have constant data-sets that are used to evaluate the effectiveness of path-based transactions.

Preface

During the past year I have been able to contribute to a fairly new research area in Distributed Systems, namely Payment Channel Networks. By no means can I claim this contribution to be solely an independent effort. Without the amazing supervision from Dr. Stefanie Roos, who guided me throughout the process this contribution would not be in the state it is in now. She managed to steer my thoughts and ideas as I thought every topic was interesting. Like a kid in a candy-store wanting to try it all. Her advise always came at the right-time, and her guidance helped me pick a path to focus on. For her ongoing advise and effort she put into making my thesis a reality, I would like to thank Stefanie for everything she has done the past year.

Getting through university has not always been a walk in the park. There have been many ups and a lot of downs through my education, and without the support of my family and friends this journey may of taken a different path. I'd like to thank my parents, Birgiet and Raoul, for their on going support and patience. Believing in me no matter the situation I found myself in. I can imagine that some of these situations must of been frustrating to watch, but signs of frustration were kept out of sight and all I received was support. For that I thank you.

As my contribution to Payment Channel Networks cannot be solely claimed as an independent effort, neither can my unforgettable time in Delft be claimed as an independent effort. Without the support and advise of my friends this time would of been shaped very differently. For that I want to thank everyone that has helped and been supportive even during this time. As the stress built over the last couple of months to finish this thesis, my roommates had to endure a constant stream of complaints and moans of suffering. For all their patience the past couple of months I thank you.

Yuup R.J.M. van Engelshoven
Delft, December 2019

Contents

List of Figures xi

1 Introduction 1
- 1.1 Utilizing Off-Chain Transactions . . . 2
- 1.2 Objective . . . 3
- 1.3 Contributions . . . 3
- 1.4 Project Overview . . . 4

2 Background on Block-chains and Payment Channels 5
- 2.1 Blockchain . . . 5
  - 2.1.1 Building Blocks of Blockchain . . . 5
  - 2.1.2 Consensus Algorithms . . . 6
  - 2.1.3 Types of Blockchain Transaction Models . . . 7
  - 2.1.4 Classification of Blockchains . . . 8
  - 2.1.5 Security in Blockchain . . . 8
  - 2.1.6 Privacy in Blockchain . . . 10
  - 2.1.7 Scalability by Increasing Blocksize . . . 10
  - 2.1.8 Scalability with a Layer-2 Protocol . . . 11
- 2.2 Payment Channels and Payment Channel Networks . . . 11
  - 2.2.1 Payment Channel . . . 11
  - 2.2.2 Payment Channel Networks . . . 12
  - 2.2.3 Transactions . . . 14
  - 2.2.4 Concurrency . . . 15
  - 2.2.5 Deadlock Detection . . . 15
  - 2.2.6 Rebalancing . . . 16
  - 2.2.7 Topology . . . 16
  - 2.2.8 Security and Privacy . . . 16

3 Related Work Concerning Payment Channels and An Introduction to SpeedyMurmurs 17
- 3.1 Payment Channel Networks . . . 17
  - 3.1.1 Existing Routing Algorithms for PCN . . . 18
  - 3.1.2 Concurrency: Blocking and Non-Blocking Protocols . . . 19
  - 3.1.3 Evaluating against related work . . . 20
  - 3.1.4 Topology . . . 21
  - 3.1.5 Dataset . . . 21
- 3.2 Rebalancing of a Payment Channel Network . . . 22
  - 3.2.1 REVIVE . . . 22
  - 3.2.2 Rebalancing in Acyclic Payment Networks . . . 22
- 3.3 SpeedyMurmurs . . . 22
  - 3.3.1 Routing and Path Selection . . . 23
  - 3.3.2 Concurrency . . . 23
  - 3.3.3 Security . . . 23
  - 3.3.4 Privacy . . . 24

4 Goals and a Debut for Merchant Algorithms 25
- 4.1 Fees and incentive . . . 25
  - 4.1.1 Rebalancing Potential of a Payment Channel . . . 25
  - 4.1.2 Fees vs Multi-hops . . . 26
  - 4.1.3 Rebalancing Fee Factor . . . 27

4.2 Merchant Algorithms . . . 28
4.2.1 SpeedyMurmurs and the inclusion of the Merchants. . . . 28
4.2.2 The Merchants . . . 28
4.2.3 Passive Merchant . . . 28
4.2.4 Completion of the Merchants . . . 33
4.2.5 Attacker Model . . . 35

5 Modeling the Synthetic Dataset and Toplogies 37
5.1 Challenges . . . 37
5.2 Market Interest in Cryptocurrencies . . . 37
5.3 Transaction Distribution . . . 37
5.3.1 Transaction Value Distribution . . . 38
5.3.2 Ripple Transaction Value Distribution Model . . . 38
5.3.3 Lightning Transaction Value Distribution Model . . . 39
5.3.4 Payment hop-count Distribution . . . 39
5.3.5 Ripple Hop-Count Distribution Model . . . 39
5.3.6 Lightning Hop-Count Distribution Model . . . 39
5.4 Channel Capacity Distribution . . . 39
5.4.1 Ripple Channel Capacity Distribution . . . 39
5.4.2 Lightning Channel Capacity Distribution . . . 39
5.5 Topologies . . . 40
5.5.1 Lightning Topology . . . 40
5.5.2 Barabasi-Albert Model . . . 40
5.5.3 Erdos-Renyi Model . . . 41

6 Methodology 43
6.1 General Workflow . . . 43
6.2 Technical Specs . . . 43
6.2.1 OMNET++ FrameWork . . . 44
6.2.2 Limitations in Implementation Simulator . . . 44
6.3 Generate Input Files . . . 44
6.3.1 Channel Distribution . . . 45
6.3.2 Transaction Distribution . . . 45
6.4 Topology Creation . . . 46
6.4.1 Scale-Free . . . 46
6.4.2 Erdos-Renyi . . . 46
6.5 Initialize Simulation . . . 46
6.5.1 Spanning-Tree Creation . . . 46
6.6 Simulation of Payment Channel Network . . . 47
6.6.1 Successful Transaction Messaging . . . 47
6.6.2 Non-Successful Transaction Messaging . . . 47
6.6.3 Sender Node . . . 48
6.6.4 Forwarder Node . . . 48
6.6.5 Receiver Node . . . 49
6.6.6 Latency . . . 49
6.6.7 Finding Paths . . . 49

7 Evaluation and Highlighting the Interesting Bits 51
7.1 Description of Generated Topologies . . . 51
7.1.1 Methodology and Metrics . . . 51
7.1.2 Experimental Setup . . . 51
7.1.3 Results . . . 52
7.1.4 Discussion and Conclusion . . . 52
7.2 Description of Data-Set Generator . . . 53
7.2.1 Methodology and Metrics . . . 53
7.2.2 Experimental Setup . . . 53
7.2.3 Results . . . 54
7.2.4 Discussion and Conclusion . . . 55

7.3 Description of the Concurrency in the Networks . . . 56
7.3.1 Methodology and Metrics . . . 56
7.3.2 Results . . . 56
7.3.3 Discussion and Conclusion . . . 57
7.4 Description of the Lightning Model . . . 57
7.4.1 Methodology and Metrics . . . 57
7.4.2 Experimental Setup . . . 57
7.4.3 Results Scale-Free . . . 58
7.4.4 Discussion Scale-Free . . . 59
7.4.5 Results Erdos-Renyi . . . 60
7.4.6 Discussion Erdos-Renyi . . . 62
7.4.7 General Discussion and Conclusions . . . 62
7.5 Description of the Ripple Model . . . 63
7.5.1 Methodology and Metrics . . . 63
7.5.2 Experimental Setup . . . 63
7.5.3 Results Scale-Free . . . 63
7.5.4 Discussion Scale-Free . . . 65
7.5.5 Results Erdos-Renyi . . . 66
7.5.6 Discussion Erdos-Renyi . . . 67
7.5.7 General Discussion and Conclusion . . . 68
7.6 Evaluation of the Success Rate with the Merchant Algorithms. . . . 68
7.6.1 Methodology and Metrics . . . 68
7.6.2 Experiment Set-up . . . 68
7.6.3 Results . . . 69
7.6.4 Discussion and Conclusion . . . 70
7.7 Evaluation of the Overhead with the Merchant Algorithms . . . 70
7.7.1 Methodology and Metrics . . . 70
7.7.2 Results . . . 70
7.7.3 Discussion and Conclusion . . . 71

8 Conclusion and Future Work **73**
8.1 Conclusion . . . 73
8.2 Future Work. . . . 73

Bibliography **75**

A Ripple Dataset Analysis **79**
A.0.1 Node Connection Distribution . . . 79
A.0.2 Channel Capacity . . . 80
A.0.3 Value of Transactions . . . 81

List of Figures

1.1 Triangle of trade-offs for current blockchain implementations 1
1.2 Representation of a Payment Channel Network, where the circles represent nodes and the links represent payment channels. The red lines indicate how a route can be made so that node A, is able to transact with node B. 2

2.1 Illustration of how the previous hash-pointer is connected in a blockchain data structure. 5
2.2 Layer-2 protocols that work on blockchain 11
2.3 Alice, Bob and Charlie have created a Payment Channel Network 12
2.4 A. Alice and Bob transact $150 with one another B. Charlie transacts with Alice for $150 C. PCN is closed and stored on blockchain 13

3.1 Graph of Raiden Network December 2019 17
3.2 Coordinate assignment by root node, prefix-embedding based routing 23

4.1 Labeling of nodes and there links 25
4.2 Transactions vs Three Sets of Channel Distribution 26
4.3 Direction of coupon after an imbalance is established 30

5.1 Topology of Lightning Network overlaying a world map generated by Lightning Network Explorer[5] 38
5.2 Link Capacity Distribution Lightning Network, October 2019 40

6.1 Messages that occur during a successful transaction 47
6.2 Messages that occur during a successful transaction 48

7.1 Transaction Value Distribution for three different Transaction Value Probabilities 54
7.2 Channel Distributions for the 3 sets of Channels 54
7.3 Transaction distributions for the 3 sets of Channels modeled after the Ripple Model 55
7.4 Channel distributions for the 3 sets of channels modeled after the Ripple Model 55
7.5 Transaction value averages vs three sets of channel distribution with the Lightning model in a scale-free topology 58
7.6 Transaction value averages vs a channel distribution with an average of 600 using the Lightning model in a scale-free topology 58
7.7 Channel distributions vs three sets of transaction distributions with the Lightning Model in a scale-free topology 59
7.8 Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a scale-free topology using the Lighting model 59
7.9 Transaction value averages vs three sets of channel distribution with the Lightning model in a Erdos-Renyi topology 60
7.10 Transaction value averages vs a channel distribution with an average of 600 using the Lightning model in a Erdos-Renyi topology 61
7.11 Channel distributions vs three sets of transaction distributions with the Lightning Model in a Erdos-Renyi topology 61
7.12 Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a Erdos-Renyi topology using the Lighting model 62
7.13 Transaction value averages vs three sets of channel distribution with the Ripple model in a scale-free topology 64
7.14 Transaction value averages vs a channel distribution with an average of 600 using the Ripple model in a scale-free topology 64
7.15 Channel distributions vs three sets of transaction distributions with the Ripple Model in a scale-free topology 65

7.16 Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a scale-free topology using the Ripple model 65
7.17 Transaction value averages vs three sets of channel distribution with the Ripple model in a Erdos-Renyi topology 66
7.18 Transaction value averages vs a channel distribution with an average of 600 using the Ripple model in a Erdos-Renyi topology 66
7.19 Channel distributions vs three sets of transaction distributions with the Ripple Model in a Erdos-Renyi topology 67
7.20 Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a Erdos-Renyi topology using the Ripple model 67

A.1 Node connectivity distribution pulled from the static and dynamic dataset used by Roos 80
A.2 Transaction distribution 82

1 Introduction

Late 2008, the majority of the world's financial market was sent into a downward spiral, due to the aftermath of unethical business practices by major lending institutions[43]. These bad lending practices with the combination of a real-estate bubble were a recipe for disaster. As a domino effect cascaded through the markets dropping stock values with the end result being the biggest financial crisis since the great depression[43]. Markets crashed and trust in fiat currencies dwindled. During this time Satoshi Nakamoto came forward with a digital currency in his white paper, "Bitcoin P2P e-cash paper"[51]. Satoshi's white paper set the rise of digital currencies in-motion.

A couple of months after publishing the white paper, in January of 2009 the first version of bitcoin was released. Since then the cryptocurrency market has been flooded with alternatives, as of late 2019, 2000 different cryptocurrencies have been added to the market[26]. Bitcoin still holds a 70% share within this market[35]. Such increase in public interest and establishment of a digital currency has attracted the attention of researchers and the industry alike.

Bitcoin is built on-top of blockchain technology[16], this technology allows a ledger to be maintained within a decentralized peer-to-peer(P2P) network. As blockchain supports digital-currencies, known as cryptocurrencies, it must guarantee certain privacy, security and consensus requirements. Guaranteeing these requirements on a distributed systems is done with consensus mechanisms and relies heavily on security provided Cryptographic protocols. Blockchain has an intrinsic trade-off, Figure 1.1 shows the constraints blockchain faces between security, scalability and decentralization[22]. Two of these constraints can be met within an implementation of blockchain but all three is not currently possible. To overcome such trade-offs researchers have had to rethink the way the ledger is utilized.

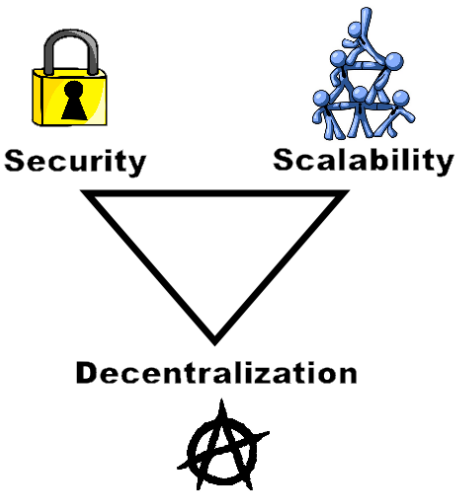

Figure 1.1: Triangle of trade-offs for current blockchain implementations

The blockchain mechanism captures a wide audience due to its ability to guarantee security and decentralization. These two components are the corner stone of the technology, thus a solution needs to be found for scalability while preserving security and decentralization. One of the main issues to scalability is the amount of transactions that can be processed within a certain time-window without breaking the security guarantees. Due to the limitation in processing in transactions, off-chain transactions are proposed. **Off-chain transactions** are transactions that take place between two parties exchanging cryptocurrency without having to publish the specific transaction to the blockchain/ledger due to the use of a smart-contract. Off-chain transactions are also known as off-chain payments and these terms are used interchangeably throughout this thesis. **Smart-contract** is a protocol that can be used in the context of blockchain to digitally facilitate, verify, or enforce certain constraints of a digital-contract[81].

1.1. Utilizing Off-Chain Transactions

Utilizing off-chain transactions reduces the load on the blockchain by reducing the amount of transactions that need to be stored/pushed onto the ledger, allowing for off-chain transactions to be an avenue to overcome the scalability issues of blockchain[27, 45]. Opening a payment channel requires two entries to be pushed to the ledger, an opening and a closing entry. After that all transactions happening between the two parties of the payment channel happen off-chain, thus not requiring each transaction to have an entry in the ledger.

Off-chain payments can be materialized with the use of payment channels that are linked to create a network, known as a Payment Channel Network(PCN), figure 1.2 represents a simplistic visualization of a PCN. Payment channels allow for two users to lock-funds within a channel by pushing a single opening transaction to the ledger. **Locking-funds** refers to depositing money within a smart-contract to open a payment channel so that the money deposited can only be accessed after the closing of the smart-contract. During this process the deposited funds will be redistributed depending on the balance of the payment channel.

These channels can then be used to route payments through the network, without sharing details about the transactions with the ledger until the users decide to close the channel with one final closing transaction that is published to the ledger. Within the Figure 1.2 a node labeled A is transacting with a node labeled B, via the route described in red.

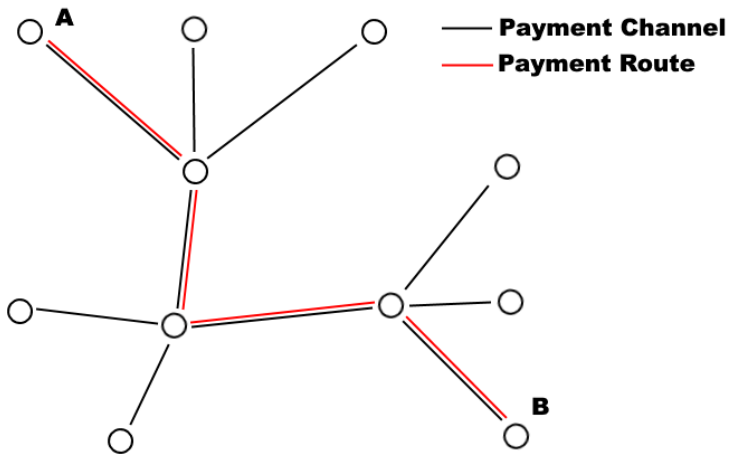

Figure 1.2: Representation of a Payment Channel Network, where the circles represent nodes and the links represent payment channels. The red lines indicate how a route can be made so that node A, is able to transact with node B.

As different implementations of PCNs come forth, the interest in researching, optimizing and understand the mechanics of such networks rises. Lightning is an implementation of a PCN that has been adopted by

bitcoin[53] and Raiden is a PCN network for etherum[54]. The lightning network already boosts more than 5,000 nodes[7], while the ripple network has around 1000 nodes[8]. To optimize PCNs different characteristics of PCN need to be identified and researched. **Success rate**, the amount of transactions a network can handle over a set of transactions, is a good metric to try and optimize. Optimizing success rate can put the privacy goals of the network at risk, thus the trade-offs of such a network need to be categorized and examined. Routing, balancing link-capacities, concurrency and privacy are some of the biggest challenges that are currently being studied for PCNs[45, 64, 65, 68].

A promising implementation of an algorithm used by PCNs that is designed to route, balance and create path-based transactions is SpeedyMurmurs. Results indicate that for certain scenarios SpeedyMurmurs is able to handle changes in the network efficiently while keeping overhead dealing with network changes low, flexible but efficient path selection and multi path-payments allowing for efficient use of link capacities. While achieving value privacy and sender and receiver privacy. **Value privacy**, is a definition for privacy that ensures that no party outside of the two trading parties is able to observe the amount of value transacted between the two parties. **Sender and receiver privacy**, refers to **plausible deniability** , plausible deniability is a privacy definition that ensures that it is impossible to identify the parties transacting with certainty[65].

One of the biggest draw-backs and limiting factors to research in the area of PCNs is the lack of having a true data-set. Due to their privacy mechanisms that are at work, extracting meaningful data from a PCN is difficult. Data can be gathered about the initial locking of funds between a link and the closing of such a link, due to these links being opened via a smart-contract on the ledger. However such data only represents two points in time and no information is stored about the transactions that have made use of the link. Thus vital information like firing rate of transactions is missed, the value of transactions and how often transactions fail due to insufficient link capacity. Without this knowledge of what happened during the opening and closing of a channel, characterizing and analyzing a PCN becomes rather difficult.

1.2. Objective

The objective of this thesis is to design and evaluate a rebalancing algorithm for PCN. The rebalancing algorithm must not be limited by the graph type and implemented along side a path-based transaction algorithm. To be able to achieve these either a path-based transaction algorithm can be designed with rebalancing in mind or an existing algorithm is utilized. SpeedyMurmurs is a good candidate to build upon due to the embedded based routing algorithm and the way it is able to split transactions over multiple paths. The embedded based routing allows SpeedyMurmurs to make privacy and security guarantees, while allowing for the routing to be flexible and adapt quickly to a changing network. The multiple paths for a transaction allow the load of a transaction to be spread over a network, such a mechanism can be used to benefit a rebalancing algorithm.

To evaluate the rebalancing algorithms simulation based experiments will be run. To run these simulations data-sets are needed. Researching a path-based transition algorithm with rebalancing capabilities without knowing the exact workload and how the mechanics of the algorithms influence the network under different workloads is non-trivial. Without being able to monitor and follow the transactions in the network, knowing the state of the network becomes impossible and being able to pinpoint the issues becomes extremely difficult. To overcome these challenges data-sets will be needed to evaluate if an increase in success rate can be found by using rebalancing algorithms.

To formalize the objectives, the objectives are presented as research questions:

1. In the absence of real-world data-sets for modeling a Payment Channel Network, is it possible to generate synthetic data-sets with concurrent transactions that allow for the emulation of a PCN under varrying workloads, to evaluate link rebalancing strategies?

2. For a set of transactions that run through a PCN, can the success ratio of a PCN be increased by adding a monetary incentive tied to the fees of each transaction be used to aid in keeping links balanced?

1.3. Contributions

Rebalancing algorithms should aim to dictate the transactions occurring over the network in such a way that an optimal route can be found for the payment, while trying to optimize the links capacity balance over the possible routes the transactions can take. The algorithms being presented in this paper try to leverage an incentive via monetary incentives. Different fees are offered depending on the state of the link during the transaction. Trying to act as a broker or merchant, selling the use of its links depending on their link capacity

balance. The proposed algorithms are called **Passive Merchant** and **Active Merchant**, where both algorithms change their fee for processing a payment depending on the capacity of the link. The passive merchant does not broadcast any information about the links and fees it is offering. While the active merchant, actively lets other nodes know about the state of its channels.

To help over-come the limited data-sets within this field of research two types of synthetic data-sets models are introduced, one based on **Lightning** and the other based on **Ripple**. The difference in these synthetic data-sets lies in the distributions of link-capacities, the value of a transactions. Different distributions are proposed as there is not one standard for PCNs. With a data-set, even it being a synthetic one, information can be gathered about how a PCN works. Understanding the inner workings of a PCN will allow researchers to scale a PCN, allowing higher throughput without having to push entries to the ledger. Due to blockchains scalability issues, and the proposal of PCNs to be used as a method to increase the scalability of blockchains. Examining throughput as a point of research will give insights into the limiting factor of the throughput in a PCN. An obvious place to start exploring throughput is by trying to examine the effects of depleted links to the overall throughput and how the influence of a rebalancing algorithm may effect overall throughput of the network.

1.4. Project Overview

During this paper background will be given on the challenges that face blockchain and how PCNs can help alleviate some of those challenges, then related work is presented on the current state of PCNs and there transaction routing algorithms. An in-depth look at SpeedyMurmurs is presented, defining its privacy and security guarantees and the mechanics of how the algorithm works on a network.

Following this the Merchant algorithms will be presented and a discussion presented on how these algorithms fit within the current implementation of SpeedyMurmrs and there potential benefits. These algorithms rely on certain core mechanics, they will be identified and discussed along side the parameters that would dictate how these algorithms operate. Before testing can be done on the effectiveness of the Merchants, the generation of the synthetic dataset is discussed. Two use cases for PCNs are examined, the lightning network and the Ripple network. With the limited information available about these PCN an analysis is done and a model is designed from the results to mimic a PCN and their transactions.

A description of the simulation model that was used to create the results is provided in the methodology. Accompanying the simulation model, a description is presented on how the data-sets are generated. Finishing off with the presenting of results and the conclusions for this thesis.

2

Background on Block-chains and Payment Channels

This chapter covers the background information about blockchain technology and how its used within cryptocurrencies and why payment channels are needed.

2.1. Blockchain

Within this section a broad overview of blockchain is described.

2.1.1. Building Blocks of Blockchain

Blockchain technology has been a buzzing topic since 2008, around that time the first white paper was published on the topic[50]. To understand the significance of blockchain it is essential to understand that at its core blockchain is a distributed system. For the first time in the research field of Distributed Systems, a distributed system was able to maintain a secure decentralized ledger without needing a central authority to regulate the system.

At the core of blockchain lies a very simple goal, to maintain a continuously growing list of ordered records. In that sense, blockchain technology is an accounting protocol that works on keeping track of a ledger. A **ledger**, for a cryptocurrency is a recording and totaling of economic transactions measured in terms of a monetary value a business has completed[25, 80]. The ledger is not limited to only storing transactions between two parties, it can also be used store and secure data for government entities or cooperation's. This thesis however focuses mainly on transactions between players and will not look further into how blockchain can be used to store data outside of the scope of cryptocurrencies. Blockchains potential stems from the way the ledger can be used in a distributed system and has been implemented in a decentralized manner. To achieve this, the blockchain protocol has pushed the boundaries of cryptography and cybersecurity[84].

Ledgers within the context of blockchain are made up of *blocks*, these blocks are interconnected due to each new block's containing a *hash pointer* of the previous block, this is illustrated in Figure 2.1. These hash pointers are used to organize and link blocks together. Essentially having the blocks in the chain build on each other, this is known as *Hash Chained Storage*. The hash pointer pointing to the previous block, is created by cryptographically hashing the data of the previous block. Making use of hash pointers allows for public

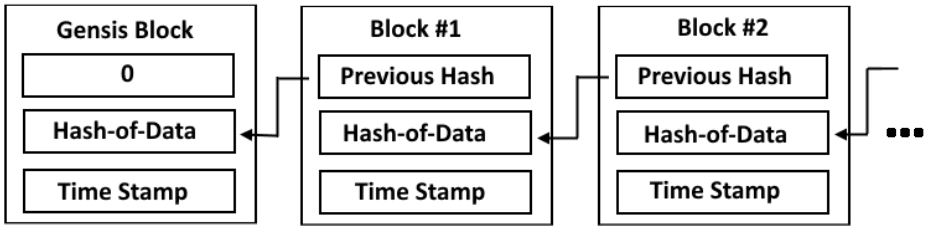

Figure 2.1: Illustration of how the previous hash-pointer is connected in a blockchain data structure.

verification of blocks to prove the stored data was legitimate and not tampered with once it has been placed in the chain. The first block in a blockchain is known as the Genesis block and is a special type of block as it is the only block without a hash-pointer to a previous block.

A hash is created by a cryptographic hash function, also known as one-way functions. These hash functions play an important role in digital security[52]. Two important properties of hash functions are as follows[82]:

- Given hash function h, if one were to calculate $h(x)$ a hash function must not allow for a reverse computation. Thus the value x should not be found from having the $h(x)$.
- Another property of a hash function requires that the hash function h given value x makes it computationally hard to find a value y such that $h(y) = h(x)$ while $y \neq x$.

With the chains data-structure being cryptographically interconnected, tampering with the data after a block has been added to a block-chain becomes extremely difficult, as the data would need to be manipulated in such away that the hash of the data remains constant otherwise the tampering can easily be detected. Adding blocks to a blockchain is dictated by the consensus algorithms that govern the blockchain implementation.

2.1.2. Consensus Algorithms

Consensus algorithms are a group of algorithms that help a distributed system reach a formal consensus even if with the system there are some bad actors/malicious players or processes have become faulty, known as fault tolerance. The classic problem is called the "Byzantine Generals Problem"[58] and the inability to reach a consensus is referred to as the Byzantine fault. For blockchain such an algorithm is vital due to the implementation allowing for decentralization. Not all parties can be trusted in a public blockchain.

Two major consensus algorithms co-exist within blockchain employed consensus algorithms. The **Proof of Work**(POW) and the **Proof of Stake**(PoS) algorithms.

Proof of Work, was first introduced in 1999, the mechanism works by having workers in the network do work[36], working is done by computing a one-way function. Such functions have two very important properties:

- **Fast computation and verification**, computing the one-way function should be fast. Allowing for others to quickly verify the proof in terms of correctness.
- **Preimage resistance**, finding the answer to the solution should not be possible if one has the solution. If the solution to the function is $H(x)$, one should not be able to compute x.

When a worker solves the current hash function, it will generate a block awarding it's self some form of reward, this reward is prescribed. Certain cryptocurrencies have opted to limit the amount of supply of digital currency allowed in the system. In the case of bitcoin, slightly less than 21 million coins can exist. Over time the rewards for mining bitcoin is halved, known as an ̈halving" event. At some point, when enough blocks have been mined, no reward is given in the form of bitcoins, the only incentive to mine blocks will come from the transaction fees fetched. These fees must then offset the costs of mining blocks, which is not cheap due to the energy consumption required to calculate the right answer from the hash function.

Once the work is done and a party has a solution, it can offer the solution to all other parties in the network, with a new block of transactions that will be stored to the blockchain. Other parties in the network will receive the solution and validate the solution, once it is validated they will append the block to their ledger and continue working on solving the next problem.

In the case of Bitcoin a block is mined roughly every 10 minutes, each block has the size of 1-megabyte. Mining blocks periodically like this allows for the system to not be overloaded with new blocks, though it does limit the amount of transactions that can be processed by the blockchain; creating a trade-off between amount of blocks mined per time unit and transaction throughput of the system.

This trade-off of amount of time it takes to mine a block and the amount of transactions that can be processed ensures the system is stable. Mining more blocks frequently creates the vulnerability of denial of service attacks. Making the system less secure, though more frequent blocks would allow for more transactions. Due to this trade-off in the way PoW works, scalability is an inherent issue to using PoW as a mechanism for blockchain.

Security, in the sense of solving the Byzantine Generals problem in PoW is achieved by granting rewards to whoever created/mined the block. Due to the rewards it helps ensure the robustness of PoW consensus

algorithm, without this attraction of multiple players to mine blocks, the chance someone achieves 51% of the mining power is great. More can be found on the 51% attack in 2.1.5.

Proof of Stake, promotes penalties-based solutions instead of reward bases solutions for its security. With a PoW based blockchain, almost anyone can become a miner and solver the cryptographic puzzles and participate in block generation. Within PoS, miners must lock funds, the stake. Once miners have locked funds they have become eligible for becoming validators. Anyone can become a validator in PoS as long as they post a special type of transaction where funds are deposited.

The identify of validators in PoS is known, and the system must keep track of all legitimate validators. A new block can be created by the validators. The set of validators must place a bets on the next block, then all validators takes turns voting for which block will be validated as a new block. Each vote has a weight that is proportional to the stake that the validator has choosen to deposit. A validators probability of creating a block is also proportional to their bet.

In contrast to the PoW, the reward is given to those validators who bet on the right block. This rewards is also proportional to the bet these validators placed on the block. The underlying security in PoS comes in the form of penalties, the cost of reverting transactions may costs thousands of times more than the rewards the validators got from validating blocks during the reverting period. Thus security is done via trying to minimize economic loss, rather then security from rewards for running energy expensive computations like in PoW.

Consensus algorithms are not the only thing that sets blockchain implementations apart from each other. Different types of transaction models also exist, effecting the way the system is able to keep track of transactions between users and how transactions are handled on the blockchain.

2.1.3. Types of Blockchain Transaction Models

The two main different transaction models that have emerged for Blockchain are unspent transaction outputs(UTXO) model[50] used by Bitcoin and an count-based transaction model, that was coined by Ethereum[83].

The UTXO model, is a transaction model that does not create accounts for each user part-taking in the blockchain. Rather in the ledger a list of bitcoin instances can be found that the user has not spent yet. A user can define their balance by the amount of unspent bitcoins in the ledger they have a key too, or are able to access. These keys are usually stored in what is known as a wallet.

For a UTXO transaction to take place three requirements need to be met. Firstly, all unspent instances of bitcoin must be signed by the owner. Secondly, transactions with multiple inputs, multiple unspent instance of bitcoin, must all come from the same owner. Thus having a similar signature. Thirdly a transaction can only be classified as legal if the input value is greater or equal to that of the desired transaction value. An inherent consequence of this implementation means that the history of transactions of a blockchain using UTXO is stored in the ledger.

To get the point across of how UTXO works, an example is given. If James and John want to send Mary some bitcoins. If James were to send 4 bitcoin and John 1 bitcoin, Mary would have two single instances of unspent bitcoin. If Mary would like to store her bitcoin under one key instead of two, she would have to also push a transaction to the blockchain ledger, so that she would have an unspent instance of 5 bitcoin.

Account-Based Online Transaction Model, is a relatively simpler model then the UTXO transaction model. Account-based online transaction model works by explicitly operating all transactions using the account of the sender and receiver. Such a model closely resembles that of a bank account in the brick and mortar banking of today. This model allows for greater space savings due to the need for only one reference and one signature to produce an output.

A transaction has three requirements. Firstly, the transaction needs to be signed by the sender. Secondly the senders ownership of the tokens value can be attested. Thirdly the senders spending account has a high enough balance to send the desired token value.

To get the point across of how account-based online transaction models differ from UTXO, an example is given. Lets take James and John who would like to send some Ethereum tokens to Mary who currently has 0 tokens in her account. If James were to send 4 tokens and John 1 token, Mary would have 5 tokens in her account without having to make an extra transaction combining the tokens as she would of had to do in the UTXO model.

Thus far the discussion about blockchain have assumed that the blockchain is public and fully decentralized. However different types of blockchain technologies can be implemented, depending on the type of blockchain, the consensus algorithms and transaction models may differ due to different security and privacy assumptions.

2.1.4. Classification of Blockchains

Boosting decentralization and allowing anyone to participate in a blockchain network is what captured the attention of the research community and industry alike. Though not all block chains are public. Three different types of categories can be classified for blockchain technology pertaining to the access and construction of the blockchain. These three classifications are:

1. **Public blockchain**, is a blockchain anyone is able to join. A property of a public blockchain allows anyone to participate in the consensus procedure, for the creation and validation of blocks. A public blockchain is fully decentralized.

2. **Consortium blockchain**, is a blockchain that has a pre-selection of participants. Within this sub-group of pre-selected participants a consortium blockchain is able to get certain write permissions and control the consensus procedure. All read functionality of the chain is open to anyone in a consortium blockchain. This may be seen as semi-decentralized.

3. **Private blockchain**, is a blockchain that has restricted read and write of the network to multiple organizations that may not trust each-other, the set of organizations may choose who may participate in the blockchain. Within a private blockchain the trust is centered around the organizations running the blockchain.

Depending on the class of blockchain implementation used, different security and privacy goals needs to be met. As a public blockchain needs stricter security and privacy guarantees compared to its counter-part of a private-blockchain.

2.1.5. Security in Blockchain

To discuss security within the context of blockchain, different adversary models need to be addressed. Requirements for security need to be formed and different attack models need to be considered. Within this section each of these topics is addressed.

Adversary Models

Security in blockchain has been a highly researched field[23, 38, 40, 84]. Being able to discuss security in blockchain and distributed systems one usually makes use of the *Threshold Adversary Model*. Typically within this model the threshold lies between two parameters:

1. All parties participating in a distributed system, n

2. Parties that are acting in an adversarial way, f

The typical threshold in the byzantine agreement case is $n > 3f$[19]. For bitcoin that makes use of the PoW consensus algorithm such an adversary model is less applicable. Due to the way computational power is the driving force of the algorithm instead of individual parties. For this reason the *Computational Threshold Adversary Model* is used[11]. The threshold lies between two parameters:

1. Amount of computational power, C, in the system, N_C

2. Amount of computational power, C, of the adversaries in the system, F_C

Using Bitcoin and the PoW for the threshold for the Computation Threshold Adversary Model is $N_C > 2F_C$[42]. From this threshold the attack model of the 51% attack was coined for Bitcoin. PoW is not the only consensus model for blockchain. For the PoS consensus model a different adversary model needs to be discussed.

The *Stake Threshold Adversary Model* can be used as an adversary model for PoS consensus[11]. The threshold lies between two parameters:

1. Total amount of resources, R, in the system, N_R

2. Total amount of resources, R, that the adversary control, F_R

In ethereum's PoS approach to modeling this adversary model it was found that the threshold is at $N_R > 3F_R$.

Requirements

Well defined requirements are important for blockchain and cryptocurrencies security. Due to the implementation of this technology having monetary value, personal assets of users needs to be protected. To guarantee the security of assests starts by defining security requirements. Zhang *et. al* in their paper "Security and Privacy on Blockchain" define a couple of security requirements for general online transactions, these are listed and reviewed:

- *Consistency of the Ledger across Nodes*, within the process of handling transactions and liquidation between various monetary institutions running nodes. These monetary institutions may have varying underlying processes dealing with transactions, no matter the underlying process inconsistencies between ledgers is not tolerated.

- *Integrity of Transactions*, the system must be able to guarantee that transactions are not to be tampered with. Such that transactions should run without the risk of being altered from initialization of the transaction process to the end of the process.

- *Availability of System and Data*, the system must be online and available to the users storing assets to it. Users must not be hindered by having the system be offline and denying access to personal assets of users. This also extends to being able to transfer and transact whenever required.

- *Prevention of Double-Spending*, digital currency must not be spent twice. A verification process must exist so that currency may not be used for double-spending.

Attack Models

Three of the key attacks for blockchain are described below. Denial-of-service, blockchain tampering and a majority attack cover the basic support of blockchain.

Distributed Denial-of-Service (DDoS) Attack

Denial-of-Service(DoS) attacks refer to an attack where the host of a service is no longer available. Most of the time such an attack is done by overloading the hosts network resources and ensuring that no other party has access to these resources. Distributed Denial-of-Service is trying to achieve a DoS attack on multiple parties in a network, trying to weaken the infrastructure of the network to such a point it can no longer function.

Attacking a blockchain with a DDoS attack may render it unavailable if the blockchain network has a limited amount of participants in the network. As the network grows it becomes more resilient to such attacks as the underlying consensus protocol, and fully decentralized construction and maintenance of the blockchain can be ensured by the still active nodes. In order for a DDoS attack to successfully make a blockchain offline it would need to attack a very large portion of the blockchain nodes, needing an incredible amount of computational resources to achieve this.

Blockchain Tampering

The tamper-resistance of any blockchain means how well it has been designed to be able to detect and resist any tampering of transactions stored in blocks. As discussed in 2.1.1 each block has a hash pointer of the transactions to the previous block. This makes tampering with blocks nearly impossible unless all past and future block pointers are also changed. As every transaction is usually signed and distributed over the whole blockchain network, it becomes practically impossible to tamper with the transaction data without being caught.

51% Attack

This attack is closely linked to the adversary model for PoW consensus models. If an adversary was able to get 51% of the computation power, it could dictate what comes onto the blockchain at all times. This increases the risk for double-spending. Another possible outcome of this attack is changing past transactions as if they never occurred. In the case of Bitcoin such an attack is unlikely due to the enormous amount of computation power that is needed, and doing such an attack may work once. The underlying monetary value of bitcoin relies on the trust of that value by all parties, if bitcoin were attack by such an attack multiple times, the trust in the whole network fails and would most likely mean the end of bitcoin. Thus such an attack is a risk for blockchain however the gains of doing such an attack are short-term.

2.1.6. Privacy in Blockchain

Besides security and privacy are important aspects when it comes to blockchain. Privacy and anonymity are closely related, anonymity of a user refers to the non disclosure of a users identity. While privacy, confidentiality of transactions, should allow users to make transactions without disclosing all the details to the public. Most cryptocurrencies have a public ledger, meaning all transactions are recorded and can be accessed publicly. While this method ensures security and allows any node to join a network, from a privacy perspective this is counter intuitive.

Examining privacy and anonymity can best be done by describing how users can join a blockchain. When an user joins a blockchain, a public and private key are generated for them. These keys are a set length of random characters that cryptographically linked. An assumption that is made is that it is impossible to guess another users private key given they have their public key. A wallet or address is created by putting the public key through a hash function, this wallet value can be shared with other users so they can exchange currency.

With the creation of public and private keys, that are not shared amount all users. Anonymity seems to be very plausible. Gathering personal information from these keys is not possible. A point of weakness in anonymity is the underlying infrastructure of the internet. The non-anonymity of the internet risks spoiling the anonymity on the blockchain, an example of this would be an user using a third-party exchange platform and having to verify his identify. Now the third-party has an identify matched to a public and private key, if these party was compromised then the identity and users keys could be leaked to the public.

Privacy as mention seems to be more counter intuitive due to most ledgers being public, and ledgers store a list of all transactions that have occurred. If identities are linked to wallet addresses, privacy is broken and one would be able to see who made transactions with who. Users are able to create any number of anonymous wallets boosting the anonymity and privacy of each user. Allowing users to create any number of wallets, can help mask the type of transactions happening. This is done by having a user use multiple wallets for a transaction, if these multiple wallets cannot be identified to the user. It will be impossible to link the multiple transactions together to see the full exchange between users. Allowing for privacy on the blockchain.

As mentioned the underlying infrastructure of the internet may ruin privacy and anonymity. Researchers have used different techniques to highlight weak points of privacy in a blockchain. The two main techniques are analyzing the blockchain and analyzing the current traffic on the blockchain. Analyzing traffic has allowed Koshy *et alter* to match an IP with a Bitcoin address[41]. Analyzing the blockchain is a more common way of removing anonymity from Bitcoin. Spagnuolo *et alter* were able to estimate the amount of Bitcoin transferred to ransomware software[69]. Androulaki1 *et alter* were able to recover 40% of users profiles even after adopting privacy measures suggested by Bitcoin[12].

Security and privacy goals can be described in one corner of the blockchain triangle, along side decentralization and scalability as seen in Figure 1.1. Maintaining security and privacy goals why allowing for a scale-able decentralized solutions seems to be an issue. While decentralization can be adjusted depending on the classification of the blockchain implementation. On a public decentralized blockchain security and privacy seems to be aspects that cannot be loosened, thus the scalability of the blockchain becomes the point in were the biggest trade-off needs to be made.

2.1.7. Scalability by Increasing Blocksize

Scalability within cryptocurrency comes with trade-offs to the security and transaction throughput of the system. Creating a system that can scale to handle global demand with high transaction throughput, however in practice the implementation is not trivial and brings forth risks in the integrity of the technology. Taking Bitcoin as an example, for Bitcoin to be able to compete on a global scale its throughput would need to at least sustain the same amount of capacity of current competing technologies. Taking VISA as an example with Visanet, Visanet being VISA's massive global credit payment network.

Currently Bitcoin is limited to 7 transactions per second(tps), significantly slower than the 2000 - 4000 tps VISA is averaging on a normal day. During the December holidays VISA's peak capacity lies much higher, it was recorded to be around 56,000 tps in 2015 [60, 72, 75]. Would Bitcoin want to compete with visanet it would need to significantly increase the capacity of transactions it can handle. Not an insignificant amount of increase in capacity by any stretch of the imagination.

Poon and Dryja have demonstrated for Bitcoin to be able to compete with Visa's payment network. Bitcoin's blockchain block size would need to increase from the current 1 megabyte to 8 gigabytes, if a block were still to be mined every 10 minutes[60]. Over the span of a year this would accumulate to 400 terabytes of data per year. Dealing with such massive amounts of data means that smaller non-industrialized computer

systems would not be able to handle the load.

Raising the concern of the centralization of cryptocurrencies, as only big data-centers would be able to handle such a type of workload. Going against the very concept of having a decentralized currency. Gervais shows in his work that such a big block-size is not possible[34] from a security stand-point. Doubling down on the claim that scalability for blockchain is not a novel task. Ensuring the security of a blockchain is a very high priority. If cryptocurrencies are not able to handle higher transaction throughput's, the technology may suffer as it is out competed by other payment solutions. For this reason different protocols are being researched that will work on-top of the current cryptocurrency solutions.

Increasing the block-chain block size to such an extend and the centralization of Bitcoin also poses security risks as with enough mining power one could rewrite and remove transactions from the blockchain. Bitcoin's security depends on the PoW-based distribution consensus protocol, that the miners are responsible for. If miners were to acquire the majority of the computation power in the network, a so-called 51% attack could be done on the blockchain[29, 30]. Allowing the attacker to remove and rewrite transactions on the blockchain. Such attacks poses more of a risk if the distribution of mining power is done only over a couple of data-centers, instead of a much broader network. As all transactions are stored on a public ledger, security and privacy risk are also a concern for individuals using the blockchain.

2.1.8. Scalability with a Layer-2 Protocol

Due to the risks to security by increasing the block-size of a block in a blockchain. Researchers propose to use *layer-2 protocols*, to aid in combating the issues that come with blockchain. A layer-2 protocol within the context of blockchain is implemented on top of the layer-1 protocol, the blockchain or ledger, by either creating a smart-contract or using opening and closing transactions. Opening and closing transactions is discussed in more detail in Section 2.2.1.

The layer-2 protocols make-use of the layer-1 trust and security assumptions. Most layer-2 protocols allow users to perform off-chain transactions through either state channels and side chains. Off-chain transactions are a type of transaction that exchange/transact cryptocurrency without storing the exchange on the ledger. Due to this transaction taking place outside of the public ledger, it is known as an off-chain transaction. Off-chain transactions provide a way to scale blockchain, without having to increase block-size or in the case of PoW reduce the computation time of new blocks to become a security issue. This thesis focuses on state channels, payment channels.

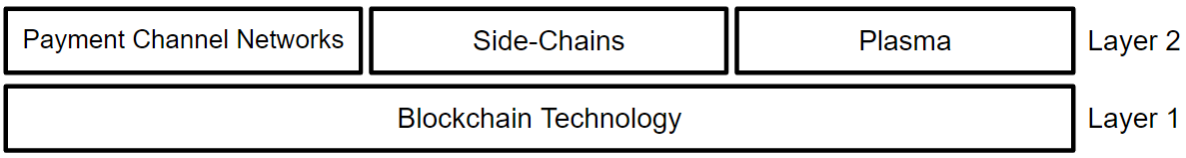

Figure 2.2: Layer-2 protocols that work on blockchain

Allowing cryptocurrencies to make use of one or more layer-2 protocols seems to be the way forward for blockchain scalability. These layer-2 protocols allow for users to make use of different solutions so that a higher throughput of transactions can be achieved. With the benefit of reducing the amount of transactions pushed to the blockchain, reducing the need to increase block-sizes of current cryptocurrencies.

2.2. Payment Channels and Payment Channel Networks

Within this section a broad overview of the workings of payment channels and payment channel networks is given.

2.2.1. Payment Channel

Payment channels, first coined as micro-payment channels[3] allow users the ability to make off-chain transactions. This technique was first implemented in Bitcoin 0.1[2] however the design first used was not secure due to one party being able to collude with a miner to puts the final version of the payment channel without the consent of the other user. Allowing for the possibility of stealing funds. Different types of payment channel protocols have been proposed. However the focus will be on the Poon-Dryja proposed protocol for payment channels while introducing the Lightnining Network[60]. Some properties proposed by Poon-Dryja was that the channels could have an indefinite lifetime, allowing for an infinite amount of transactions to

occur on a channel without having to make use of the blockchain. Another property was that the channels were bi-directional.

Opening and Closing Transactions, can be done by pushing a special type of transaction to the blockchain. Though the specific name for such a transaction may be different depending on the implementation, here openChannel and closeChannel represent such an transaction. Within the proposal for payment channels, a 2-of-2 multi-signature was suggested to lock the funds in the chain. A 2-of-2 multi-signature refers to a signature protocol where both parities create a digital signature before the funds can be locked. Such signatures also exist in the form of 1-of-2 where two parties have a signature but only one is needed or a 2-of-3 situation where three parties have a signature, one signature is always necessary and only one of the other two need to sign.

An openChannel operation opens a payment channel between two nodes via a transaction. This transaction is stored on the blockchain with an initial deposit of Bitcoin or other currency, with a couple of extra key pieces of information. The two nodes need to record their Bitcoin addresses, initial capacity of the link in both directions, channel timeout, the fee charged for using the channel and a channel identifier. Once the 2-of-2 multisig is established the channel has opened.

A closeChannel operation closes a payment channel between two nodes via a transaction on the blockchain. The two nodes closing their channel are able to close the channel by defining the state of the capacity of the link and updating their Bitcoin balances. Closing a channel can be done by either one party or both parties. If a channel is closed unilaterally the funds of the party requesting to close the channel are temporarily locked for a period of time, allowing the other party to dispute the state transmitted by the closing party.

Dispute handling, is done outside of payment channels. If any disputes occur either due to malicious behaviour, or other factors like timing-out a Hashed Time Lock Contract(HTLC) then a *dispute handler* needs to find a resolution. When parities close a channel the dispute handler is also opened for a fixed time. During this fixed time both parties can send evidence to the dispute handler. On the basis of the signed messages provided and their last-known balance the dispute handler is able to compute and disburse the money accordingly. Most often the evidence with highest round number wins, though a verification process takes place to ensure no false evidence is presented.

2.2.2. Payment Channel Networks

A node is not limited to establishing a payment channel with only one other party allowing for nodes to connect with a multitude of nodes. Once multiple nodes join together with payment channels a *Payment channel network*(PCN) is created. An incentive for nodes to join a PCN is that it allows nodes to transact with nodes in the network, even though no direct payment channel link has been established. This gives the flexibility to nodes to only establish links with nodes they are willing to trust.

Nodes can forward transactions between their direct neighbours, so the network is able to make multi-hop transactions. A multi-hop transaction is a transaction that spans over multiple different nodes between two users, the nodes that participate in the transaction not as the receiver or sender node are known as forwarding nodes. It is common to collect fees for forwarding transactions, thus creating an incentive to do so. Off-chain transactions are a great advantage for payment channels from a privacy perspective, seeing as this enables private transactions that cannot be traced once they have completed. The tracking of transactions can only be done in a very limited manner by examining the opening and closing transactions to the blockchain, where the difference in capacity in each link could be monitored. Though thousands of transactions could happen before the payment channel is closed, only the difference between opening and closing the link can be examined on the blockchain.

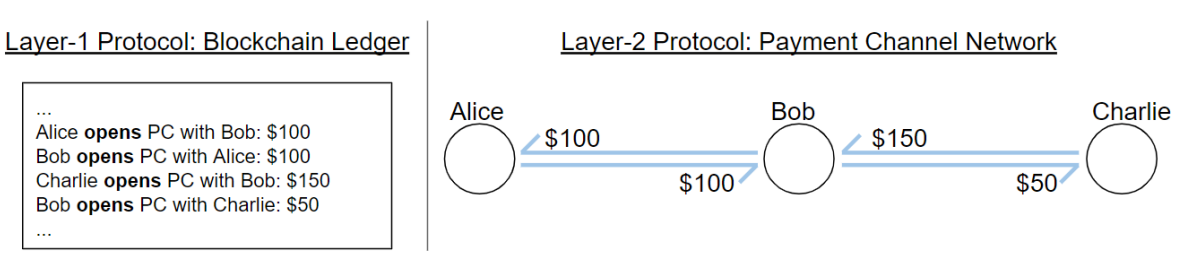

Figure 2.3: Alice, Bob and Charlie have created a Payment Channel Network

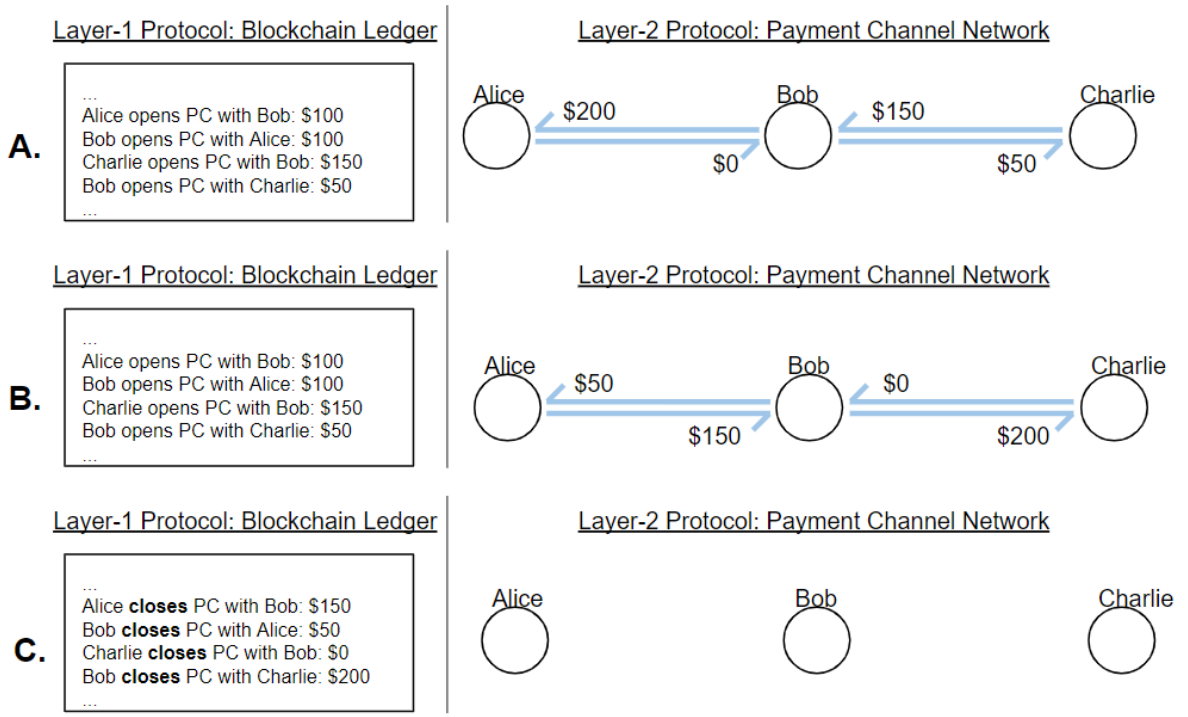

Figure 2.4: A. Alice and Bob transact $150 with one another B. Charlie transacts with Alice for $150 C. PCN is closed and stored on blockchain

To illustrate how a PCN works imagine that Alice, Bob and Charlie come together to create a small PCN. Alice is only able to set-up a payment channel with Bob where the capacity on the line is $100 in both ways. Charlie is only able to set-up a payment channel with Bob, they agree on having a capacity in the direction of Bob with $150 and the direction from Bob to Charlie will have $50. The PCN is illustrated in figure 2.3 also showing the transactions that would be placed on the blockchain in the Layer-1 Protocol.

Alice decides to buy a Duck from Bob for $100. Alice makes the transaction to Bob and in figure 2.4 it can be seen in **A** how the state of the channel changes of the layer-2 protocol. However nothing is appended to the blockchain ledger. Charlie is jealous of the Duck that Alice bought from Bob and offers to buy the Duck for $150. To make this transaction happen Charlie needs to do a multi-hop transaction, where Bob is acting as a forwarding node. In this example no fees are taken from forwarding nodes however usually such fees are taken as an incentive. The final state of the PCN can be seen in figure 2.4 at **B**. After these transactions Alice, Bob and Charlie decide to close the channels and the final transaction is made to the blockchain ledger as seen in **C**.

Important properties of payment channels and PCNs are described below:

- **Conservation of Capacity**, between two nodes that have established a payment-channel the total amount of capacity that is established between the two nodes in the opening of the channel on the blockchain also has to equal the total amount of capacity when the payment-channel is closed.

- **Transaction Connectivity**, a node is only able to transact with another node if it is directly or in-directly connected with a node. Two nodes are in-directly connected when one or more intermediary nodes need to be used to establish a connection. If no path can be found between two nodes, then the two nodes are unable to transact on the PCN.

- **Transaction Capacity**, if a node does not have enough capacity on its link to complete a transaction. The node without enough capacity is not able to take part in the transaction and must notify nodes trying to complete the transaction it is not able to be part of the transaction.

2.2.3. Transactions

Transactions within a PCN require multiple protocols to come together. Simply put a transaction requires a route through the network, a payment protocol to ensure that the transaction is done successfully and securely. Also the network needs to establish some sort of accountability. As Roos *et alter* pointed out path-based transactions(PBT) are built on three key algorithms, namely: *routing*, *payment* and *accountability*[65]. The specfic tasks for these three algorithms are described as follows:

- **Routing algorithm**: finds and establishes the path that transactions will take from the sender to the receiver. Characterized by effectiveness, efficiency, scalability and privacy [65].
- **Payment algorithm**: mediates the funds between two parties and all the connecting nodes connecting the two parties. Characterized by robustness and trust.
- **Accountability algorithm**: resolves any disputes in the presence of malicious behaviour in the network. Characterized by robustness, scalability and provability[37].

Within the context of this paper the accountability algorithm is abstracted away. Within the assumption that no nodes behave maliciously.

These three algorithms establish the basis for a functioning PCN. Where the routing algorithm for off-chain routing has been a point of interest since the draft of the lightning network in 2016[60]. Ensuring secure payments has a slew of different options

Routing Algorithm

The purpose of the routing algorithm in a PCN is to find a route the transaction can take. Routes for payments should ideally be as short as possible as every extra hop will mean the sender will be paying an extra transaction fee. However not all routes are equal due to the capacity of links, if a link has become directional then transaction routes going against the directional link may not use that link. Thus routes in a network need to consider the link capacities and not just the links themselves when deciding on a route.

Many different types of path finding algorithms exist and are applicable to PCNs, though the best option depends on how the PCN is implemented. If nodes know the whole topology of the the network then a flow algorithm may be feasible. Such algorithms have high complexity and as networks scale may no longer be useful. If nodes have no idea about the topology and only have only local knowledge then tree-routing or simply broadcasting the transaction may be applicable. Broadcasting a transaction may overload a network in message complexity and may reduce the privacy and security aspects of a PCN. In Section 3.1.1 different types of implemented routing algorithms are discussed.

Payment Algorithm

The payment algorithm helps secure the payment from the sender through the intermediary nodes to the receiver. Different proposals have been given for such an algorithm. Within the context of this paper, *Hashed Time-Locked Contracts* will be used as the primary payment algorithm.

Hashed Time-Locked Contracts [27, 78, 79] are the backbone to most PBT currently implemented, with some modifications.

To explain how HTLC works, a sender node S and a receiver node R will need to connect along a path. S is sending Bitcoins, B, along the path to R. Assuming for this case that S and R are not directly linked to one another. The path, p, between S and R is made with

$$p = (u_0, u_1, ..., u_{n-1}, u_n) \; where \; S = u_0 \; and \; R = u_n$$

Assuming the path has enough funds to allow for the transaction to happen and no concurrent transactions are taking place along p. S is able to send its B along p. To ensure that B is not lost along the path, a HTLC is established. R can only claim the Bitcoin if it can provide a pre-image of a cryptographic hash. Each HTLC is initiated with a time t, after this time has passed the HTLC is closed and the Bitcoins on the channel can no longer be claimed. Once R receives a notice that S would like to make a payment, R makes a random key Y. Y is put through a hash function, $h = H(Y)$. R sends h to S. With this hash-value S is able to set-up a contract with its neighbour in p, S will set-up a HTLC with u_1 that is dependent on Y, and will expire after a curtain time t. u_1 will do the same with its neighbor in p and so on till every node has an HTLC with its neigbouring nodes in the path. The timer is usually decreased along the path with HTLC contracts, ensuring that no node along the path can expire HTLC before R collects its Bitcoin, or lets u_{n-1} expire. With the cryptography locks in HTLC relatively secure transactions can be made.

Accountability Algorithm

The accountability algorithm ensures that if any disputes occur in the system that these can be resolved. A dispute may entail if a transaction fails one a HTLC contract has been created, or one node would like to close a channel and the other node has not signed off on it. Some PCN rely on the blockchain to resolve any disputes, if this is the case then the Network does not need to implement its own accountability algorithm. Ripple is a cryptocurrency that works on a PCN, the PCN is not a second layer protocol and thus makes use of an accountability algorithm to ensure all disputes can be managed.

2.2.4. Concurrency

Concurrency refers to having multiple transactions in a network happening at the same time. With the consensus algorithm, like PoW in bitcoin, the miner creating the block is able to fetch from a pool of concurrent transactions certain transactions. The subset of transactions chosen to go into the block may be related to the transactions willingness to pay fees and will be ranked accordingly.

Within a PCN transactions are not picked up by the consensus algorithm and users may not be able to avoid issues that arise from concurrency due to payments involving intermediary nodes. If a link does not have enough capacity, and an user is unable to avoid this link they will have to wait for an alternative path or till the balance is restored. Either no transactions are able to get through the network or transactions can get stuck in a deadlock while they wait for each-others transactions to finish.

To help identify certain concurrency issues within a PCN, three common types of concurrency protocols are used. These protocols are *Blocking*, *Non-Blocking* and *Partial-Blocking*protocols.

Blocking Payments

An effort to avoid deadlocks in PCN blocking payments are suggested. In blocking payments channels are fully blocked during a transaction. Allowing no other user to make use of this channel till the transaction has completed or failed. A way blocking payments is able to avoid deadlocks is by having payments fail, seeing as aborted payments do not affect the balance of involved users. Then transactions can be tried again after a random period of time to ensure that if two channels were blocking one another they will not try and transact over the same path at the same time.

A blocking payment protocol is simple, and easy to implement. An issue with such a protocol is that if the transaction volume increases, the chance of two transactions blocking each other and failing will increase. As more transactions try to complete in the network, the network may no longer be able to find suitable paths that are not blocked.

Partial-Blocking Payments

During partial-blocking payments the capacity on the link that is required by the transaction is blocked apposed to the whole link. This strategy allows multiple transaction to transverse over a link. By not having such a binary open or closed link transactions are more likely to complete. Though no guarantee can be given.

Non-Blocking Payments

During a non-blocking payment it should be guarantee that at least one of a set of concurrent payments is able to complete. This can be done with queuing or trying to create a global order of payments. Then the payment that either has the highest or lowest identifier can be pushed through the network. This means all channels are open for multiple transactions, though as more transactions enter the system queuing may result in very slow processing.

2.2.5. Deadlock Detection

Currently most deadlock detection in a PCN is done by letting the HLTC run out of its operational time-frame, then re-trying the transaction at a later time. This is not an optimal solution, and may lead to issues when high capacity is expected from a PCN.

Within distributed systems deadlock detection has been a topic of discussion at least since 1983 two different models were developed by Chandy *et alter*[20]. These models focused on finding a deadlock and having nodes release resources if one was found. Such methodology could also be implemented in a PCN if done carefully.

This may bring up security risks, and privacy risks as nodes will be probing around the network. However if the protocol is done probably and the assumption that most nodes are acting in the best interest of the network it could be conceivable.

2.2.6. Rebalancing

Within PCNs links that are bidirectional allow transactions to come from either side of the link. Once a bidirectional link becomes directional that link is no longer available for certain transactions. This may cause the PCN to become locked as no more transactions are able to find paths with funds in the direction that is needed. For this reason re-balancing is an important topic for PCN research. Within this thesis different re-balancing strategies will be analyzed.

To allow a high throughput of a network concurrent transactions must be able to be handled by the network. This becomes more of an issue when link balances become unbalanced and directional. For this reason finding optimized re balancing solutions may help aid in keeping PCN transactions going through the network. Some networks operate by have nodes replenish funds during operation with one another to ensure that transactions can continue. This works great in a more centralized PCN where one party may have multiple nodes in the PCN. Thus no monetary value is lost when the re-balancing occurs, as a close and open-transaction are pushed to the blockchain. However for fully decentralized PCN's re-balancing with the chain may not be a viable option.

2.2.7. Topology

Due to the relatively new research area for PCNs, the networking and topology side of this research area has been limited. The lightning network depending on which resource is used either has 10,150 nodes with 35,175 channels[7] as state by `"1ML.com"` or the network has almost 5000 nodes with 27,711 channels according to `"Bitcoinvisuals.com"`[44] , while ripples network is totaling 1030 nodes[8].

The discrepancies on information about the lightning network make it difficult to quantify how the lightning network is doing and how many nodes are actually participating in the network.

Zombie Nodes

Something that has not been discussed in any literature on the topic of PCNs is the existence of Zombie Nodes. Seres *et. al* never made any mention of zombie nodes during their analysis of the lightning network[67]. **Zombie Nodes** can be classified as nodes that are connected to the network and have open channels, though have not sent any *channel_updates* or *node_announcement* for a couple of weeks. These nodes may hinder routing as it is unknown whether routes can pass over their channels.

As of December 2019, `"hashxp.org/lightning/node/"` suggests that their are 3054 active nodes, while their being another 11815 zombie nodes in the network. While all sources can agree on the amount of BTC in the system, the node count is a defendant discrepancy between sources. Hashxp suggests that the total zombie nodes collectively have 36.65 BTC on their channels, suggesting that this problem is relatively small compared to the active 815 BTC the live nodes are working with. To be able to more accurately describe a Lightning topology data would need to be collected on the collection of zombie nodes. Research would need to inquire about how they effect routing algorithms, do networks need to employ a type of heart-beat to ensure that zombie nodes do not stay part of the network.

2.2.8. Security and Privacy

If a PCN is built ontop of a blockchain, then the PCN can assume that the security guarantees of the blockchain hold.

Two notions of security and two on privacy have been presented by Malavolta *et. al* in their paper *Concurrency and Privacy with Payment-Channel Networks*[45]. These formulated notions are:

- *Balance Security*, an intermediate note part-taking in a transaction may not lose any coins throughout the transaction. Even if during the transaction all parties involved are malicious.
- *Serializability*[55], All executions of a PCN must be serializable, allowing for every concurrent execution there must exist an comparable sequential execution.
- *Value privacy*, should guarantee that the transaction value is not leaked to users outside of the payment path, as long as the users in the path are honest and trusted users.
- *sender/receiver anonymity*, should guarantee that the sender and receiver sending a transaction along the path with at least one honest user. Corrupted intermediate users should not be able to determine the pair of sender and receiver for a given transaction.

3

Related Work Concerning Payment Channels and An Introduction to SpeedyMurmurs

Within this section related work will be explored for PCN in general.

3.1. Payment Channel Networks

PCN are a new research area, however some of the issues and concepts overlap with credit payment networks like that of VISA. Looking at previous work done on these networks, and comparing issues faced between decentralized PBT and credit payment networks allows for previous found solutions to be adopted by PCN[21]. Malavolta *et alter* identified note-able issues that cross-over from credit payment networks that decentralized PBT will face. The highlighted issues are liquidity, network formation, routing scalability, concurrency and privacy[45]. These issues are non-trivial to solve, requiring dedicated research to come up with viable solutions.

Currently different cryptocurrency solutions are looking to PCN. The Lightning Network is based on Bitcoin[60], and is currently the biggest second layer protocol that helps relieve transactions from the blockchain, the network as writing this thesis has just shy of 5000 nodes[7] with 27,000 edges. Raiden is the PCN that has been built on-top of Euthereum[83], Euthereum is the second largest cryptocurrency according to CoinMarketCap[4]. The Raiden network currently has 35 nodes and the network is depicted in Figure 3.1.

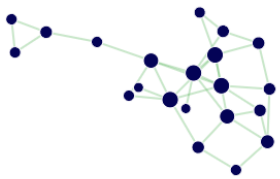

Figure 3.1: Graph of Raiden Network December 2019

As one can imagine not every PCN is getting as much traction as Lightning. Seeing as Raiden has a lack luster amount of nodes in there network. Ripple is another cryptocurrency relying on PCN[66]. While the consensus algorithm for Ripple does not rely on a blockchain but is built into the PCN, it has a much bigger audience than both the Lightning Network and Raiden Network combined. Currently there is about 8.9 billion dollars locked into the network compared to the 4 million of lightning[4]. Showing that there is definite promise in Payment Channel Networks, being it the stand alone solution for a cryptocurrency or a second layer protocol.

3.1.1. Existing Routing Algorithms for PCN

Offchain Routing

Offchain routing is an integral part to any PBT in a PCN. Emphasis is laid on the routing algorithm for numerous reasons. Allowing high-throughput and low latency is highly dependent on the routing algorithm. Two connecting nodes should find the shortest path between themselves to establish a quick transaction. A caveat being in PCN that link capacities change dynamically and need to be balanced in a bidirectionally to allow for high success rates of transactions. Otherwise links may become uni-directional limiting the success for transactions trying to cross over the link in the direction without any capacity.

Adaptive vs Non-Adaptive Routing

Routing has been classically done in two ways, either adaptively or non-adaptively. Both these types of routing have its advantages and disadvantages. For adaptive routing, a lot of overhead is created, though for a dynamically changing system like a PCN it can create more optimal solutions. Non-adapative also known as static routing allows guarantee of reachability, best used in networks with an abundant capacity. However static routing may not always be suited for dynamically changing networks, due to it creating routes based on the knowledge of link capacities.

Numerous different off-chain routing solutions have been proposed that make use of dynamic or static routing. These solutions can be characterized as: **Landmark routing**, **Beacon routing**, **Embedding-based routing**, **Partial Max-Flow and Routing tables**, **Centralized routing** and **Max-flow routing**. Different algorithms for each category will be reviewed below.

Landmark Routing

Landmark is a static routing algorithm and is done by selecting certain *landmark* nodes in a network that store routing tables. All other nodes that are not a landmark route connect to the landmarks[73]. SilentWhispers[48] makes use of the landmark routing algorithm. One of the biggest drawbacks to this routing technique is that if two nodes happen to be in the same branch all transaction will be lead via the landmarks. Possibly leading to performance issues.

Beacon Routing

Flare[61] proposes a beacon routing algorithm. Each node tracks its neighbours to a curtain maximum hop-count k, this forms a k-neighbourhood that each node needs to track. Each node also connects with nearby *beacon* nodes that maintain paths to other beacon nodes. A downside to this algorithm is that nodes update their k-neighbourhood with each credit change. Resulting in inefficient ways to deal with messages, reducing the amount of transactions possible and may result in privacy breaches.

Embedding-based Routing

Embedding-based routing also known as distance-based routing is classically a static routing algorithm that makes use of a vector embedding for each node. The vector is generated in such a way that for nodes that lie close to one another in the network with short hop distances, are also close in embedded space. Each node relays each transaction to the neighbor whose embedding is closest to the destination's embedding. This type of routing is used by SpeedyMurmurs[65], inspired by the VOUTE system[64].

SpeedyMurmurs was designed with privacy in-mind, making sure that sensitive information is not leaked throughout the system. A disadvantage of embedding-based routing is that over time the weights of the links and topology of the graph changes. Meaning routes may have to be recalculated over time.

An interesting aspect to combining embedded-base routing with AMP Payments, is it allows nodes to try multiple paths for different spanning trees to find a route that is available for a transaction. Though the number of routes available can be greater than the number of routes AMP Payments need. Allowing a node to probe more routes to find a suitable one. This type of dynamic use of paths could lead to a higher throughput of the system, seeing as lots of different paths can lead to a receiving node.

Partial Max-Flow and Routing tables

Flash[77] is an algorithm that combines both max-flow and landmark routing into its routing algorithm. Meaning that Flash operates both as a static and dynamic routing algorithm. The algorithm distinguishes between elephant payments and mice payments, as one can imagine elephant payments are large and fairly rare transactions on the PCN. While mice payments are classified as smaller transactions that occur continually.

For elephant payments, the payments are split into chunks and a paths are calculated using a modified max-flow algorithm. The issue of sending elephant payments lies in trying to find enough capacity in a dynamically changing environment. Distributing the calculation gives the most accurate result given capacity. Though the overhead created and latency induced by having a distributed max-flow calculation will make these transactions non-feasible. To get around this issue, the max-flow calculation is done locally. A node

probes the capacity of the network to gather information about the link capacities. Then run a modified algorithm based off of Edmonds-Karp[24] max-flow algorithm.

Mice payments make use of a routing table. Trial and error is used to find a path that is able to handle the capacity of the transaction, till it finds a path from the table that will work. The routing table is updated each time the topology of the network is updated.

Centralized Routing

Centralized routing centers around a central trusted server in a PCN. The central server has a map of all linked nodes in the network and will calculate a shortest path for each transaction on the network. All nodes will need to communicate with a central server before being able to complete the transaction.

Canal[76] implements centralize routing in an efficient manner with tree-only routing, as the central computer constantly updates the spanning-tree. Constant recalculation of the spanning-tree is done to make sure the weights of the links stay balanced in the system. The central computer also processes the shortest-path over the tree between the sender and receiver. Due to having a central server their is a high potential for security breaches and scalability of the system becomes an issue.

PrivPay[47] also implements centralized routing similar to that of Canal. Though this algorithm lays higher focus on value privacy and transaction privacy for the sender and receiver. Still this routing algorithm has a central-point of breach making it a security risk.

Max-flow routing

The lightning network draft routing algorithm falls into the max-flow routing category[60]. This can partially be attributed to its critical benchmark, this benchmark was to achieve maximum success rate of transactions. To accomplish this each node stored a routing table of the entire network and the PCN would run a distributed implementation of the Ford-Fulkerson method to find a path through the network that supported the transaction volume.

The proposed algorithm is an almost ideal solution to the problem, however the biggest draw back to this algorithm is its overhead. The overhead is created by having to calculate the max-flow routing for each transaction with the Ford-Fulkerson method is linked to the number of nodes in the system. Meaning the computation per transaction limits high throughput and the computation time will increase as the network scales.

Routing the path through a network is not the only concern routing algorithms have to deal with. As PCN networks take on more transactions, transaction will start to interact with one another. Due to transactions having to deal with capacity of links and other transactions, links that were once thought to be available could be used by another transaction. Bringing up the issue of how to deal with concurrency in PCN.

3.1.2. Concurrency: Blocking and Non-Blocking Protocols

Flugor

Flugor[45] was designed with the assumptions stated below, it also makes use of three operations: *openChannel*, *closeChannel* and *pay*.

Assumptions that Flugor and Rayo were built on

Both Flugor and Rayo were built upon a list of assumptions of the system. These assumptions are as follows:

1. Every node in the PCN is aware of the topology of the whole graph, including every pair of connected payment channels between nodes. This can be trivially obtained due to examining the transactions on the blockchain and inspecting which transactions are opening a payment channel.[62].
2. Every node chooses a transaction path to its receiver via its own criteria.
3. The current capacity of each payment channel is not published, though it is shared locally with other nodes as otherwise privacy is trivially broken.
4. Nodes are aware of transaction fees along the paths that are chosen.
5. Pairs of users communicate through secure and authenticated channels.
6. Sender and receiver can communicate via a secure and direct channel.

7. The sender of the payment is able to create an anonymous payment channel with each intermediate user

8. *Bounded synchronous communication setting*[13] is implemented in the system with loosely synchronized clocks among users.

9. Total order among the users.

A pay operation transfers money from one node to another. Flugor makes use of the blocking mechanic. For a node to send a transaction, it first calculates if it has enough funds to do the transaction. Calculating this is done by adding all the fees along the path to the receiver plus the amount that was intended to be sent. If the links balance is enough to cover the costs of transaction the transaction is initialized. The sender node sets up the a HTLC with each node along the path to the receiver and with the receiver. Within section 2.2.3 HTLC is explained in greater detail.

HTLC allows nodes to verify that the connection that is being established is non-malicious, this each hop between sender and receiver is able to verify the Hash-lock without revealing the full payment. A time frame is added to each HTLC allowing the contract to be valid during this time frame. Once the receiver pulls the transaction amount it releases, the HTLC contract and this propagates down the path till the sender node is notified.

Each intermediate nodes checks to see if the link capacity that is requested has enough funds to complete the payment. Once the payment is released by the receiver, this release propagates down the path. If any node aborts the transaction the receiver does not release the condition seeing as no payment is made. The rest of the nodes will also release the transaction after the timeout is reached and the HTLC contract is voided.

To deal with deadlocks the solution that is offered consists of letting payments fail. By allowing the timeout of the HTLC contracts. The sender will choose a random waiting period and try again once this timer runs out.

Suggested Modifications for Flugor

Modifications can be made to the original Flugor to guarantee progress as suggest by Werman *et alter* [78]. The proposed idea is to have each edge have an unique ID. It is assumed that the path from the sender and receiver has a communication line traversing multiple edges. To initiate the transaction the requests for payment will be send to the nodes in lexicographic order according to the edges ID's. Once all edges accept the HTLC is established to let the transaction commence.

For this to work, either a node will need to send a probe message to collect id's. Otherwise it would need to be constantly aware of changing edges and the new id's that can be found in the topology of the network.

Rayo

Rayo[45] is a non-blocking concurrency algorithm that also makes use of Multi-Hop HTLC. To achieve this a *channel-state* is defined, that works as a type of queue. A channel-state consists of three parts, an array, *cur*, that denotes the payments currently using part of the capacity. An array, *Q*, that denotes payments waiting for available capacity on the payment channel and a value *cap* that denotes the current capacity on the payment channel.

During a pay transaction if there is enough capacity in the intermediate node to allow for the payment to be made, then the current transaction is stored in the cur queue as an in-flight payment. When the payment channel is saturated, and not enough link capacity is available for a transaction, the transaction may be put in the Q. Figuring out if a transaction is stored in the Q, the value of the transaction ID must be higher than that of current transaction IDs in the *cur* array. If a current transaction in the cur is aborted then a transaction stored in Q will be recovered if the transaction ID is higher than that off all the current transaction ID's in the cur array. If the transaction ID is lower when its taken from the Q it is automatically aborted.

Queuing is an interesting aspect of a non-blocking algorithm. Rayo makes use of the transaction ID to make decisions for determining if a transaction is worth queuing and if queued whether or not the transaction is still worth pursuing once capacity is available. Queuing can also be done in a different manner, instead of taking a FIFO approach the queue can be searched for transactions that can still be completed with the current capacity. Though with possibility of starvation. Other possibilities are having dynamic prioritization for transactions that are queued.

3.1.3. Evaluating against related work

As the state of current research in PCNs is limited, and the area has started to gain more attention as of late. Evaluation via comparison to other methods is difficult due to the limited or non-existent research papers

of current solutions to the problems proposed. The performance of the synthetic data-set can be compared to the "data-set" from ripple, as mentioned in Section 3.1.5 there are some concerns about the validity of the data-set and thus it will be ignored. Further more, researchers are looking for data-sets to be provided by Lightning or Ripple, or other blockchain solutions that make use of PCN. However, currently no data-sets have been made available to the research community, and with some speculation even the companies themselves are unsure how to gather this data.

Comparing measurable results between rebalancing strategies currently is not possible due to the lack of data available in this area of research. While rebalancing has been proposed by Lalitha *et al* in their paper Rebalancing in Acyclic Payment Networks[70] for the rebalancing strategy, their work does not evaluate their strategy against a simulation and make use of an extra protocol within their PCN. This extra protocol rearranges link capacities with multiple nodes, this happens outside of transactions. Rami *et al.* also propose revive[39], a rebalancing strategy with a leader selection, a nodes sending in requests. In their paper the results from the simulation are non-comparable, as not much is known about how the simulation was run.

While comparison against other research is difficult, both of the proposed questions can be compared to different topologies and against one another. The limitation to such an evaluation is that it will be impossible to evaluate the effectiveness of both the algorithms and synthetic-data relative to other solutions.

3.1.4. Topology

Research on the topology of PCNs is limited, on the lightning topology Seres *et. al* have published a study[67]. Such studies may not be indicative of the whole network as the number of nodes looked at was 2344, while currently the number of nodes in the lightning network is 4 times that, at 10,150 nodes[7].

Lightninings Topology Model

An interesting aspect of the study found that the topology of the Lightning network can be classified by the scale-free model, with the distribution of channels following a power law. Power law distribution of degree's is often found in natural occurring networks and such networks have been studied for network resilience.

Lightnings Robustness

Network resilience is a probabilistic measure of potential disconnections in a network[49]. A network may have to deal with two types of potential disconnections, *random failures* and *targeted attack*. In a *random failures* random nodes within a network will fail, such a test tries to emulate hardware failures and other random occurrences that may put a node in a network offline. Within a *targeted attack* specific nodes are targeted to see how the network holds up to such an attack. These attacks are usually performed by attacking nodes with the highest degree or in the case of PCN the nodes with the highest amount of channels.

Random failures, using the Molloy-Reed criteria[46], Seres found that the percolation threshold measured was $f_c = 0.9797$. Such a threshold has been proven to be adequate for retaining stability of the network under random failures. A random selected node in the lightning network failing will have little impact due to most nodes not contributing highly to the topological connectivity.

Targeted Attacks, attacking a PCN can be done via DDoS attacks. Seres found in their research that removing the single largest node in the network fragmented the network into 37 different connected components. Attacking the 30 largest hubs fragmented the network in 424 components. This shows the fragility of current PCN, a DoS attack would currently be able to disrupt the lightning network.

3.1.5. Dataset

Analyzing and making use of data-sets allows researchers to emulate a PCN accurately and allows helps validate testing new ideas. In Roos *et. al* work on SpeedyMurmurs a dataset was presented[65]. This dataset was taken from ripple and made publicly available. After a data analysis of the dataset questions were raised to how use-full the dataset is. Ripple is a credit network and allows establishment of channels with infinite capacities and this does not translate well to other PCNs. Where establishing a channel means funds must be deposited, in ripple this is not necessary. Giving very skewed data results. In the appendix A more can be found out about the dataset and its analysis.

3.2. Rebalancing of a Payment Channel Network

Two different methods have been proposed for the rebalancing of PCNs. Both of these rebalancing strategies make use of a protocol outside of the transaction protocol.

3.2.1. REVIVE

Khalil *et al.* proposed REVIVE[39], the first ever PCN rebalancing protocol that did not require nodes to interact with the blockchain to rebalance links. The protocol requires multiple leaders to be elected in the system. The job of a leader is to gather information on the link capacities of the nodes that have elected the leader so that the leader can make a scheme so that nodes will be able to rebalance links. Then nodes can make a payment to themselves in a *cyclic* graph, allowing the restoring of link capacities. The leader aims to rebalance the link in such a way that the link is equally balanced bidirectionally.

The leader makes use of a Linear Problem to optimize the link balances between links. This search is done locally instead of having a leader calculate it for the whole graph. Multiple leaders that have been selected, calculate locally the optimum rebalancing strategy, the local area of the rebalancing can overlap with other local-areas. While it is suggested that this is less effective then having a global calculation for the optimum rebalancing strategy, it reduces the calculation time of the problem. This design has two big draw backs, one is that it only works for cyclic graphs and the security risk created by sending all link capacity information to a chosen leader may proof to be problematic.

3.2.2. Rebalancing in Acyclic Payment Networks

Subramanian *et al.* have proposed a rebalancing strategy for Acyclic Payment Networks[70]. Within this paper unidirectional and bidirectional links are able to benefit from the proposed rebalancing strategy. The protocol consists of three phases the *Rebalance request phase*, *Respond phase* and *Reserve and pay phase*. During the rebalance request phase, a node will ask a sub-set of its neighbours if it they can help in the rebalancing. This sub-set consists of all neighbours except for the neighbour that the intended rebalance is meant to take place in. The neighbour that the node wants to rebalance with will be also requested, the response of this request is a signed contract that locks the *value* of the rebalancing agreement between them. Once the node has received enough contracts to finalize the rebalancing from its neighbours and possibility its neighbours neighbours till a fair way is found to rebalance, then the rebalance transactions occur.

3.3. SpeedyMurmurs

Currently one of the promising PCN algorithms is speedymurmurs. This algorithm was first proposed by Roos in 2017, and it is based off-of VOUTE[65]. VOUTE being an algorithm that allows for efficient message delivery in dynamic route-restricted networks[64]. A route-restricted network is a network that does not allow nodes to set-up arbitrary links between themselves. A key feature of VOUTE is the privacy-preserving embedding-based routing algorithm[57]. VOUTE was designed to work within P2P networks, transmitting messages in uni-directed and unweighted networks. The privacy guarantees, and dealing with routing make VOUTE very interesting for PCN, though due to being unable to work with weighted links and a credit network it has been adapted into speedymurmurs.

Speedymurmurs key properties are composed of:

- **Path selection**, path selection for SpeedyMurmurs is based on two attributes of the PCN. The funds available and the distance a neighbouring node is to the destination. The specific path selection for SpeedyMurmurs results in a flexible and efficient algorithm.
- **Low overhead**, PCNs are dynamic graphs. For that reason SpeedyMurmurs makes use of an on-demand efficient stabilization algorithm. The overhead due to the algorithm is low without compromising SpeedyMurmurs efficiency in reacting to changes in the PCN.
- **Split Link Capacity**, SpeedyMurmurs tries to ensure efficient usage of resources in a PCN. For this reason during transactions, only the link capacity that is needed to complete the transaction is blocked-off allowing for multiple transactions over the same link to take place. As long as there is link capacity available.
- **Multi-path Transactions**, using multiple paths to send transactions over, reduces the load curtain links experience during a transaction. By spreading the load of a transaction, SpeedyMurmurs distributes

and balances the load on link capacities of transactions over as many nodes as possible. Trying to mitigate links becoming over utilized and no longer being able to be traversed due to bad link capacity.

3.3.1. Routing and Path Selection

Embedding-Based Tree Shortcut Routing

PCNs are not directly comparable to other networks due to the rigid structure the network is created in. Peers predefine connections by opening payment channels with one another and have an established topology that is not easily altered. SpeedyMurmur's makes use of the similarity of Friend-to-Friend(F2F) networks, these networks only allow connections to between nodes in the network if a mutual trust relationship is established.[65]

F2F networks current use of routing is embedding-based routing[56, 64], such routing relies on a coordinate system. Coordinates of nodes are dictated by the root node from a spanning tree, thus to initialize an embedding-based routing system, a spanning tree must be created. After coordinate system has been assigned to different nodes it is common to disregard the tree and base routing on the coordinate system.

A root node is able to assign a coordinate system by forming vectors. The root node has an empty vector and will assign a coordinate to each of its children. These children will assign coordinates to their children and so on. As seen in figure 3.2 a, the root node has an empty vector referring to the root node, where as its direct children get assigned a vector value.

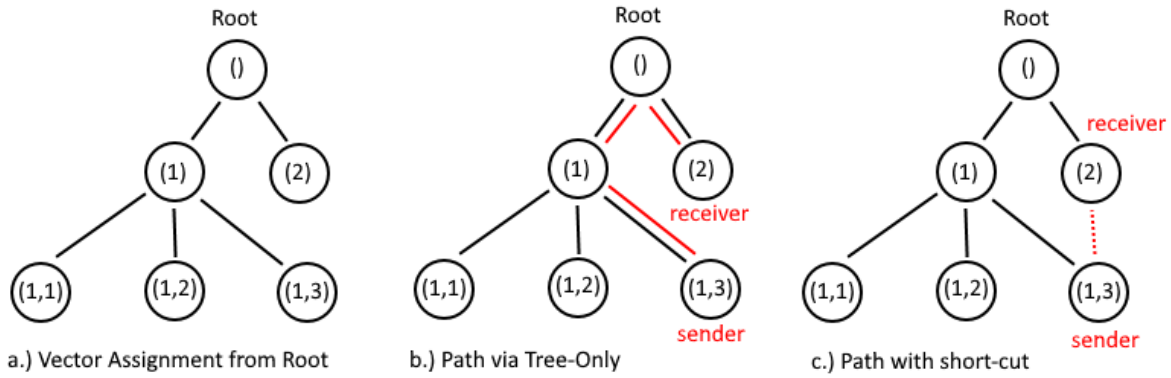

Figure 3.2: Coordinate assignment by root node, prefix-embedding based routing

With a coordinate system paths do not have to follow the path of a tree, as nodes may be linked to a node that they are not directly connected to in the tree. Thus allowing for paths to be shortened by taking a shortcut instead of precisely following the path established by the tree. Shortcuts are not guaranteed, while tree paths do guarantee that a path can be established between two nodes. For the reason that paths must be guaranteed, disregarding the tree after coordinates are assigned is not possible.

3.3.2. Concurrency

SpeedyMurmurs works in a *partial-blocking* manner, allowing for **Multi-Path Transactions**. Multi-path transactions allow for a transaction to complete over multiple paths. The amount of value that is being transacted will be divided into different amounts over each path. The thought behind a multi-path transaction allows for many smaller payments to go over the network. This may reduce the dependency on certain links and may give an over-all better concurrency performance.

3.3.3. Security

Security within SpeedyMurmurs, a layer-2 protocol, is built upon the security and trust assumptions from the underlying layer-1 protocol. For the layer-2 protocol, the model that is being built also extends its security for the layer-2 protocol to accountability, corrupt users and denial of service.

Trust: parties that are transacting within one another or are aiding in the transaction should be trusted to report the value of the links accurately. Non of these parties should be misrepresenting the capacity on the link, nor miss represent the value of the transaction. If any user is actively participating in any distribution of misinformation, the honest peers will be able to detect and validate that a party is acting malicious.

Each user after participating in a transaction will actively re-balance their values for the link capacity. These values will be updated properly and without tampering to the agreed upon values that was established between the participating parties in the transaction. Third-party peers should be able to validate that these values are correct.

Corrupt User: failure, data corruption and network failure is a reality every distributed network needs to be able to deal with. within this model a corrupt user that has hardware failure or network failure will no longer be able to participate in this network. Once the corrupt user is able to fix hardware or network issues, it will be allowed to join the network again. Such a user will join the network with values for link capacity that are correct, these values should be validated by a third party.

Data corruption is an issue that is more difficult to detect. Though due to every transaction being created in such a way that users in the transaction and third-party users in the network should be able to validate it. The network will realize that data-corruption has occurred and the values are not to be trusted. The user with data-corruption should be able to revert back to the proper values by either addressing a back-up or validating link capacities from its connected users and adapting its own values.

Denial of Service: if a participating node is denying other nodes of part-taking in the network via a denial of service attack or any-other means. This party will be dropped from the network. Nodes are assumed to have equal opportunity to participating the network, this assumption is important and will be protected by the network. Creating a fair network allows every node to participate equally and will create the most robust network.

3.3.4. Privacy

Privacy in the SpeedyMurmurs starts with the privacy assumptions of the underlying blockchain privacy assumptions. The privacy that can be achieved is directly related to the privacy of opening and closing a payment channel on the blockchain. If this privacy is compromised the anonymity of the user in the layer-2 protocol will also be broken. However the amount and value transacted between two compromised parties will not be known. Except for the starting deposited channel and the closing values of the channel.

Link Privacy: a node in the network will be unable to gather information about link capacities between nodes that it is not directly connected to. Even if a node is participating in a multi-node transaction, the node will only be able to gather knowledge about link capacity of the nodes its directly connected to. Any other link capacity information in the transaction will not be shared or cannot be requested by any adjacent node not linked to the capacity.

Anonymity: a couple of properties need to stay anonymous during a transaction. The amount during an exchange, value privacy, is an important key to the privacy goals of speedymurmurs. Secondly no certainty can be given to whom the sender/receiver are in a transaction. Though participating nodes in a transaction cannot be obscured from one another, nodes will not know if they are the first to receive a message, helping in finding the sender. Nor will they be able to get the information to know if a node is the receiver. Thus protecting the privacy of sender and receiver.

4

Goals and a Debut for Merchant Algorithms

Within this chapter an introduction is given into the Merchant algorithms. Answering one of the research questions. This question is for a set of transactions that run through a PCN, can the success ratio of a PCN be increased by adding a monetary incentive tied to the fees of each transaction be used to aid in keeping links balanced?

4.1. Fees and incentive

The main concept behind the Merchant algorithms is that it relies on adjusting fees based on the rebalancing potential of a payment channel. **Rebalancing potential** refers to the potential amount a link capacity will change if a certain transaction transacts over that link. The potential amount should reflect if a transaction is allowing a link to be more balanced and have a bidirectional connection or the transaction influences the capacities on the link in such a way the link becomes directional.

Formalizing the definition of bi-directional, a **bi-directional** link refers to a link that has nonzero funds in both directions, in figure 4.1 two nodes are shown with there respective links (u, v) and (v, u). The amount of capacity on a link is described by c(u, v). A link is bidirectional when c(u, v) > 0 and c(v, u) > 0. A link is denoted as **directional** when one of the two links equals 0, e.g. c(u, v) = 0 and c(v, u) > 0.

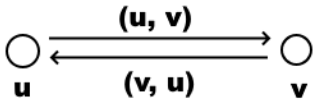

Figure 4.1: Labeling of nodes and there links

4.1.1. Rebalancing Potential of a Payment Channel

The rebalancing potential of a payment channel describes the potential-effect an incoming transaction can have on the link's capacity. While not every transaction depletes the capacity of a link, each transaction will contribute and change the balance of a link. Determining if a transaction is detrimental to a links capacity depends on the direction the transaction is going and the point of reference for a balanced link.

To determine a rebalancing potential, a value needs to be determined that symbolizes the ideal balance for a link so it is able to handle the most possible transactions. Depending on the perspective and the dynamic behaviour of a PCN the rebalancing reference value may have to be adjusted over time, other factors may also influence the value for the reference of a balanced link, i.e. the topology or the direction most transactions go over the graph. To simplify this challenge two different points of reference are proposed in this research, **50-50 Rebalance Potential** and **Initial State Rebalance Potential**. To reduce complexity and without having accurate insights into a PCN both of these rebalancing potentials are statically assigned and do not consider the behaviour of the network over time, these concepts are introduced below in there respective sections below.

With a point of reference for the rebalancing potential, the change in rebalancing potential can be quantified. With quantification of rebalancing fees can be attributed to a path for a transaction. The way fees are determined and what factors are considered are discussed in the subsection **Fees vs multi-hops**.

50-50 Rebalance Potential
A 50-50 rebalance potential refers to the perfect balance in a link to be equally on both sides, i.e. $c(u,v) = c(v,u)$. If a transaction occurs while the channel is in perfect equilibrium, it would have a negative potential. However if a link was unbalanced and it was brought closer to a 50-50 split then the potential would be positive. Such a scheme seems like an obvious choice for a rebalancing potential scheme. As with an equilibrium in the link capacity the link is able to process transactions going in both directions. One issue with this method is that if nodes have intentionally created a link with an unbalanced capacity due to it being known that transactions will mostly go in one direction, the rebalance potential will try and balance the link around the wrong point of reference.

In the Figure 4.2, a scale-free topology is illustrated. Such a topology is defined by its power-law distributions for the degree of the nodes in the graph. Such graphs occur naturally and are a common topology within a natural occurring network, i.e., electricity, internet, connection of airports[14]. Most nodes are connected through a couple central nodes, these central nodes defined by a high degree act as hubs. Having a 50-50 rebalancing strategy for the hubs is an obvious solution that will aim to always have nodes be able to transact. For the nodes with a single link in this scenario a 50-50 rebalance potential would be beneficial, thus nodes with a singular link should follow a different protocol.

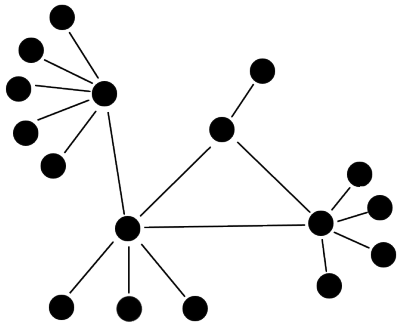

Figure 4.2: Transactions vs Three Sets of Channel Distribution

Initial State Rebalance Potential
Another way to approach an rebalancing potential is by taking the initial state of a payment channel the ideal state of the channel. Taking the initial state of a payment channel works under the assumption that during the set-up of the payment channel, the two parties creating the payment channel have deliberately created a balance in the payment channel that is needed.In such a scenario, the initial state of the payment channel is stored and referred to as a reference point.

Dynamic State Rebalancing Potential
As more knowledge is gained on PCN and the flow of transactions can be modeled more accurately. One can imagine that the point of reference for a balanced link may change over time depending on the conditions of the nodes. Such conditions may include linkage, degree, the role of the node in the network and so on.

4.1.2. Fees vs Multi-hops

Fees in the context of PCN are calculated by totaling the Base fee and a Fee rate. **Base fee** is the fee that is always payed when a transaction goes for a link and is constant, **Rate fee** is the fee that is inquired by the amount of value that the transaction is trying to send. In equation 4.1 the equation to calculate fees can be seen.

$$Fee = base\_fee + rate\_fee * Transaction\_value \tag{4.1}$$

Currently in the lightning network the median base fee is 1 sat and the rate_fee for a transaction is 0.000001sat[1], this indicates that the vast majority of nodes in the system have not altered the fee rate from the default rate

provided by the Lightning service package. Allowing for the assumption to be made that the fees that are inquired during a transaction are directly proportional to the hop-count of the transaction. For nodes wanting to rebalance a link, this means that incremental fees will need to be generated. **Incremental fees**, refers to the notion that fees must be multiples of a fee, $fee = N * defaultfee$. Where N is an integer.

With the hop-count being directly proportional to the amount of fees induced for a transaction, it needs to be beneficial for a transaction to take a longer path if it helps re-balance the network. Thus a positive rebalance potential must also allow for a transaction to take a longer path, then a shorter path. This can be achieved by finding a good relationship between a positive and negative rebalance potential.

For a transaction to take a longer path but pay a lower fee, the shorter paths must either have a negative rebalance, or neutral rebalance and the positive rebalance must be so beneficial to the network it is able to compensate for the longer path. Finding such a balance is non-trivial and may heavily depend on the topology of a network and the average length of a transactions hop-count.

4.1.3. Rebalancing Fee Factor

The rebalancing fee factor is an integer that is multiplied by the base fee. Equation 4.2 gives the total fee that is requested by the forwarding node.

$$Fee = rebalancing\_fee\_factor * base\_fee + rate\_fee * Transaction\_value \tag{4.2}$$

The fee factor is an integer, this stems from the assumption that every hop in a transaction requests the same fee amount due to almost all nodes making use of the default fee values. The rebalancing fee factor is only run in forwarding nodes, as these are the only nodes that request fees during a transaction.

The goal of a node is to have enough capacity on its edges to help aid in transactions, if a links value is at the reference point or above; the node will most likely be able to help a transaction over that link. If a links value falls below that reference point the node may no longer be able to send transactions over that link. Determining the rebalancing fee factor can thus fall into two scenarios depending on how a transaction effects the link.

1. Transaction reduces capacity on the link below reference point
2. Transaction does not reduce capacity on the link below reference point

From this logic, if a transaction sends the value of the link below the reference point a high fee is requested to try and deter the sending node to use that path. If a link has more capacity on its line than the reference point and the transaction does not cause the the capacity of the link to fall below this point then the node would like to encourage the sender node to make use of this link.

To determine the rebalancing fee factor for the scenario that the capacity falls below the reference point, the difference needs to be found between the reference point and the new capacity as seen in equation 4.3.

$$diff\_c = 1 - (new\_capacity \div reference\_capacity) \tag{4.3}$$

The factor is dependent on the percentage the new_capacity takes on compared to the reference_capacity. Now depending on the tenths value of diff_c a factor is created, that is why the diff_c is multiplied by 10 then truncated as seen in equation 4.4.

$$rebalancing\_fee\_factor = trunc(diff\_c * 10) \tag{4.4}$$

To determine the rebalancing fee factor for the scenario that the incoming transaction does not bring the link capacities below the reference point the fee factor is not determined by the outgoing link. This transaction should be encouraged, and thus a negative fee will be allocated to this transaction. Such transactions should be encouraged due the rebalancing effect it will have on the other side of the link. While this link is positively unbalanced for the node forwarding the transaction, the next node in the path would have a link capacity below that of the reference point.

In Equation 4.5 the calculation can be seen for diff_c. Currently the min value is taken for the giving diff_c a max value of 0.9, this limits the amount the rebalancing_fee_factor can be. Equation 4.6 shows that the rebalancing factor is negative and is truncated the same way as before.

$$diff\_c = min(0.9, 1 - (new\_capacity \div reference\_capacity)) \tag{4.5}$$

$$rebalancing\_fee\_factor = -1 * trunc(diff\_c * 10) \tag{4.6}$$

4.2. Merchant Algorithms

Within this section the passive and active merchant algorithms will be described. The main working principle in both of these algorithms is the re-balancing-factor

4.2.1. SpeedyMurmurs and the inclusion of the Merchants

SpeedyMurmurs is a transaction routing algorithm for a PCN that lends its self very well to the inclusion of the Merchants. As each node is able to act as their own merchant, no additional information needs to be shared in the network. Allowing for the protocol to take advantage of the embedded routing scheme it uses. Another great advantage to SpeedyMurmurs for the merchants is that it is able to route payments over multiple paths. This gives SpeedyMurmurs the flexibility to spread the load of a payment across multiple paths in a network, though these paths may converge to one if the fees for that specific route are best.

4.2.2. The Merchants

Both the Merchant algorithms make use of similar protocols, that effect the sender and forwarder. For the sake of completion how the receiver handles messages is also shown. Both Merchant algorithms make use of the functions given in the list below:

- Initial Path Request
- Locking Paths
- Completing Transactions

While the Merchants algorithms differ, the main mechanism that separates the Passive Merchant with the Active Merchant is a messaging protocol. The Passive Merchant does not share information with anyone in the graph, while the Active Merchant actively tries to get transactions to make use of its links when unbalanced. As the main difference between the active and passive merchants is sharing and sending of information about the link capacity, the methods that handle locking and completing the transactions is identical and will be covered together.

The general work-flow of these algorithms can be described by a sender, a forwarder and a receiver. The sending node, known as the sender, will query multiple trees to find a path. Each forwarding node, known as the forwarder, will give a fee depending on the algorithm chosen and the amount that is being transacted. The receiving node of the transaction, known as the receiver, will reply to the messages to accept the path.

Once all or enough paths have been returned to the sender node, it will choose the paths to transact over. From here the process is fairly standard, the transaction amount will be pended in each path. Then once enough funds are secured to complete the transaction the sender will start an HTLC contract over all paths so that the receiver can receive all payments. Then the transaction has been completed.

4.2.3. Passive Merchant

The passive merchant algorithm enables nodes to act as a passive merchant. A passive merchant does not let neighbours or other nodes know if it's links are unbalanced. During a transaction each node in the path will act as a passive merchant and will adjust it's fees according to the re-balancing policy that has been set. The fees are passed along during the initial probe of investigation for a transaction. Once an path has been accepted the fees from the initial probe are set. The algorithm starts with an initial path search from the sender.

Algorithm 1 Sender: Initial Path Request

```
   initializeTransaction()
2: transId = random()
   nQueries = numOfPaths + N
4: [X,Y] = getCordinate(endNode, neighbourhood)
   amount = totalAmount / numOfPaths
6: for all nQueries do
     pathId = rand()
8:   sumOfFee = 0
     randomTree = getNewRandomTree()
10:  sendQueryMessage(X,Y,amount, transId, pathId, sumOfFee, randomTree)
   end for
```

Algorithm 1 shows the how the sender handles the initial path request. During this stage SpeedyMurmurs will send initial path requests to see if a path can be found along a route to a receiver node. These paths follow the underlying tree structure that has been set-up during the initialization of a network. Different initial probe messages are sent along different sets of spanning-tree graphs within the network. These initial probe messages are forwarded in the direction of the receiver via the coordinate system, and short-cuts are taken when possible. As is described in the SpeedyMurmurs implementation[65] and Section 3.3.1.

During the initial probe the sender sends out more probes then that paths it requires, seen in line 3 where N symbolizes the extra number of trees to try and numOfPaths the number of paths that are required to do the transaction. Asking more paths than required allows the *sender* to base the path of the transaction on the cheapest fees. Each path is sent the same transId, as this is the unique identifier for the transaction as seen in 2. Each path does get a unique identifier as paths may share a path along the same node, as seen in 7. Line 9 summarizes how a random tree is selected, this function must keep track of selected and non-selected trees. As sending the same inquiry down the same tree should not happen.

Algorithm 2 Forwarder: Initial Path Request

```
   receivedQueryMessage()
2: if virtualCapacityTowardsReceiver > amount then
     feeFactor = calculateRebalancePotential(amount, linkTowardsReceiver)
4:   fee = feeFactor * defaultFee
     sumOfFee = sumOfFee + fee
6:   forwardQueryMessage(x,y, amount, transId, pathId, sumOfFee)
   else
8:   capacityError(edgeTowardsSender, transId, pathId)
   end if
```

Algorithm 2 shows the process a forwarding node goes through when receiving an initial path request. Line 2 shows that the algorithm must first check if it expects to have enough capacity to handle the transaction. If not enough capacity is on the line, then the receiver will send an capacity error message along the path to the Sender. The Sender can try a different path or if it receives enough capacity errors the transaction will fail for the time being.

Then in line 3 the rebalance potential is calculated and a fee factor is designated from this potential as explained in Section 4.1.3. Line 5 shows how the sumOfFee is calculated. The sumOfFee is a variable that is sent with each initial request send message, to ensure that the sending node can determine which path is cheapest.

Algorithm 3 Receiver Node: Initial Path Request

```
for all nIncomingQueries do
  sendPathCompleted(edgeTowardsSender, transId, pathId)
end for
```

Algorithm 3 showcases the receiving node. The receiving node accepts all incoming transaction queries, as it is the receiving node. The receiving node does not calculate fees for the transaction and accepts all initial

path requests. Then the path accept is sent down the links till the sender receives all the information.

Once the path is accepted by the receiver, the accept message is past down along all the forwarder nodes. When the sender node receives

Active Merchant
The Active Merchant algorithm enables nodes to disseminate information about their links trying to lure in transactions. An active merchant actively tries to keep the balance between its linked nodes in balance to allow for transactions. Once the balance in a payment channel has reached a critical in-balanced state or is a a bidirectional link has become directional the active merchant will send messages down certain trees to encourage the usage of that tree and give a discount for fees, trying to access that path. This message sending is the reason why it is called active opposed to its counter part the Passive Merchant.

Such discounts are given with a *coupon*, these coupons can be sent along the path during an inquiry of a path to get the discount for the fees. The amount of discount given is the determined by the last rebalancing_fee_factor for that link. This amount is given by:

$$discount = -1 * rebalancing\_fee\_factor * base\_fee \tag{4.7}$$

Such a coupon guarantees the sending node the fee on the coupon, thus it has the pre-knowledge of paths to try. Figure 4.3, shows two nodes where the link capacity of Node A is only 20, and for Node B is 120. The coupon is sent along the path of the link it has the imbalance with. The rebalancing_fee_factor in this instance is -8.

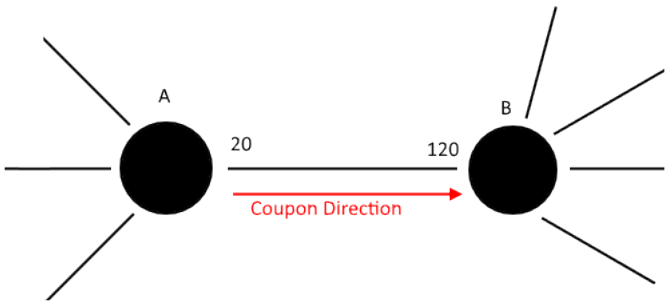

Figure 4.3: Direction of coupon after an imbalance is established

Sending out Coupons:
Coupons are sent out along the link on different spanning-trees if the imbalance of the capacity crosses a certain threshold. This threshold is any arbitrary value that may risk the link closing. For this model 50% was used as threshold.This process is initiated after a transaction has been completed, then the capacity has a new balance and the new balance can be compared to the reference point and the threshold. Each coupon is sent with an expiry time, if the link balance has not been fixed before this expiry time the coupon is sent again.

Algorithm 4 Coupon Creation

```
    for all nLinks do
2:    linkBalance = checkLinkBalance(nLinks)
      if checkCouponAlongLink(nLinks) then
4:      coupon = getCoupon(nLinks)
        if coupon.linkBalance == linkBalance then
6:        if checkExpiryTime(coupon) then
            resendCoupon(coupon)
8:        end if
        else if coupon.linkBalance > linkBalance then
10:       killCoupon(coupon)
          expiry_time = EXPIRY_TIME + system_clock;
12:       couponId = rand()
          store(couponId, expiry_time)
14:       sendOutCoupons()
        else
16:       killCoupon(coupon)
        end if
18:   else
        outGoingEdge = getOutGoingEdge(nLinks)
20:     BALANCE_THRESHOLD = getBalanceThreshold(nLinks)
        if linkBalance < BALANCE_THRESHOLD then
22:       expiry_time = EXPIRY_TIME + system_clock;
          couponId = rand()
24:       store(couponId, expiry_time)
          for all SpanningTreeUsingLink(nLinks) do
26:         sendOutCoupons(couponId, maxNumHops, expiry_time)
          end for
28:     end if
      end if
30: end for
```

Algorithm 4 starts after a transaction has been completed. Once a node finishes a transaction it will check the link-balance of each of its edges. Line 1 represents a loop and *nLinks* represents the nth link that is connected to a node. In Line 3 the algorithm first checks if a coupon has been sent along that edge, if a coupon has been sent along that edge another check will take place. In Line 5 the balance of when the coupon was sent out is checked, if this has not changed the algorithm will check the expiry time of the coupon. If the coupon has expired the coupon will be resent otherwise the algorithm will continue on to the next link.

In Line 9 a check is done to see if the previous link balance was higher than it is currently, if so a new coupon needs to be created. A killcoupon function as seen in line 10 sends out messages stating the previous coupon no longer can be used. Then a new coupon is sent along the edges. Otherwise the coupon is no longer needed and is killed.

If the algorithm was unable to find a coupon for that link it will check if a coupon needs to be created. In Line 21 this check takes place, if the link Balance is below that of the balance_threshold a coupon needs to be sent. This coupon is sent along the link along all spanning-trees, if they involve that link.

Algorithm 5 Node receiving a coupon

```
receivedCouponMessage()
if not doesCouponExist(coupon) then
  store(Coupon, SpanningTreeId)
  forwardOutCoupons(couponId, maxNumHops, hopsMade)
else
  doNothing()
end if
```

Algorithm 5 showcases what a node does if it is downstream of a node sending out coupons, it needs to handle the coupon message. This is done by checking if the coupon has already been received, if not then the coupon is stored locally to be accessed during a transaction. The coupon is also forwarded along the spanning-tree path downstream.

Initial Path Search:

Algorithm 6 Sender: Initial Path Search

```
Initialization
transId = rand()
for all nQueries do
    (x,y) = getCordinate(endNode, neighbourhood)
    amount = totalAmount / numOfPaths
    pathId = rand()
    sumOfFee = 0
    sendQueryMessage(x,y,amount, transId, pathId, sumOfFee)
    if receivedCoupons and
    couponPathAlongTransactionPath then
        sendQueryMessage(x,y,amount, transId, pathId, sumOfFee, couponId)
    end if
end for
if numReceivedCapacityError < numOfDeclinesAllowed then
    (x,y) = getCordinate(endNode, newNeighbourhood)
    sendQueryMessage(x,y,amount, transId, pathId, sumOfFee)
else
    cancelTransaction(timeTryAgainLater)
end if
```

Algorithm 6 displays the situation of an active merchant when the node is a sender. This algorithm is close to that of a passive merchant. Except the sender in the active merchant case has information about possible paths, thus reducing some of the randomness associated with picking of paths. Line 9 would not be needed in a passive merchant sender, for the active merchant, if a coupon is found that is going in the direction of the payment the coupon ID is added to the data-structure of the message. Multiple coupons can be added to a message if it has come from a certain, to try and maximize coupon usage for the sender.

Just because a node sends a message along the path of a received coupon does not mean the coupon can be used. This is due to the embedded routing of SpeedyMurmurs, if a short-cut is found to the receiver skipping the sender of the coupon, the coupon will not be redeemed.

Algorithm 7 Forwader: Initial Path Search

```
receivedQueryMessage
if virtualCapacityTowardsReceiver > amount then
    if couponOwner then
        fee =coupon_fee
    else
        rebalancePotential = calculateRebalancePotential(amount, linkTowardsReceiver)
        fee = rebalancePotential * defaultFee
        sumOfFee = sumOfFee + fee
    end if
    forwardQueryMessage(x,y, amount, transId, pathId, sumOfFee)
else
    capacityError(edgeTowardsSender, transId, pathId)
end if
```

Algorithm 7 showcases a forwarding node during an active merchant, the active forwarder acts the same as the passive merchant unless it is the giver of the coupon. Upon receiving the initial path request, and the message is checked for a coupon. If the forwarder is the owner its fee will be adjusted to that of the coupons

price.

Algorithm 8 Receiver: Receiving Initial Path Search

```
for all nIncomingQueries do
    sendPathCompleted(edgeTowardsSender, transId, pathId)
end for
```

Algorithm 8 showcases a receiving node receiving an initial path request the receiving node sends acknowledgement down the path.

4.2.4. Completion of the Merchants

The algorithms presented in this section showcase how the rest of the transaction is handled by the Merchants. This protocol stems from SpeedyMurmurs, it has been added for completeness of the algorithms.

Locking paths:

Algorithm 9 Sender: Locking Paths

```
for all nReceivedCompletedPaths do
    choosenPaths[] = chooseCheapestPaths(nReceivedCompletedPaths)
    for all choosenPaths[] do
        sendMessageRequestingLockFunds(x,y,amount, transId, pathId)
    end for
end for
```

Algorithm 9 dictates the sender node during the locking of the paths selects the paths with the least number of fees. Once the selection is made a message is forwarded along the path to lock the requested funds.

Algorithm 10 Forwarder: Locking Paths

```
if receivedMessageRequestingLockFunds then
    sendMessageReqyestingLockFunds(edgeTowardsReceiver, transId, pathId)
else
    if receivedMessageLockFunds then
        if virtualCapacityTowardsReceiver > amount and
        rebalancingPotential < 0 then
            updateVirtualCapacity(amount)
        else
            dontUpdateVirtualCapacity()
        end if
        sendMessageLockFunds(edgeTowardsSender, transId, pathId)
    else
        capacityError(edgeTowardsSender, transId, pathId)
        capacityError(edgeTowardsReceiver, transId, pathId)
    end if
end if
```

Algorithm 10 gives the psuedoco if a forwarder node receives the requesting of locking funds, it locks the funds from the transaction into the virtualCapacity. **VirtualCapacity** is a pessimistic capacity value that represents the true capacity. However if the transaction is unable to continue once a locking funds has been granted the virtual capacity is reverted. This is supposed to help ensure that a forwarding node does not promise outside of its capacity limits for a transaction. If during this time the capacity has changed from the initial probe and the funds cannot be locked, a capacity error is sent along the path to the sender.

Algorithm 11 Receiver: Locking Paths

```
if nOfPaths == receivedMessageRequestingLockFunds then
    sendMessageReceivedAllLockedFunds(edgeTowardsSender, transId, pathId)
else
    if waitTimedOut then
        sendMessageCancelTransaction(edgeTowardsSender, transId)
    else
        waitForAllRequestsToLockFunds()
    end if
end if
```

Algorithm 11 shows for the receiving node its process, it accumulates all the requesting Locking Fund requests into one transaction to ensure that enough value is being sent over the links to ensure a successful transaction. If enough funds have been secured to enable the transaction to be fulfilled an acknowledgement is sent down each path to the sender.

Complete Transaction:

Algorithm 12 Sender: Completing Transaction

```
if receivedFundsLocked then
    sendMessageCompleteTransaction(edgeTowardsReceiver, trandsId)
else
    if receivedCancelTransaction or receivedTimeOutError then
        cancelTransaction(transId)
        del(transId)
    else
        waitForFundsLockedMsg()
    end if
end if
```

Algorithm 12 represents how the sender node deals with receiving of a Lock_Funds acknowledgment, the sender node responds to each one as they come and starts the HTLC process. If during this stage the transaction cannot continue then the dispute is sent to a mechanism on the blockchain.

Algorithm 13 Forwarder: Completing Transaction

```
if receivedCompleteTransaction then
    sendMessageCompleteTransaction(edgeTowardsReceiver, transId, pathId)
else
    if receivedCloseTransaction then
        updateCapacity(amount)
        if rebalancingPotential > 0 then
            updateVirtualCapacity(amount)
        else
            alreadyUpdatedVirtualCapacity()
        end if
        sendMessageCloseTransaction(edgeTowardsSender, transId, pathId)
    else
        if time < timeOut then
            wait()
        else
            timeOutError(edgeTowardsSender, transId, pathId)
            timeOutError(edgeTowardsReceiver, transId, pathId)
        end if
    end if
end if
```

Algorithm 13, during the closing of the HTLC contract the forwarder updates the capacity of the link. The capacity and virtual capacity will be equal if no other transactions are occurring on the link. If value is added to the link capacity then at this point the virtual capacity is increased, as the virtual capacity is pessimistic and value is not added untill the transaction is complete.

Algorithm 14 Receiver: Completing Transaction

```
if receivedCompleteTransaction then
    updateCapacity(amount)
    sendMessageCloseTransaction(edgeTowardsSender, transId, pathId)
else
    if time > timeOut then
        timeOutError(edgeTowardsSender, transId, pathId)
        timeOutError(edgeTowardsReceiver, transId, pathId)
    else
        wait()
    end if
end if
```

Algorithm 14, received node receives ending of all HTLC contracts and the transaction has been successful.

With the addition of the Merchant algorithms the security goals and privacy goals for SpeedyMurmurs need to be evaluated again. To ensure that the addition of these algorithms do not compromise the current state of privacy and security within SpeedyMurmurs, and if compromises need to be made they should be addressed.

4.2.5. Attacker Model

SpeedyMurmurs was built against an adversary model that tries to gather information on a user's financial situation. The adversary is in control of a subset of nodes within the network, this is achieved by joining the network with it's own nodes or is able of corrupting existing nodes. Not all nodes are corrupt-able, it is assumed not all users will be as effected by malware attacks or can be swindled by social engineering.

Security Risks

With the addition of the Merchants, information to the sender may show that a singular path may be the most beneficial for the payment to take to rebalance certain links in a network. Taking a single path goes against protecting the transaction value privacy goals of SpeedyMurmurs, thus no matter the incentive, sender nodes must use multiple paths to send a transaction. **Value Privacy** is defined by PrivPay[47] as the ability for an adversary not to be able to determine the value of a transaction between two non-compromised users. However in the case of this adversary the transaction value privacy may also be broken if an adversary has enough nodes in a network.

The adversary can purposefully lure transactions through its links via fees, allowing an adversary to aggregate the transactions value and possibility determine the value of transactions. This may break the transaction value goals of SpeedyMurmurs, for such an attack to work the adversary would need to play a large role in the network. As such an attack can not be done with 1 or 2 nodes.

An adversary attacking the network and trying to undermine transaction throughput is not considered. It must be noted that such an attack with the Merchant algorithms may become easier by fabricating low fees for a transaction, thus routing all transactions over its paths and then stopping the transaction from closing. An adversary may also gain knowledge of the state of certain links, due to the broadcasting of discounts from the Active Merchant. It is not unthinkable that an adversary with this knowledge is able to hinder the network by creating a node in a place with bad throughput and routing payments through its nodes by manipulating the fees.

Privacy Goals

The addition of the Merchant algorithms should not affect the privacy goals of SpeedyMurmurs. As these algorithms mainly focus on the forwarding and rebalancing aspect of the network and do not effect the anonymity of a sender or receiver. In a PCN **Sender Privacy** can be achieved if a transaction takes place between a non-compromised set of transacting nodes. In this case the senders identity cannot be known

with 100% accuracy unless all incoming connected nodes to the sender are known to the adversary and compromised. For **Receiver Privacy**, the same notion hold as for sender privacy. The receiver cannot be known unless all neighbouring outgoing nodes are compromised by the adversary model.

Conclusion
The merchant algorithms do pose a security risk against the proposed adversary. If the adversary has enough nodes in the network, it can lure transactions and could know the transaction value of a transaction. While this may be the case, the adversary will never known if it has been able to intercept all-paths unless it comprises of a set of nodes in the network missing the two nodes who are transacting. The privacy goals with the merchants has not changed in SpeedyMurmurs.

5

Modeling the Synthetic Dataset and Toplogies

Within this section an explanation will be given into how the synthetic dataset was created and how one can manipulate parameters to get different effects. Alongside the dataset two synthetic topologies will also be introduced.

5.1. Challenges

Creating a synthetic dataset for a PCN is not a trivial task, many factors need to be accounted for to make a well rounded model. Without pre-existing data-sets of current PCN networks being available makes it impossible to verify the validity of a synthetic dataset. To overcome this challenge, instead of trying to create a model that tries to model what is going on in a PCN. An approach will be taken to modeling a dataset around use cases of a PCN and trying to incorporate available data to tune models.

5.2. Market Interest in Cryptocurrencies

A study in England showed that only 2.85% of Brits have bought cryptocurrencies[15], while in America that number is 8% and another 8% were looking to buy in the near future[31]. While the study did not conclude on how many of those individuals are interested in PCNs or have started a node. It does go to show how little of the population has adopted cryptocurrencies.

Currently 819 bitcoins have been locked into the lightning network[59], while 17,992,075 bitcoins have been mined by the blockchain[17]. The percentage of locked bitcoins in the Lightning Network is 4.5e−8%, its adoption by the community has been limited. Due to such a small amount of BTC being found in the network, one can assume that most transactions taking place have little financial value. A reason for the issue may be that there are limited amount of available interfaces for consumers and companies alike to interact with the network.

Ripple is the third biggest cryptocurrency on the market by value of market cap. It runs it network as a PCN, allowing for insight into how such a network may be modeled. Though the current datasets on this market are limited, market analysis can be used as a reference to model against.

5.3. Transaction Distribution

To be able to create an adequate transaction distribution, two factors need to be considered. The value distribution for transactions and the hop-count distribution for transactions. Together these two distributions should form a transaction distribution. Some reasoning for why these two distributions are needed and are related is due to human shopping and spending behaviour.

McKinsey published a report on global payments in their work "McKinsey Global Payments Map"[71] found that on average 6,5% of payments go across border for in the commercial sector[71]. In Europe that number is 12.5%, countries in that area are situated closely and have open-border trading. Consumers globally have lower cross-border transactions sitting at 4,5% and in Europe that is 8%.

One could argue that a great benefit of a PCN is being able to trade globally for a low fee, and in the future such an advantage may prove to be where PCNs get adopted by the market. If that were the case trading would most likely be done by big establishments in high transaction values. Ripple makes use of credit as apposed to debit based paying system. A market strategy ripple has adopted is working with current financial systems and banks[32].

Taking a look at the market from mid-September to mid-October for a span of 30 days; ripple has had a trading volume of £8.53e9[10]. Where 5 trading companies have participated in 71,8%. A use-case formed out of ripples PCN could give insight into how high value trading volumes shapes the network.

Another use case can be formed by taking a look at the Lightning Network, with a market cap of 6.05e6[6]. The market size indicates that many high value transactions are not possible. With a smaller market an assumption is made that it's user-base trades in a small amount of financial value, however these trades occur more frequently.

Figure 5.1: Topology of Lightning Network overlaying a world map generated by Lightning Network Explorer[5]

5.3.1. Transaction Value Distribution

Ripple Transaction Value Distribution Model and Lightning Transaction Value Distribution Model are introduced. Both of these Transaction Value Distributions will be limited by an *lower-* and *upper-bound*. The lower-bound will always be greater than a fee for doing a single transaction. The upper-bound is heavily dictated by the Channel Capacity Distribution.

5.3.2. Ripple Transaction Value Distribution Model

In the Ripple Transaction Value Distribution Model case one can assume that banks and other financial institutions are using the network to trade. With that assumption comes the need for a large network, this network would connect various institutions and have networks interconnecting between different branches of the same company.

To model such a value distribution a normal-distribution will be used. By limiting the rate of fire for transactions per hour the normal distribution can be modeled around the desired trading volume. By taking the cumulative distribution function and setting p around the variance of trade volume per hour one could find varying normal distributions that would model a banking case.

$$F_X(VolmeTradePerHour) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\pi^2}} \tag{5.1}$$

Such a model could give insights into how trading at higher values can impact a network.

5.3.3. Lightning Transaction Value Distribution Model

In the Lightning Transaction Value Distribution Model the assumption is made that a higher trading frequency occurs but the value of these trades are of lesser financial value. For this model a exponential function will be used. Modeling this after the Lightning Network is not trivial as trading volume data is hidden behind the privacy of off-chain payments.

$$F_X(VolmeTradePerHour) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{5.2}$$

5.3.4. Payment hop-count Distribution

The hop-count can be calculated for in a network as shortest-path between two nodes, this value can be varied depending on the type of model. Howerver, this may not be the path the transaction takes due to the internal routing algorithm.

5.3.5. Ripple Hop-Count Distribution Model

Assuming that financial institutions trade more readily across borders. This model makes use of the Exponential Distribution, trying to emulate across border transactions.

5.3.6. Lightning Hop-Count Distribution Model

Due to the nature of off-chain payments gathering a dataset around such information is non-trivial. The base assumptions for this model takes away from the McKinsey reports that it is an indication of consumer behaviour on a PCN. Most transaction happens will stay within borders, limiting the amount of hops a transaction can take. An pareto distribution will be used for this model.

5.4. Channel Capacity Distribution

Channel Distribution is heavily dependent on the topology of the network that is being simulated. As the lightning topology is scale-free, the network is dependent on hubs. Due to the high traffic demands of a hub, modeling these channel capacities may require a higher capacity then the rest of the network. For this reason a channel capacity distribution could distinguish between hubs and normal nods, giving both a different distribution.

Within the lightning dataset, nodes with 0 degree made up 30%. With such a high degree of singular nodes, to create a channel capacity distribution these nodes to be grouped in a subset of singular nodes. The singular node subset would get a distribution mimicking more individuals joining a network with lower capacity then the rest of the nodes.

For the Erdos-Renyi topology a channel distribution may be tried with a normal distribution. Taking away that it is not reliant on hubs, each channel may help serve in the network.

5.4.1. Ripple Channel Capacity Distribution

On the `xrpl.org` there is an api to fetch data from ripple. This may allow for more insights into the modeling of the ripple channel capacity distribution. For this thesis a normal distribution will be used.

5.4.2. Lightning Channel Capacity Distribution

The channel capacity distribution will be based on the lightning network. The graph data is taken from `www.rompert.com/networkgraphv2`, this graph has 4,798 nodes. The units of capacity gotten from rompert are *sat*, also known as Satoshi. 1 Satoshi is the smallest unit of btc available. Within this dataset 31098 channels are examined, each channel is directional. Looking at table 5.1, the average channel capacity lies around 200 euros, while the standard deviation is higher at around 363 euros. This indicates that there is little clumping of capacities and the distribution of capacities is not centered around the average. The median of the capacities lies at 43 euros, indicating that the distribution of capacity around the network is not normally distributed over the network.

	Capacity [sat]	Capacity [euro]
Total	80,948,627,914	6,102,957.22
Average	2,603,017	196.30
Standard Dev	4,815,936	363.18
Median	575,839	43.30

Table 5.1: Table showing capacity stats of 31098 channels in a lightning network

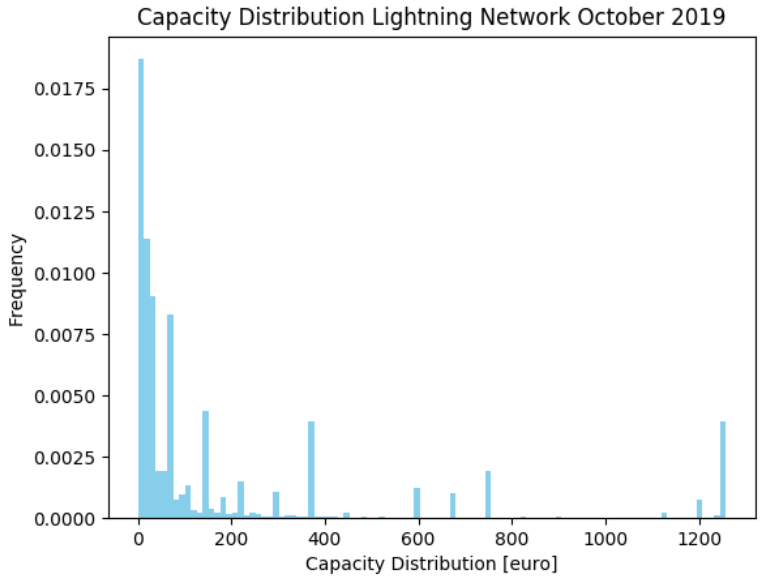

Figure 5.2: Link Capacity Distribution Lightning Network, October 2019

5.5. Topologies

The topology of a network defines how robust and resilient a network is to attacks and failures. Despite the network having such a crucial role, networks tend to grow naturally following a degree distribution associated with the power law[14]. The occurrence of this phenomenon is wide spread and the power law distribution can be found in telecommunication networks, social networks and more. Seres *et. al* have shown that the Lightning Network also follows a power law distribution for degrees and has been classified as scale-free network. For the reason that the Lightning Network already follows a power-law distribution, one could make use of the the Barabasi-Albert Model to create a synthetic topology.

Seres *et. al* showed in their work that attacking the node with the highest degree allowed for the network to fracture into 32 components. Such results shows how vulnerable a network can be. Trying to model a more robust approach to PCN topologies may give valuable insights into what is needed and what the trade-offs are to such a network. Due to the infancy of the PCN industry the topology may still be malleable and allow for a different structure to form. To generate such a topology the Erdos-Renyi model will be used.

5.5.1. Lightning Topology

To simulate the Lightning Topology a topology will be generated from `https://rompert.com/networkgraphv2`, retrieving the JSON and parsing it will allow for a network topology to be made.

5.5.2. Barabasi-Albert Model

Creating a model for Lightning Topology is no trivial task currently. As discussed in Section 2.2.7 the amount of nodes in a network is not easily established as different sources have different numbers. The dataset being used to create the Lightning Topology Model shows there are 4,809 nodes. With only limited insight currently into the topology of a Lightning network, the model made using the Barabasi-Albert Model will be based on the network topology gathered from Rompert[] that has 4,809 nodes.

5.5.3. Erdos-Renyi Model

While PCN are still in there infancy, changing the shape of the network may still be possible. While it is widely known in the networking community that most natural networks have a power law distribution when it comes to there degree connectivity. Such networks fall short due to nodes relying on hubs and have a limited amount of robustness. **Robustness**, is a measurement of how well a network is able to handle and deal with attacks on its network. This may be due to links failing or malicious parties activity trying to break links. Another downside to a scale-free network is that the diameter of the network is relatively low, reducing the amount of different paths payments can take. **Diameter**, is the length of the longest shortest path in a network.

An Erdos-Renyi graph is a graph that has an more evenly distribution of degrees for its nodes. This allows the network to have a higher diameter, more paths and be in-general more resilient to failures within the network. Such a network is harder to realize, however if proper incentives can be found to create such a PCN it is more likely that paths can be found to finish transactions. Such a network lends well to the merchants, as many different paths can be tried. The overall fees will be higher as most transactions will have a higher hop-count, but transactions will not be limited to certain links that have a high connectivity.

6 Methodology

Within this section a general overview will be given into the workflow of the simulation and how data was gathered. Limitations of the simulation will also be described and how they can be improved upon.

6.1. General Workflow

To evaluate the synthetic data-set and the Merchant algorithms, an analysis was done on a simulation of a PCN running SpeedyMurmurs. The process of gathering data for the analysis from the simulator can be described in four steps. These steps are as follows, *Generate Input Files*, *Initialize Simulation*, *Running Simulation* and *Data Processing*.

For each simulation that is run this process is followed. Generating input files entails three processes, the creation of a graph, allocating capacity to all edges in the graph and generating a stream of transactions for the simulation to run. These three distinct input files are needed to run a simulation. Allocating capacity to edges in a graph, creates the initial state of the channels. The initial state is determined by the channel distribution, as described in section 5.4, depending on a nodes degree a node is classified. Depending on the classification and the model used to generate the synthetic data-set edges are assigned capacity values based on a random distribution.

The transaction distribution encompass the firing-rate of transactions, the transaction value and the hop-count of the shortest-path a transaction must traverse over the network to get to the receiving node. Varying these values creates distinct different transaction profiles, the description of these different profiles came to be can be found in 6.3.2. The graph of the network is generated based off of one of two different topology models. The models are either *Scale-free* or *Erdos-Renyi*, further description can be found at 6.3.2.

The initialization of the simulation involves reading the three input files described, figuring out a nodes neighbours, loading in the transaction stream for each node and generating a spanning-tree. Once all of these actions are completed the simulation is in a state that it is ready to run.

While running the simulation runs the SpeedyMurmurs algorithm with different re-balancing algorithms. These algorithms are tested along side different transaction distributions, channel distributions and topologies. Data is collected during the running of the simulation, depending on the size of the network data is collected at different rates due to limitations in memory. Finally when the simulation is done all data is processed, the results of the data-processing can be found in the results section 7.

6.2. Technical Specs

The simulation runs with the OMNET++ networking framework, having written the simulation code in C++. To have enough memory the simulation was run on the DAS-4 (The Distributed ASCI Supercomputer 4). Running the simulation of the servers was automated by the use of bash scripts and python scripts. The input files were pre-generated and transferred to the servers, where the python scripts managed the directories and movement of files to allow the simulation to run.

Generation of the input files was written in python. Different libraries were used to help aid in the creation of these files. The most important libraries are the Networkx library, as it generates graphs using different models, and the Numpy library to easily make use of different probability distributions.

6.2.1. OMNET++ FrameWork

The OMNET++ networking framework is a discrete event-based simulator[74], with a huge library of tools to help in the making of a network, i.e. connections made in the network were based on a TCP connection coming from OMNET++ library. As the language used to program in is c++ the amount of memory the simulation requires can be managed and efficiently implemented.

Another great tool that is provided by OMNET++ is that it has a visual run-time environment, this visual environment allows for an easy understanding of how communication is working between nodes. While such an environment is not practical for a huge number of nodes, debugging and checking certain corner cases can be done in the visual environment to have a better understanding of how the network behaves. The simulator also makes use of '.ini' file types, that help configure the simulator. These files provide powerful tools to help run the simulation in different configurations to allow for quick runaround between simulations. OMNET++ also has graph generation capabilities, creating a special type of file that can be read. Though this is not used within this thesis, it was a tool that made OMNET++ appealing to use. The file can also be easily generated by other tools allowing for flexible use of networks.

Outputting data with OMNET++ can be done in multiple ways. As the framework is based on being a simulator, outputs can be scalar or vectors. A lot of the implementation is taken out of the hands of the programmer, allowing for easy and efficient mechanisms to store data. With proper pipe-lining and the use of Pandas in python gathering data is very flexible. However, the amount of data that can be collected is limited by the amount of storage available. Recording too many events brings its own problems of data-storage. For this reason mostly scalar values were stored.

6.2.2. Limitations in Implementation Simulator

OMNET++ is a very good framework to build networks on, as mentioned already mentioned. A limitation to the simulation implementation used in the evaluation of this thesis is the separation of creating transactions, the graph of the topology and link capacities. This design meant that these input files had to be read by the simulator, via non supported methods by the OMNET++ framework. This reduces the efficiency and is believed to be the cause of the limitation of nodes to 500.

Once the initiation of the simulation was finished, all transaction files were fed into the simulator. Having 500 nodes trying to read a file, process the file and store the important information was possible, once more nodes were added to the system the servers had memory allocation problems. It is believed that these issues come from reading the input files. While solutions can be thought of to fix this problem, running simulations in this set-up is not as streamed lined as can be. As input files need to be replaced before simulations can be run, and a lot of management of documents needs to be done.

From this learning experience, a recommendation can be made to not have the creation of input files be done outside of the OMNET++ environment but have it built into the simulator. If a seed is used to generate transactions during the initialization of transactions, the transactions should be the same across each simulation. This same code could be used to output the transaction-list created and that can still be evaluated.

6.3. Generate Input Files

To emulate the behaviour of users transacting on the network a framework is presented in chapter 5 for creating a synthetic dataset. Within the framework different parameters can be adjusted to allow for networks and the load on the network to display different different characteristics. These characteristics are dependent on capacity balance in the network and transaction value, and transaction rate going through the network. In the way the implementation works there are two separate mechanisms for formulating a channel distribution and a transaction distribution independently of one another. Details of how these are created can be found below, in there respective subsections.

Both the channel distribution and the transaction distribution are fed to the simulation during initialization. Having both these distributions in separate files allows for easy analysis of what is happening within the simulation. Once the simulation is completed verification can be done on the amount of transactions that should of been sent and analysis can be done on how the initial capacities changed from the starting state of the simulation to the end state of the simulation.

In the table 6.1 the different parameters that are used to dictate the generation of the input files are given. The type of value the parameter takes is also shown, and a description of the parameters. In the table under model, "LN" refers to the Lightning Model and "XRP" refers to the Ripple Model. The GRAPH_TYPE "SF" refers to a Scale-Free topology and "ER" refers to an Erdos-Renyi Topology.

Parameters	Inputs	Definition
NETWORK_SIZE	INTEGER > 1	Dictates the amount of nodes in the network
MODEL	"LN" or "XRP"	Dictates the model type for capacities and transactions
GRAPH_TYPE	"SF" or "ER"	Dictates the graph Topology
GRAPH_P	0 < DOUBLE < 1	Probability of links forming, only applicable to Erdos-Renyi
CAP_P	INTEGER > 0	Average value for a link capacity of a low-degree node
TRANS_NUM	INTEGER > 0	Number of transactions that need to be generated
TRANS_TIME	DOUBLE > 0	Maximum time in-between transactions
TRANS_P	INTEGER > 0	Average value of a low-degree node

Table 6.1: Parameters for creating input Files

6.3.1. Channel Distribution

Within the proposed synthetic data-set models not all nodes and edges are equal. This can be attributed to edges being bidirectional and certain edges in a network having a higher betweenness. **Betweenness**, indicates the centrality of a edge in a network, having a higher centrality occurs when a link has many shortest paths traversing it. An edge having a high betweeness can be seen as a bottleneck and allocating low capacity to such edges is naive, due to the usage of the link in the network. For that reason not all edges are distributed equally. For the creation of the channel distribution one must have an edge list, and be able to find the degree of each node that the edge is connected to.

Edges that are connected to two nodes with high degree can be classified as hub-nodes. The exact threshold for hub-nodes depends on the topology of the network and the size of the network. Links connected between two high degree nodes will need a higher capacity on the link, with the argument that these nodes in general are set-up by bigger players in the network. CAP_P is multiplied by a factor of 3 for hub-nodes. These bigger players anticipate that there edge will be used to transfer funds across the network and thus have a higher capacity, to allow the flow of transactions.

Edges connected to a node with a singular degree are referred to as *singular nodes*. These nodes represent players in the network that are seen as consumers. Due to the need for an initial deposit on a link, the assumption is made that these players most of the time generate a directional link towards the network. Thus a discrepancy is made between the bidirectional link capacity. Where one link will also be ensured capacity and the other may or may not get capacity depending on the distribution.

Certain nodes act not as consumers, however there degree is less than that of a hub-node. Due to the uncertain nature of these nodes. Edges in both direction that are connected to these nodes will get capacity.

6.3.2. Transaction Distribution

The concept behind the transaction distribution can be found in 5.3. Two different models are proposed, these models have different distributions for the value of the transaction. For the value of the transaction these values will be determined by the designated distribution for the model.

Nodes transact over the PCN with different nodes, due to general consumer behaviour that is elaborated on in 5.3. Having nodes randomly transact with different nodes will lead to a variation of long hop-counts and short-hop counts depending on the connectivity of the network. To have allow for more predictable hop-counts, a uniform distribution is used generate the hop-count that lies between $[2,6]$ hops. Then an analysis is done on the graph, determining all possible receiving nodes for the chosen hop-count. A random node is selected from the set of nodes that are a certain hop-count away. In that way greater control over the type of hop-counts for transactions can be determined.

The value of the transaction heavily depends on the model that is being emulated. Due to the exploratory nature of creating and testing the synthetic dataset, different profiles of the distribution are tested against a single set of a channel distribution. This gives insight into how different profiles of the distribution effect the throughput of the network.

As stated in the channel distribution, not all edges are created equal and their capacity may be limited due to a node being a singularity. As the model stands, nothing is done to counter act creating impossible transactions. A way to ensure that this is possible would be by implementing some sort of max-flow algorithm, or dictating the capacity of the channels from a set of transactions to ensure enough capacity is available. It has been chosen to keep the creating of the capacity and transactions independent for ease of use and fabricating channel capacities from a set of transactions can add a positive bias to the system in relation to success of

transactions. The implementation presented here is more pessimistic.

6.4. Topology Creation

The topology of a network influences the hop-count, the amount of paths that can be taken and the resilience against failures and attacks. Within the scope of this project no malicious adversary models are examined nor does the topology of the network lose and gain nodes during the running of the simulation. This is a large limitation in the emulation of the problem, however this simplifications help reduce the complexity of the simulation. This reduction allows for lower stabilization messages in the network and stability of transactions. If transactions fail during the *HTLC* due to a node going offline, such transactions get disputed on the level of the blockchain. Without implementing a simulation of the blockchain the reduction of this complexity fits the scope of the project.

6.4.1. Scale-Free

The creation of the scale-free networks is done using a python library *Networkx*. The parameters needed to create the network are based on:

Variable	Definition
N	Node Count
α	Probability for new node connected to an existing node, with in-degree distribution
β	Probability for adding an edge between two existing nodes. Based on in-degree and out-degree between connected nodes
γ	Probability for adding a new node connected to an existing node chosen randomly according to the out-degree distribution

The creation of the graph is based on the work of a Bollobas *et. al* in their paper Directed Scale-Free Graphs[18]. An important aspect when determining α, β, γ is that $\alpha + \beta + \gamma = 1$ and $\alpha > 0, \beta > 0, \gamma > 0$.

6.4.2. Erdos-Renyi

The creation of the Erdos-Renyi netowrk is done using the same python library *Networkx* as the scale-free model. An erdos-renyi graph is more random and has smaller variation in the degree distribution of nodes. Not creating the charateristic hub-nodes that can be found in graphs with a power-law distribution of the degree in nodes. Meaning graphs have a higher connectivity, with the draw-back that the hop-count can increase to trade between nodes. The parameters needed to create the network are as follows:

Variable	Definition
N	Node Count
p	Probability for edge creation.

The creation of this graph is based on the work of Erdos *et al.* in their paper, on the evolution of graphs [28]. Where the distribution of degree for nodes is created by a binomial distribution.

6.5. Initialize Simulation

During the initialization of the simulation, all nodes are created and connected to other nodes provided by the edge list that is provided to the framework omnetpp. The most important process of during the simulation is the creation of multiple spanning-tree's to allow for embedded based routing. After the spanning-trees is created, the root initializes a coordinate creation protocol and coordinate vectors are created for each node within that spanning-tree. These coordinates are shared along all connected neighbours of a node to allow for paths to make use of short-cuts. Once all spanning-trees are created, capacities have been loaded from the initialization file and the coordinates have been shared amound the network. The simulation will start processing transactions. During the initialization of the simulation a list of transactions is stored. The simulation starts once all nodes are part of the number of spanning-trees specified, all nodes have coordinates shared.

6.5.1. Spanning-Tree Creation

The spanning-tree is made using the a minimum-weight spanning tree algorithm proposed by Gallager *et. al* in their work on spanning-trees[33]. An important aspect of the creation of the spanning-tree relies on the

different tree-structures that are created during this stage. If all 10 spanning-trees have the same structure then a multi-path payment system is redundant because all paths converge to one. To ensure that the trees are different each instance of the spanning-tree generates a random value. These values are shared with the nodes neighbours and in response a weight for the link is created.

$$\left.\begin{aligned} Weight &= Node_{1,weight} - Node_{2,weight} \\ Weight &= Node_{2,weight} - Node_{1,weight} \end{aligned}\right\} \quad Weight > 0$$

Due to the randomization of the weights, the minimum-weight spanning tree will be different during every iteration of the algorithm.

6.6. Simulation of Payment Channel Network

During the simulation of a PCN events are triggered by either the transaction event list, that is created by the transaction distribution. Or messages are received from a nodes neighbours who are trying to transact. During a transaction a node can be labeled along the path as either *SENDER*, *FORWARDER*, *RECEIVER*. These nodes react different to incoming messages depending on their role in a transaction event.

Depending on the concurrency algorithm used, certain paths may give lower fees then others. The amount of fees collected of each forwarder node is stored. Due to the nature of the algorithms a nodes fees may be negative indicating that a lot of re-balancing was needed.

Transactions can face multiple issues, from not having enough capacity on a path to timing out. Paths that fail due to capacity issues will be closed and a new path will be tried. Once no more paths are able to be found the transaction cancels and is tried again at a later time point. The re-transmission time is in a higher order than the time it takes to complete a transaction. Transaction's must wait long enough for re-transmission for the balance of the network to be different, thus the re-transmission can not be done after a couple of seconds. If a timeout occurs all paths are closed and the transaction will wait for re-transmission.

6.6.1. Successful Transaction Messaging

The messaging protocol takes a total of 6 messages, 3 that start from the sender and 3 that start from the receiver. In figure 6.1 these messages are represented in the order they are sent, a forwarding node is not represented in the image. The forwarding node in this situation only forwards messages and locks the funds needed. The odd numbers in the illustration correspond to the sender messages and the even numbers correspond to the receiver messages. Before the sender sends the "Request Pend" message, it awaits answers from all the initial path requests, or till a timer runs out and it has received enough accepts to send out the "Request Pend" message. The receiver will only send out the "Pend Accept" messages along the paths when it can validate that enough funds have been locked to complete the transaction.

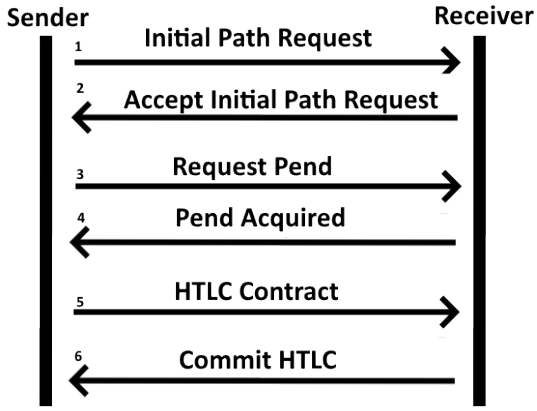

Figure 6.1: Messages that occur during a successful transaction

6.6.2. Non-Successful Transaction Messaging

The sender and the forwarding messages have a separate set of messages in path failing conditions. In figure 6.2 the different types of messages the forwarding node can send are given. These messages occur when the

path is unable to help aid the completion of the transaction. The forwarder node will send a "Path Request Denied" when the virtual capacity of a node is to lower than the requested amount, the mechanics of the virtual capacity are described in more detail in section 6.6.4.

A capacity error is sent when the node no longer has the funds on the link to complete the transaction. This can occur during concurrent transactions of the same link. When the sender link receives a capacity error, it will examine other paths that it has received accepts from or try in find a new path. If no paths exist the transaction will fail and the paths will be closed with the "Close Path" message. A "Timeout Error" message is sent when the forwarder has either accepted the initial path request or has pended the transaction amount and has not had a response within a time limit.

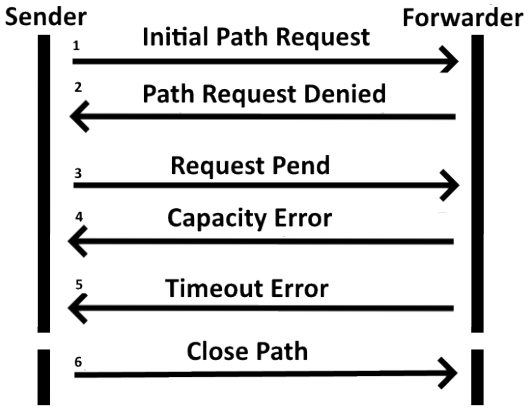

Figure 6.2: Messages that occur during a successful transaction

6.6.3. Sender Node

A *SENDER* is the starting node for a transaction. This node decides the number of paths the transaction will take, which spanning-tree routes each part of the transaction will take. The role of the sender is trying to minimize transaction fees, while trying to complete as many as transactions as possible. Within this implementation no limit is set on the amount of fees a transaction can have, thus all transactions are accepted. Only limited by the capacity on the link.

During first communication the coordinates of the receiving node is exchanged with the sender node, as well as the amount that will be transfer is sent. These transactions do not fail or timeout as such transactions are emulated to occur on the Internets infrastructure and do not have to take paths across the PCN network. Due to not having implemented this level of communication the latency of these messages is greatly reduced when sent along nodes to compensate for the message having to be passed along the network.

Once the coordinates are received, a set of random spanning-trees is chosen to initialize the payment over. For each outgoing message a latency is calculated, more on this in 6.6.6. If the sender receives all paths to be accepted, paths are choosen with the least amount of fees. Then a broadcast along the selected paths that ask nodes to pend the capacity is sent. Here paths can still fail if they are no longer able to acquire the requested pending capacity. If this occurs that path is closed and the sender node will try to request capacity along a different path, until it either succeeds or fails.

Once the capacity has been pended the transaction goes into its final stage and a HTLC contract is emulated. Verification of this contract gets sent along the path from the receiver to the sender, once all contracts are found by the sender the transaction is finalized and the path is closed.

6.6.4. Forwarder Node

A *FORWARDER* node forwards transactions along a path. A forwarding node plays an important role in a PCN, it must try and keep its links balanced in such a way that it is able to pass transactions along paths that make use of its links. A forwarding node will accept all transactions as long as it has the capacity available to it.

Forwarder nodes do not try to optimize for fees collection but rather link balance, secondly once a transaction is established and the capacity has been pended it follows a non-greedy application. A forwarder is non-greedy due to the way the pending mechanism works, if a payment is pended and it reduces link capac-

ity on a certain link, this capacity is subtracted from a **virtual capacity**. However if the transaction were to add capacity to the link, during pending this is not added to the virtual capacity. Giving the node forwarder a pessimistic view of its current link capacities.

6.6.5. Receiver Node

A *RECEIVING* this node receives payments, and will not pend any transaction as the link capacity will only increase once a transaction is completed. This node initializes the HTLC contract and will close all paths once it has received verification that all link capacities along the paths have successfully been updated.

6.6.6. Latency

Latency is a very important aspect when emulating a distributed network. Within this simulation three different latency's can be subjected to a message. These latency's are *Network Link Latency, Operating System Services* and *Cryptography Latency's*. It can be noted that these three latency's do not occur at the same-time point in a messages life, as network latency is a by product of sending a message across a link and OS latency occurs during the processing of a message. These latency's are all induced when a message is sent out from a node and is placed in a buffer waiting to be sent across the network.

Network Link Latency
Modeling network latency can not be encompassed within a single distribution. As presented by version in there latency statistics[9] the latency between packet delivery heavily depends on the length of the link that connects two nodes. Due to the added complexity of giving nodes a geo-location and deriving link capacities from that it has been opted to make use of a normal distribution for link latency. Europe as presented by version has a regional round trip time of 30ms, this is taken as the average value and the standard-deviation lies at 5ms. With such a standard-deviation latency's in the network will not be consistent, such differences in latency will allow message ordering to be scrambled. Scrambling message ordering will allow for a more realistic simulation, then if all messages are passed along sequentially.

Operating System Service
Within a network hardware will not be homogeneous, certain nodes will have hardware that takes away from professional internet services while others in a PCN may be running a home-server node. Without any reliable data on what type of nodes run what kind of hardware and the type of hardware that can currently be found within a pcn. It has been opted to use a normal distribution to emulate small OS delays into the message handling. The average time of computation will be estimated at 15ms, with a standard deviation of 10ms. Allowing for a broad-scope of latency's to be induced.

Cryptography Latency's
While considered as a latency that should be induced during the creation of the HTLC contract. In the work of Rifa-Pous *et al.* on the computation and energy costs for cryptographic algorithms on handheld devices[63], the computation time is less then 1ms. If handheld devices display such characteristics it may be inferred that computers in the nodes will also be able to handle such computations at least as fast thus creating negligible latency compared to that of the OS and Network link.

6.6.7. Finding Paths

SpeedyMurmurs makes use of embedded-routing along a trees path with short-cuts that can be found using vector coordinates. This ensures the shortest path possible for a certain tree is found between two nodes. Each node has a set of coordinates for each of its neighbours corresponding to different spanning trees. Before a node is sent along the path the coordinates are checked to ensure that the shortest path is taken along the spanning-tree path.

7

Evaluation and Highlighting the Interesting Bits

Within this chapter the method of evaluation is given, with the results run on the proposed experiments. All experiments are done with the use of the synthetic data-set generator and simulator. The chapter can be broken into four-parts, starting with an evaluation of the synthetic data-set generator then going into an evaluation of the synthetic data-set based on Lightning. Followed by the evaluation of the synthetic data-set based on Ripple and the last section will cover the results of the Merchants Algorithms.

In Section 1.2 two research questions are proposed, one of these questions is considering if it is possible to synthetically create a data-set for PCNs in such a way that allows for the emulation of an implementation of a PCN. The second questions considers if it is possible to increase the amount of successful transactions by having a rebalancing strategy for a set of transactions. For each question different evaluation methods have been considered. Most of the evaluation is centered around a single metric. The **Success Ratio**, the success ratio can be defined by the ratio of number of transactions that successfully completed measured against the total number of transactions in the simulation.

During all simulations where SpeedyMurmurs and the Merchant algorithms take part in. 10 spanning-trees are initiated during the initial-phase of the simulation. This was chosen to aid the diversity of trees in an Erdos-Renyi graph, the diversity of spanning-trees in a scale-free topology will be less. During the initial search the algorithms will query 6 tree paths and then select 4 different paths to take if enough paths are found that will allow the transaction to continue.

7.1. Description of Generated Topologies

The inter-connectivity of nodes is dictated by a PCNs topology. Within networks applied to PCNs, the topology plays a huge role in governing the way transactions are able to traverse a network. During the evaluation of the synthetic data-set and the Merchant algorithms the topologies generated play a role in the success of a transaction. For each set of an experiment topologies were randomly generated. To give a broad overview of the characteristic of these generated topologies an evaluation is done on the Scale-Free topology and Edros-Renyi Topology.

7.1.1. Methodology and Metrics

The method to generate these graphs is done by a python library networkx. Networks has a graph generation tool, generating directional graphs. To complete the graph all missing edges are added. These generated classes can be exported to the file system that omnet++ is capable of handling. To evaluate the graphs a couple of metrics are used, as seen in Table 7.1.

7.1.2. Experimental Setup

These graphs were generated for all all experiments running the scale-free topologies, for the evaluation of both the synthetic data-set model and the Merchant algorithms. Each of these experiments ran with 500 nodes. Each simulation was run a total of 10 times, the total amount of topologies that were compared is 70. For the generation of the Scale-Free model no input-parameters were given except the graph size. For the

Metrics	Definition
Nodes	Number of nodes in the network
Edge	Total number of bi-directional links in the graph
Diameter	Longest-shortest path in the network
Degree	Average number of links connected to a node
Clustering Coefficient	Measure of how nodes tend to cluster together in a graph
Node with 1 Link	Number of nodes with a single Link

Table 7.1: Definitions of metrics used in evaluation of generating topologies

generation of the Erdos-Renyi graph the probability for a degree to be generated between a node was 0.01, this value is above the correctness threshold meaning all nodes will have at least one edge in the graph. This probability for degree generation is also not so high that it does not allow for nodes to be connected with a single edge, allowing the graph to still have a relatively low average degree.

7.1.3. Results

In Table 7.2 the metrics of the Scale-Free graph are given. The total number of nodes with a single link is very high giving an easy indication that this is a Scale-Free graph. A Scale-Free is more formally defined by a power-law distribution for the nodes degree.

Average Attributes	Average	Standard Deviation
Nodes	500	0.0
Edge	817	39.46
Diameter	7.5	0.92
Degree	3.44	0.17
Clustering Coefficient	0.15	0.02
Node with 1 Link	282.43	8.67

Table 7.2: Average properties of the Scale-Free topology used in the experiments.

In Table 7.3 the metrics of the Erdos-Renyi graph are given. By law of large numbers an approximation can be given by the total number of edges a Erdos-Renyi should have. This approximation is given by Equation 7.1.

$$\binom{n}{2} * p \tag{7.1}$$

The resulting number from this equation is 1248 edges. The difference between the resulting graph is 22% on average. While the approximated value and the outcome of the number of edges from the graph do not perfectly match up, the difference is not so great that this cannot be considered an Erdos-Renyi graph. The number of nodes with a single link is low, indicating that most nodes have a higher degree. The average degree also indicates that this is the case.

Average Attributes	Average	Standard Deviation
Nodes	500	0.0
Edge	1529	30.47
Diameter	7.7	0.89
Degree	5.43	0.67
Clustering Coefficient	0.0082	0.0033
Node with 1 Link	18.7	3.2

Table 7.3: Average properties of the Erdos-Renyi topology used in the experiments.

7.1.4. Discussion and Conclusion

Looking at Table 7.2 for the Scale-Free graph and Table 7.3 for the Erdos-Renyi graph. Variance in the number of edges, can result in variations in the number of paths between two nodes, as less edges means less paths. The amount of edges in the Scale-Free graph is rather low compared to that of the Erdos-Renyi network as

seen in Table 7.3. With the amount of edges almost being doubled on average within the Erdos-Renyi graph. With such a lowe edge count, the underlying spanning-trees in the Scale-Free graphs, will have a similar structure compared to that of the Erdos-Renyi graph. While that also may be the case due to the inter-connectivity of the graph, less edges generally means less paths.

Another interesting aspect of general Scale-Free network is the amount of nodes with a single link. Nodes with a single degree are unable to help forward transactions only part-taking in a transaction if they are the receiver or sender. Such nodes will also have quicker transaction failures, as they only have a singular link that the transaction can traverse as the first step in a path. Limiting the amount of transaction value that can be sent over the link by the initial set-up.

In the table 7.3 all metrics have been aggregated of the used Erdos-Renyi graphs to an an average. The standard deviation in these values indicates that while the graphs were randomly generated similar graphs were used for the experiments. The edge count for these graphs is much higher than that of a Scale-Free graph, while the diameter of each graph is similar. Such a relationship shows that every node is as closely connected as in a scale-free topology but the number of paths between the two nodes can be assumed to be much higher.

The average degree of each node is also higher, with significantly lower nodes with a single link. In the case of Erdos-Renyi, the average degree gives a much better insight to the degree of an average node. Within the Scale-Free topology the average degree for a node is lower, and a significant amount of nodes have a single link. Such differences in degree, creates graphs with distinct different characteristics when it comes to path availability for transactions.

The results shows that the two graphs differ, and that both graphs can be classified as what they are intended to be.

7.2. Description of Data-Set Generator

The experiments were run against two different synthetic data-set models. The Lightning model, that was modeled after exponential distributions and the Ripple model that made use of a normal distribution. Within this section the different models will be examined, presenting the transaction and capacity distributions that were created for the experiments. These distributions represent the set of distributions that were not changed during the experiments, as these sets were run against there counter-part parameter.

7.2.1. Methodology and Metrics

This evaluation is run on the transaction and capacity distributions that were used in the evaluation of the synthetic data-set. To generate a transaction distribution, the number average value of a transaction is used to center the distribution around. Also the nodes are needed to distinguish the amount of edges they have. To generate a capacity distribution, the average value of a capacity needs to be defined and a node list to generate the capacities. This process is better described in Section 6.3.1.

7.2.2. Experimental Setup

The values to generate the experiment are shown in the Table 7.4. A distinction is made between different classes of nodes, as different classes interact with the synthetic data-set model differently. 'Small-nodes' are nodes that have a degree of 2 or 1. Such nodes will rarely have to forward nodes. Between a degree value of (3,8) nodes are considered 'Medium-Nodes' and any node with a degree above 9 is considered a 'Big-Node'. 'Big-Nodes' can also be classified as hub-nodes, and will be utilized a lot to run transactions through.

	Transaction Distribution	Capacity Distribution
Parameters	Value	Value
Nodes	500	500
Average Value	[200,400,600]	[600,800,1000]
Small-Node	2	2
Medium-Node	8	8
Big-Node	9	9

Table 7.4: Average properties of the Erdos-Renyi topology used in the experiments.

7.2.3. Results

For the Lightning model, in the experiments for 'Capacity Distribution vs a set of Transactions'. The figure 7.1 represents the average transaction values used for the experiment. It can be clearly seen that these transactions are modeled after the exponential distribution. As one expects the lower the transaction value probability is, the more the transaction value occurs closer to the origin.

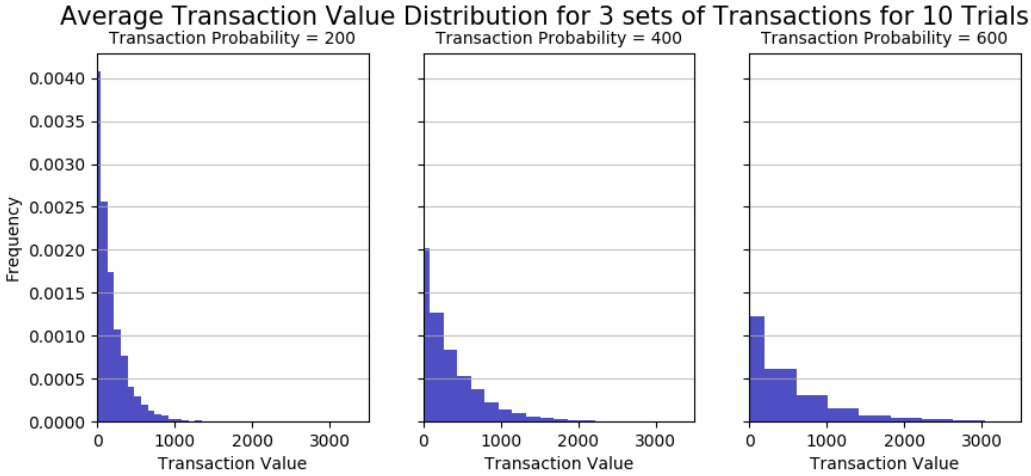

Figure 7.1: Transaction Value Distribution for three different Transaction Value Probabilities

For the experiments concerning the Lightning model, where the channel distribution is kept the same across the experiment a representative of the distribution can be found in figure 7.2. Also this graph represents an exponential distribution.

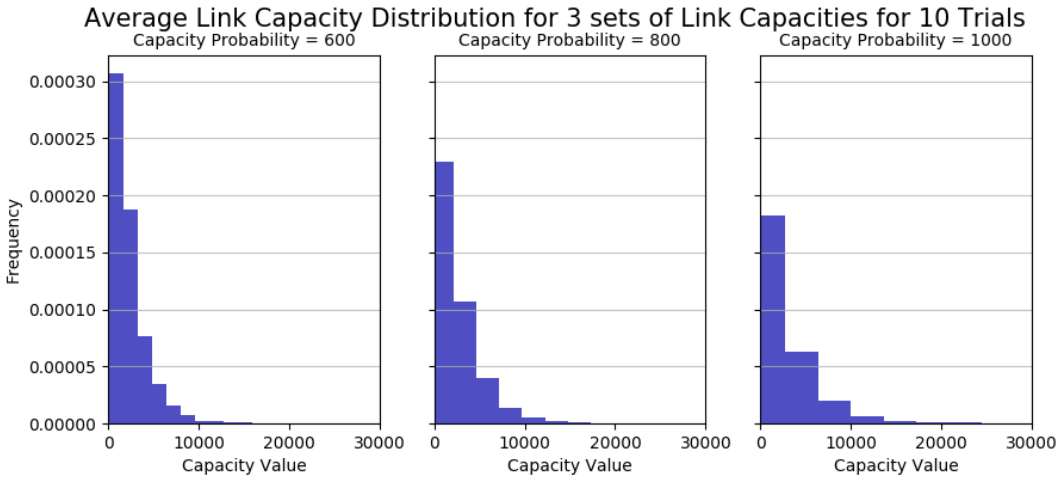

Figure 7.2: Channel Distributions for the 3 sets of Channels

The transactions produced by the Ripple model can be seen in figure 7.3. These transactions are modeled after a normal distribution and from the figure 7.3 the bar graphs do illustrate a normal distribution. Each of the normal distributions center lies around the average transaction value it was modeled after. It is interesting to note that these distributions at a value of 400 and 600 do not have transactions that are near the origin of the graph.

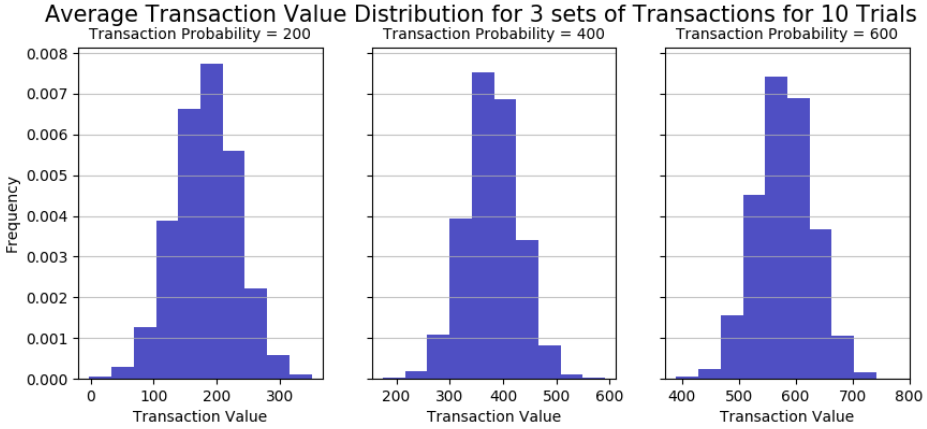

Figure 7.3: Transaction distributions for the 3 sets of Channels modeled after the Ripple Model

The shape of the channel distribution from the Ripple model represented in 7.4 is not simply a normal distribution. This is caused by the way the model was created to take into account nodes with different degrees. Nodes with a lower degree were given a lower value to model the capacity around than that of higher degree nodes. This was done to ensure that hub nodes, would have enough capacity in relation to the singular nodes to handle multiple transactions at once.

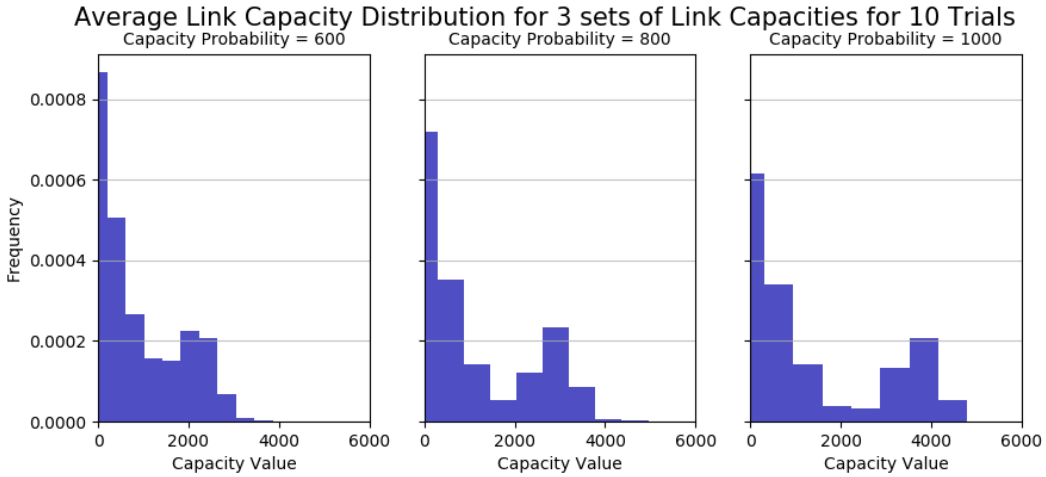

Figure 7.4: Channel distributions for the 3 sets of channels modeled after the Ripple Model

7.2.4. Discussion and Conclusion

For the both of the distributions the Values for the channel distribution have a much higher overall value than that of the transactions. This is influenced by two factors, in general the average transaction value is modeled lower than that of the average capacity value. The second factor stems from how the model differentiates between nodes with low, medium and high degree. Nodes with a higher degree get a higher average capacity to ensure that the edges mostly used in transferring transactions have enough capacity to handle more than one transaction.

Within the instance of the Ripple model a characteristic of these transaction values is that much less values occur near the origin and are centered around the transaction probability they are meant to represent.

With more centrality, the system is transferring more similar transactions across the network compared to that of the Lightning model for transactions.

A benefit to this model may be that noticeably less high values are trying to be transacted, in the Lightning transaction model for a probability of 200, transaction values exceed 1000. While for the Ripple model it can be seen the higher transaction value is below 400. While the opposite may also be said as the amount of transactions valued around 0 are significantly less in the Ripple model than that of the Lightning Model. Such discrepancies in characteristic may be the cause that lead to the different results from the data-set evaluation experiments.

The Figure 7.4 displays the distribution from these set of channel distributions modeled after the Ripple distribution. The figure have some slight characteristics of the capacity distribution from the Lightning network in figure 5.2. While not a perfect replica by any means, it does hep indicate that some of the assumptions may conform to what is happening in the actual Lightning network.

The transaction distributions represent the distributions they were modeled after, for the capacity distributions this holds the same. The results help validate the modeled data-set transaction and capacity distributions are modeled as expected.

7.3. Description of the Concurrency in the Networks

Within this section an evaluation will be done on the concurrency of transactions in the network.

7.3.1. Methodology and Metrics

To ensure that concurrency is happening an evaluation is being done on the network and the average number of transactions a node is handling during a simulation. Over time this number fluctuates during a simulation, for that reason the average is taken over the starting time of when transactions are sent across the network after the initialization of the system has finished. Till all transactions have been sent. The values being examined came from the simulations running the evaluation on the synthetic data-set.

The concurrency average is also influenced by the graph. In the Scale-free graph, not all nodes are considered for this average value. This is largely due to the bias introduced by the nodes with a single degree. As they are unable to help aid in transactions and can only receive or send transactions, and seeing as in a Scale-Free graph these nodes are most abundant they heavily influence the average. No distinction is made between the XRP model and the Lightning Model when it comes to transaction concurrency as both models have the same firing rate for these simulations thus no tangible difference can be seen.

7.3.2. Results

For this reasoning they have been removed from Table 7.5 and only the nodes with a degree of 2 or higher are considered.

TRANS_P	Nodes Considered	Average	Standard Deviation	Max
200	120	6.9	3.7	29
400	123	6.1	2.9	24
600	112	7.7	4.3	33

Table 7.5: Number of transactions a node is handling at one time with a Scale-free Graph

In Table 7.6 the concurrency is given for a transaction set running on an Erdos-Renyi graph. All nodes are considered in this evaluation of concurrency due to each node playing a larger role in transactions that in a Scale-Free graph.

TRANS_P	Nodes Considered	Average	Standard Deviation	Max
200	500	3.5	1.9	21
400	500	4.2	1.7	18
600	500	3.4	2.3	25

Table 7.6: Number of transactions a node is handling at one time with an Erdos-Renyi Graph

7.3.3. Discussion and Conclusion

Within the Scale-Free transaction models, a higher concurrency of transactions crossing the same nodes can be found compared to that of the transactions going over the Erdos-Renyi network. This can be expected as more nodes are considered for the averages in the Erdos-Renyi graph than that of the Scale-Free graphs. Also the topology of the network will play a large role, due to the way the networks are laid out.

7.4. Description of the Lightning Model

Within this section the results for the synthetic data-set modeled around the Lightning model will be presented. A broad overview of the way the synthetic data-set behaves under various conditions under the SpeedyMurmurs algorithm is presented. Values for the static sets of distribution the experiments compare against have been chosen to mimic realistic values. Without validation it cannot be confirmed these values are realistic.

This section can be broken down into two parts, as two different topologies were used to conduct the experiments. The first section covers the Scale-Free topology, a topology that occurs more often in the natural world. While the second topology is the Erdos-Renyi topology, with a more evenly distribution of edges and nodes. These types of topologies are harder to establish.

7.4.1. Methodology and Metrics

The evaluation of the synthetic data-set created by the proposed Lightning Model will used during the evaluation. The value of transactions in this model are modeled after an exponential distribution and the capacity of links in this model are also modeled after the exponential distribution. These distributions can be seen in Section 7.2.3.

Evaluation of each topology with Lightning model will be done in two parts. The first part is the evaluated by taking a fixed set of link capacities as shown in Section 7.2.3 and a graph topology where some metrics can be found in Section 7.1.3. This is then compared to a larger set of transactions that have a varying TRANS_P value. The metric used to evaluate these simulations will be the success ratio. The success ratio will be plotted in a graph to give a depiction of how the success ratio changes according to different average values of transactions compared to a fix capacity.

The second evaluation will be the counter-part, taking a fixed set of transaction value distributions as seen in Section 7.2.3 and a constant topology while changing the link capacities. The metric used to compare the fixed set of transactions vs the average link capacity, CAP_P, allowing for insights into how the influence of average link capacity has an effect on the effectiveness of transactions. These two evaluation methods for the Lightning model will be compared to two topologies.

7.4.2. Experimental Setup

The values for the distributions can be found in the experiment set-up in Table 7.4. During the simulation 28 different points were examined, these incrementally increase in value.

7.4.3. Results Scale-Free

Within this subsection the success ratio for the Lightning synthetic data-set is presented while the data-set is run on an Scale-Free network.

Transactions vs a Set of Link Capacities

The success ratio of different transactions modeled against a set of channel distributions can be seen in Figure 7.5. The figure illustrates that the average success rate is higher when the average capacity is higher. This means that more capacity is available to the network, allowing for a higher chance of transactions finishing. It is interesting to note that while the higher the average capacity has better results overall, it performs only slightly better when the transaction average value is low or very high.

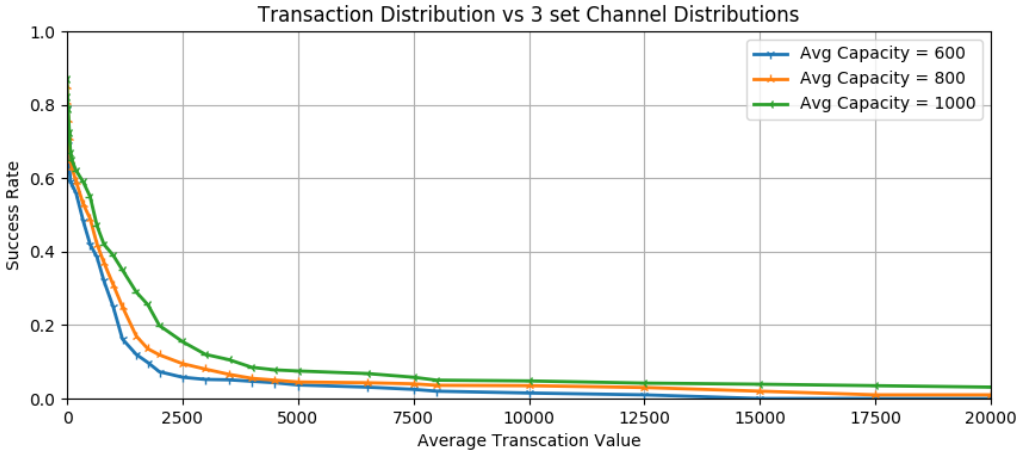

Figure 7.5: Transaction value averages vs three sets of channel distribution with the Lightning model in a scale-free topology

Within Figure 7.6 the error bars are given by the standard deviation of the aggregated 10 simulations done, from the figure it can be seen that the average error is some-what constant. This figure also illustrates that between the values [250, 1500] it seems as if the success rate has can be plotted with a trend line.

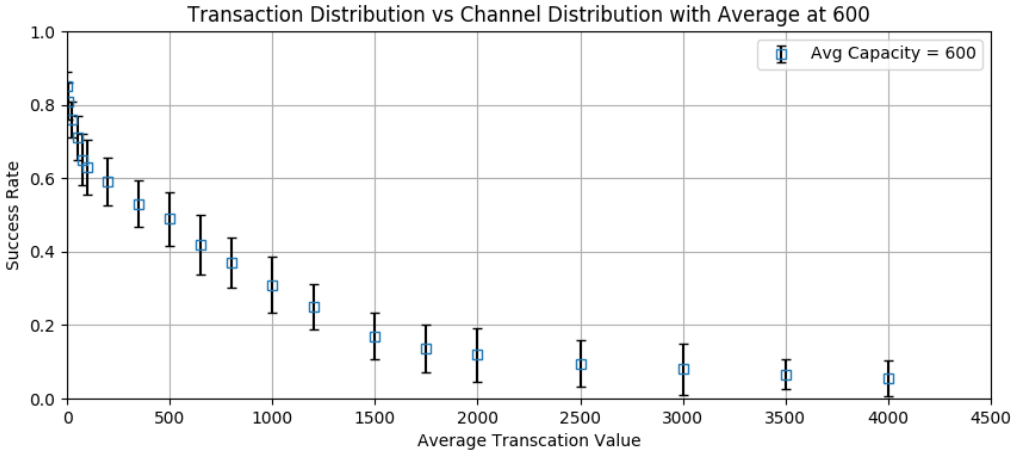

Figure 7.6: Transaction value averages vs a channel distribution with an average of 600 using the Lightning model in a scale-free topology

Link Capacity Averages vs Three Sets of Transactions
Comparing the channel distribution to that of a set of transactions for a scale-free topology with the Lightning model can be found in figure 7.7. The blue line in this figure represent an average transaction value of 200, and it can be seen that the lower the average transaction value is the better the model performs. Within the context of this simulation there seems to be some upper limit to how well the model is able to perform.

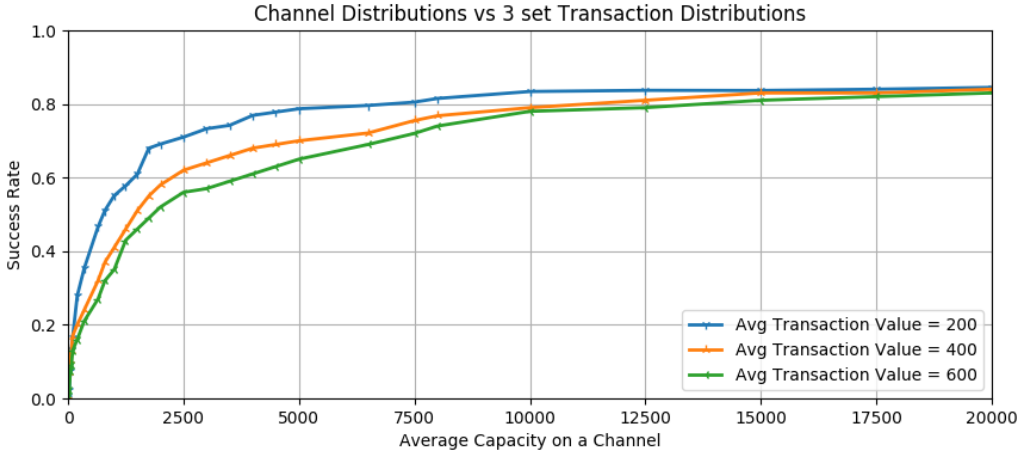

Figure 7.7: Channel distributions vs three sets of transaction distributions with the Lightning Model in a scale-free topology

In figure 7.8 the error bars are once again given by the standard deviation of 10 trials run for this experiment. Within this figure it can be seen how the relation holds between the average transaction value and the average capacity on the channel. The success rate increases rapidly as the average capacity on a channel increases, however this increase in success rate slows down when the average capacity is about 10 times that of the average transaction value.

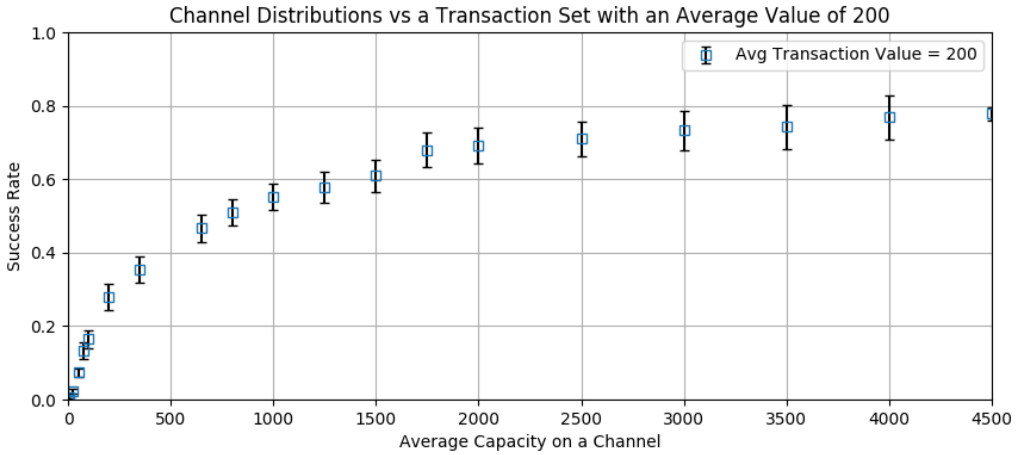

Figure 7.8: Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a scale-free topology using the Lighting model

7.4.4. Discussion Scale-Free

An interesting aspect to examine in Figure 7.5, is that with all three sets of channel distributions, the lowest transaction probability value does not allow for 100% success rate of transactions. While it is not clear what the reason for this is, it may be caused by the amount of singular nodes in the Scale-Free network. As nodes are chosen at random to transact, eventually nodes with a single degree may deplete there link capacity, thus any subsequent transactions will fail.

This does not always have to be the case if a node receives a transaction. Though due to the random nature of the generation of transactions, it seems that nodes try to spend outside of there link capacity without the

link capacity being re-balanced. A method to ensure such transactions are not allocated would be to run a check before transactions are allocated. However, due to the endless amount of possibilities this becomes an NP hard problem.

There seem to be some linear decay in the relationship between increasing transaction distribution vs a fixed capacity distribution. However the values near the origin for the transaction distribution show significant decrease in success rate, and around 1500 a limit seems to have been reached. While it is not impossible to imagine a 0% success rate, achieving this with random distributions is difficult.

Within Figure 7.8, a limit of around 80% is found, this limit is something that was also seen when the transactions were compared to the set of capacities. Further indicating that something within the network is limiting the amount of successful transactions possible.

As one may expect the success ratio for transactions increases as the probability for higher capacities in the distribution increase. The success ratio increase quicker with a lower transaction value distribution, meaning on average the transactions have lower value. Such a result is to be expected from a network.

Within Figure 7.8, less of a linear pattern can be seen between the points. Possibly indicating that the relationship between capacities vs transactions is not fully linear. As multiple paths are taken to complete a transaction, having slightly higher link capacity values may be more beneficial to the success rate till a certain percentage. Within the figure 7.8 after around a 70% success ratio the increase in success ratio needs a much higher change in the channel probability then before, indicating that increasing the capacity values after a certain point in this network gives much smaller return to success.

7.4.5. Results Erdos-Renyi

Within this subsection the success ratio for the Lightning synthetic data-set is presented while the data-set is run on an Erdos-Renyi network.

Transactions vs a Set of Link Capacities

Within the Figure 7.9, the transaction distribution vs 3 sets of channel distributions can be seen running on an Erdos-Renyi network. When using the Erdos-Renyi graph the initial success rate of the PCN seems to increase, indicating that the more even distribution of degrees in nodes positively effects the success rate. In this model of the PCN the higher average of capacities on the links the better the performance, such a relation is expected between transaction value and capacity value.

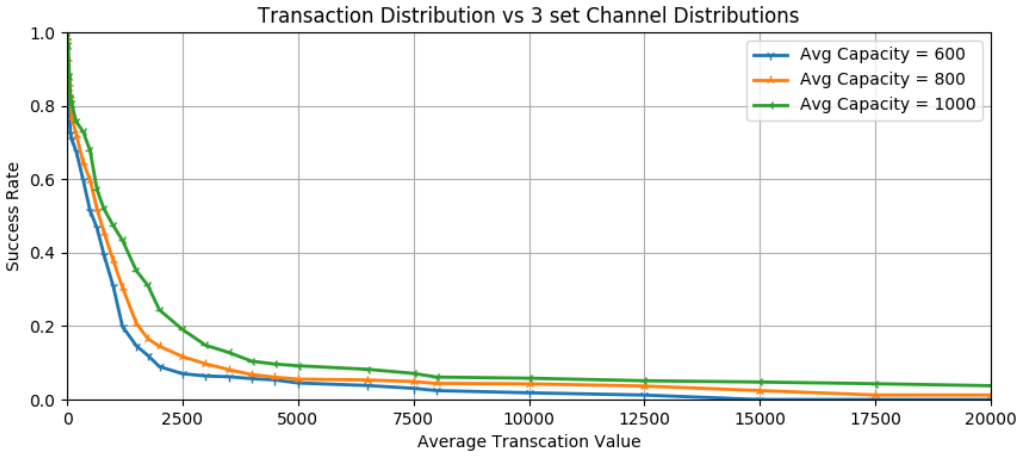

Figure 7.9: Transaction value averages vs three sets of channel distribution with the Lightning model in a Erdos-Renyi topology

In figure 7.10 the error bars are once again given by the standard deviation of 10 trials run for this experiment. When the average transaction value is really low in this simulation the success rate is able to complete. This success rate degrades rather quickly as the transaction value increases, then once again a steady decrease in success rate can be seen. Till the model seems to hit a limit and slowly converges to 0.

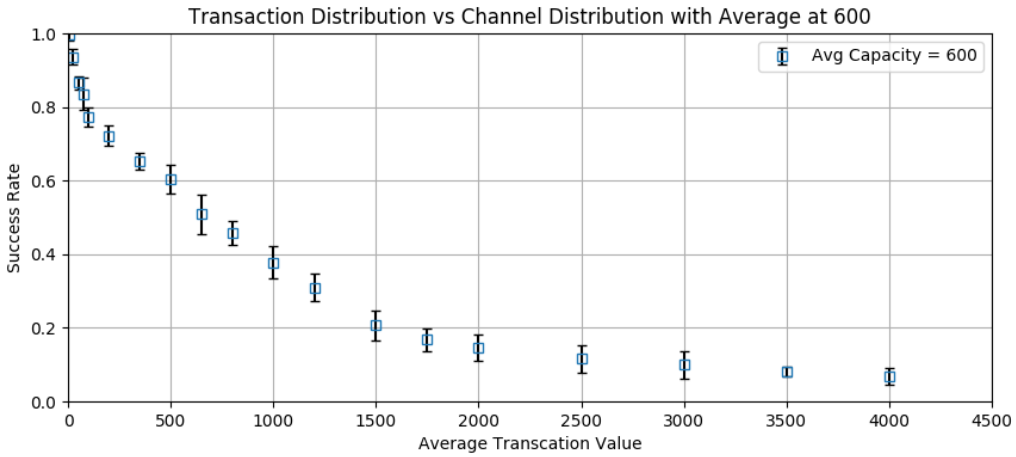

Figure 7.10: Transaction value averages vs a channel distribution with an average of 600 using the Lightning model in a Erdos-Renyi topology

Link Capacity Averages vs Three Sets of Transactions

Changing the channel distribution value against a set of transactions for an Erdos-Renyi graph is represented in the figure 7.11. As the average capacity of a channel increase so does the success rate, while the figure does not reach 100% success rate, the lines for all three transaction values seem to generally increase. While not certain, with a high enough average value for capacity the success rate may become stable at 100% for the set of transactions.

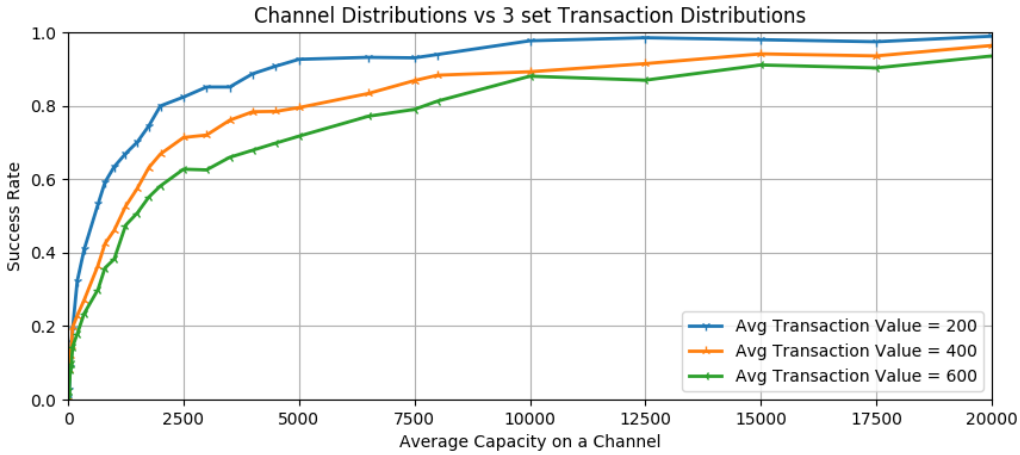

Figure 7.11: Channel distributions vs three sets of transaction distributions with the Lightning Model in a Erdos-Renyi topology

In figure 7.12 the error bars are once again given by the standard deviation of 10 trials run for this experiment. It can be clearly seen that with an almost average value of 1 for the capacities the network is unable to process a lot of transactions. This has to do with the discrepancy with between the transaction value and the value available to transactions. The Erdos-Renyi graph is able to quickly achieve some success as around an average of 500 the success rate is almost 50%.

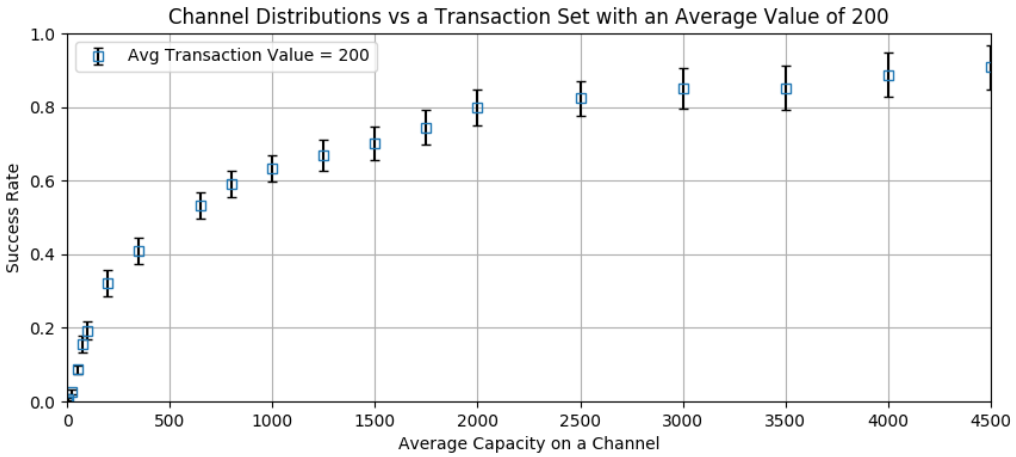

Figure 7.12: Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a Erdos-Renyi topology using the Lighting model

7.4.6. Discussion Erdos-Renyi

Within the Figure 7.9, the success ratio is almost 20% higher initially, compared to that of the Scale-Free topology. This may be attributed to the fact more connections and paths exist, allowing for transactions to have a higher chance of success. Within the experiments done a 100% success rate does not seem to be possible in the Lightning model, this may be due to the way the exponential distribution works, as a wider range in transaction value is possible compared to that of the used normal distribution for these models.

Comparing figure 7.6 with figure 7.10 the results seem very similar. Taking a look at a specific value like $t_p = 1500$, for the Scale-Free graph the success rate is below 20%. While for the Erdos-Renyi graph the success rate is slightly higher and above 20%. Such small differences indicate while an Erdos-Renyi graph may have more paths, and return slightly better results. Overall the difference is minimal.

As the Channel Distribution vs a Set of Transactions is related to the Transaction vs A set of Channel Distributions, the results show slightly better performance than compared to the scale-free graph. The limit for success rate for the set of transactions in the simulation also hovers above 80%, similarly found in the Scale-Free Graph.

7.4.7. General Discussion and Conclusions

The proposed synthetic data-sets models are currently the only known models for the creation of data-sets concerning PCN. As the only other data-set for a PCN is a Ripple dataset where the validity of the data-set is questionable as mentioned in Section 3.1.5. Without other data-sets, comparing the synthetic data to current PCN implementations and trying any type of validation becomes problematic. Validation does not need to be limited to mimicking actual PCNs.

Without being able to compare to actual implementations, comparing the firing-rate of transactions and the occurrence of concurrent transactions effecting each other is difficult. The simulation does showcase that transactions can be completed and that depending on the ratios between the value of the transaction and the link capacities the success rate of transactions changes. The simulation is also able to handle transactions failing, and is able to handle concurrent transactions.

7.5. Description of the Ripple Model

Within this section the evaluation of the synthetic data-set that is modeled after Ripple will be presented and evaluated. This will be done for both the Scale-Free topology and the Erdos-Renyi topology. Differences will be examined between the two topologies.

7.5.1. Methodology and Metrics

The evaluation of the synthetic data-set created by the proposed Ripple Model will used during the evaluation. The value of transactions in this model are modeled after an exponential distribution and the capacity of links in this model are also modeled after the exponential distribution. These distributions can be seen in Section 7.2.3.

Evaluation of each topology with Ripple model will be done in two parts. The first part is the evaluated by taking a fixed set of link capacities as shown in Section 7.2.3 and a graph topology where some metrics can be found in Section 7.1.3. This is then compared to a larger set of transactions that have a varying TRANS_P value. The metric used to evaluate these simulations will be the success ratio. The success ratio will be plotted in a graph to give a depiction of how the success ratio changes according to different average values of transactions compared to a fix capacity.

The second evaluation will be the counter-part, taking a fixed set of transaction value distributions as seen in Section 7.2.3 and a constant topology while changing the link capacities. The metric used to compare the fixed set of transactions vs the average link capacity, CAP_P, allowing for insights into how the influence of average link capacity has an effect on the effectiveness of transactions. These two evaluation methods for the Lightning model will be compared to two topologies.

7.5.2. Experimental Setup

The values for the distributions can be found in the experiment set-up in Table 7.4. During the simulation 28 different points were examined, these incrementally increase in value.

7.5.3. Results Scale-Free

The results presented within this subsection cover the Scale-Free topology for both types of experiments transaction vs set of channel distributions and channel distributions vs a set of transactions.

Transactions vs a Set of Link Capacities
The success ratio of different transactions modeled against a set of channel distributions can be seen in Figure 7.13. In the scale-free topology with an XRP model the success rate is above 80% for the the smallest values of the transaction distribution. The figure also shows that with a high enough average transaction value the simulations success rate slowly goes to 0.

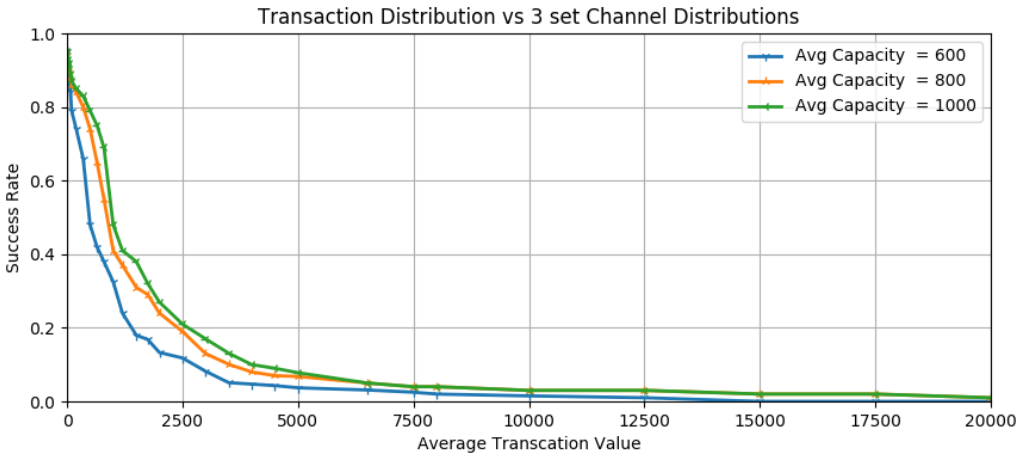

Figure 7.13: Transaction value averages vs three sets of channel distribution with the Ripple model in a scale-free topology

Within figure 7.14 the error bars are given by the standard deviation of the aggregated 10 simulations done. The success rate never achieves 100% but is very close when the average transaction value is 1. A jump in performance is seen between the average value of 350, and 500. Once the average value of the transactions close in on the average capacity value the success rate reduces quickly.

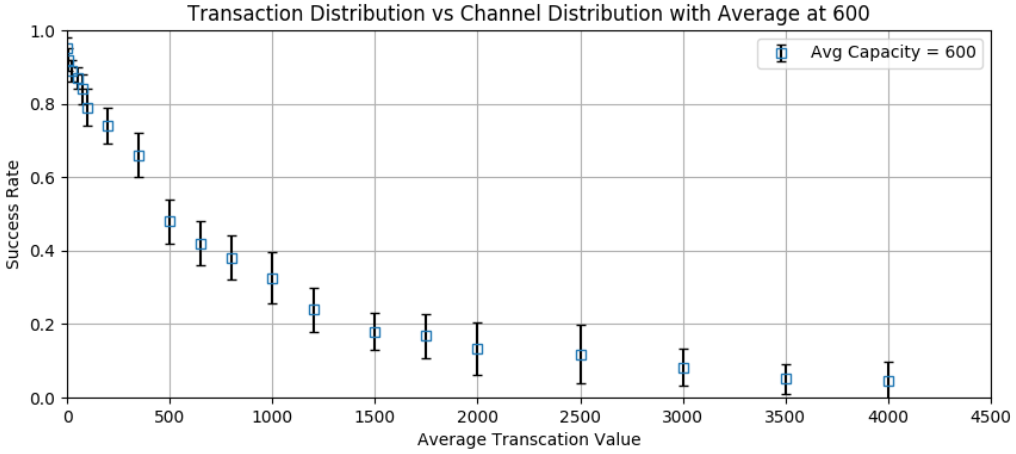

Figure 7.14: Transaction value averages vs a channel distribution with an average of 600 using the Ripple model in a scale-free topology

Link Capacity Averages vs Three Sets of Transactions
Comparing the channel distribution to that of a set of transactions for a scale-free topology with the Lightning model can be found in figure 7.15. The figure illustrates that there is a bound on the success rate of the simulation in the case of the scale-free model. The three transaction success-rates are fairly close to one another, showing a close relation between the normal distributions and the success rate.

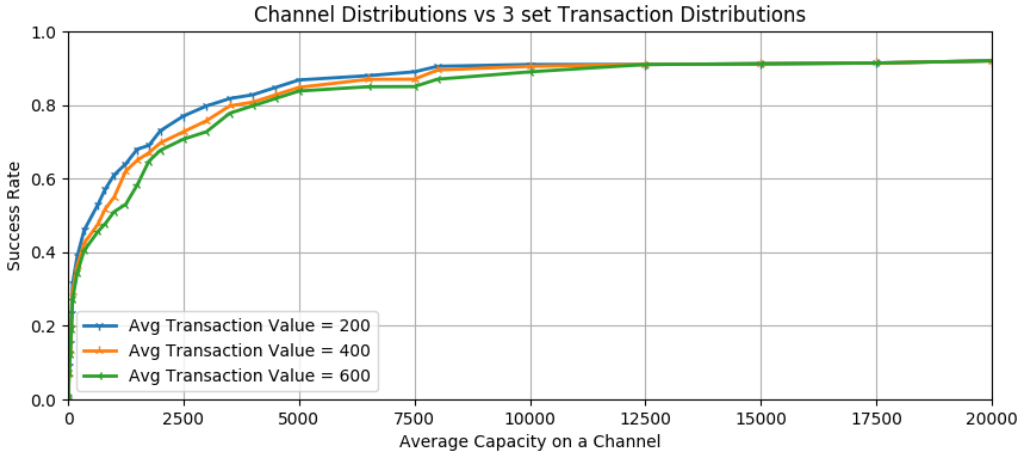

Figure 7.15: Channel distributions vs three sets of transaction distributions with the Ripple Model in a scale-free topology

In figure 7.16 the error bars are once again given by the standard deviation of 10 trials run for this experiment. The success rate increases fairly quickly as the average capacity on the links increases, at around 450 average value for capacity the transactions almost succeed 50% of the time.

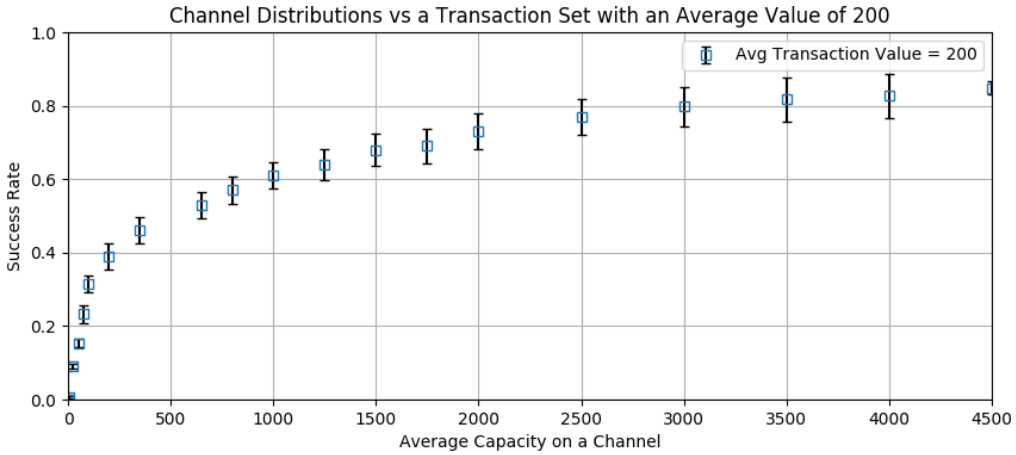

Figure 7.16: Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a scale-free topology using the Ripple model

7.5.4. Discussion Scale-Free

The Ripple model with a Scale-Free topology seems to be limited at around 90%. As seen in figure 7.15, no matter the increase in channel capacity through-out the network certain transactions will not complete. This limitation may be attributed to the way the Scale-Free capacities are distributed depending on if a node is singularly connected to a hub or not. As in trying to model consumer behaviour, some capacities are not initialized in both directions, possibly excluding some nodes from transactions no matter the capacities found within the network.

7.5.5. Results Erdos-Renyi

Within this subsection the transaction vs a set of channel distributions will be explored for the Erdos-Renyi topology using the Ripple Model. The success ratio for channel distribution vs a set of transactions is also explored.

Transactions vs a Set of Link Capacities

Within the Figure 7.17, the transaction distribution vs 3 sets of channel distributions can be seen running on an Erdos-Renyi network. The success rate in this configuration of the simulation allows for 100% success rate, this can be heavily attributed to the fact nodes have on average a high degree. Allowing for multiple paths to be found.

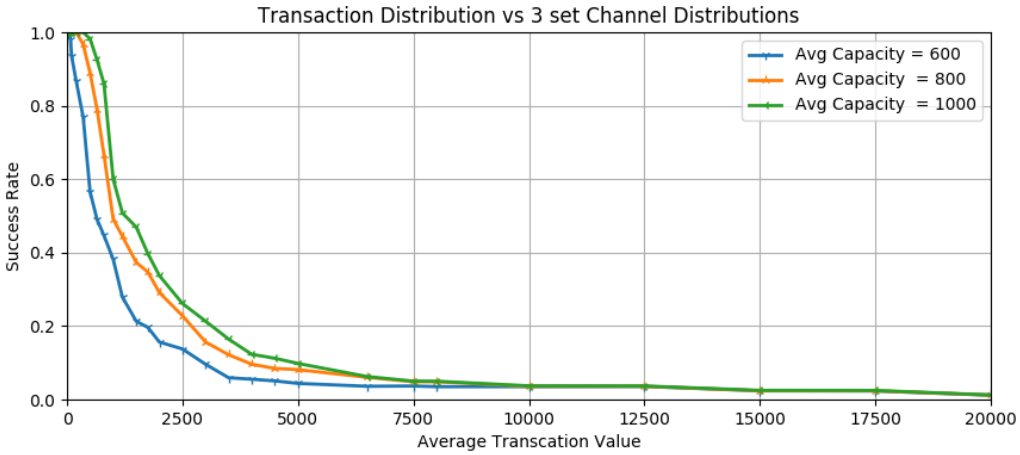

Figure 7.17: Transaction value averages vs three sets of channel distribution with the Ripple model in a Erdos-Renyi topology

In figure 7.18 the error bars are once again given by the standard deviation of 10 trials run for this experiment. A jump can be seen in the success rate as the transaction value approaches the average capacity value. Within the context of this simulation it seems that an equal value for capacity and transaction allows for roughly 50% of transactions to complete.

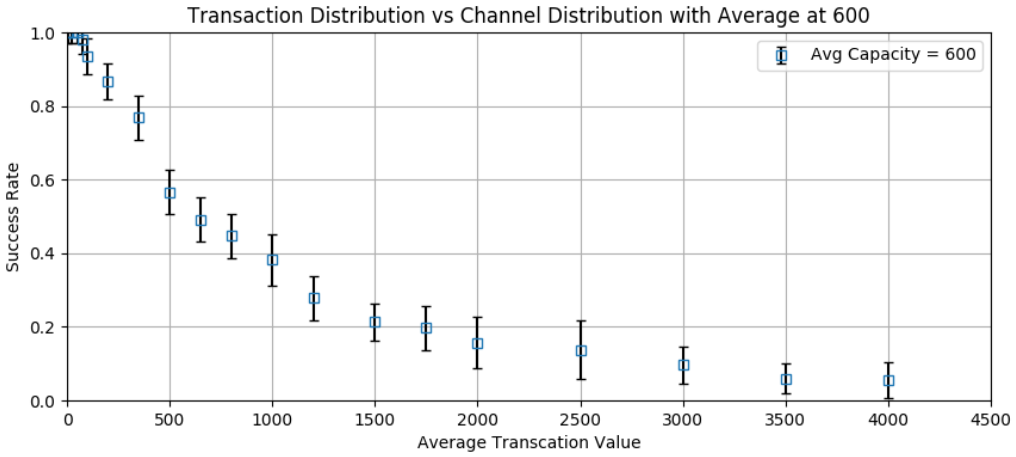

Figure 7.18: Transaction value averages vs a channel distribution with an average of 600 using the Ripple model in a Erdos-Renyi topology

Link Capacity Averages vs Three Sets of Transactions
Changing the channel distribution value against a set of transactions for an Erdos-Renyi graph is represented in the figure 7.19. The success rate of this simulation is the highest of all simulations discussed within this chapter, for that reason the x-axis has been limited to 10000 as the simulation will continue to have 100% success. It appears that an Erdos-Renyi graph allows for good transaction success rate if both the capacity and the transactions are modeled around a normal distribution.

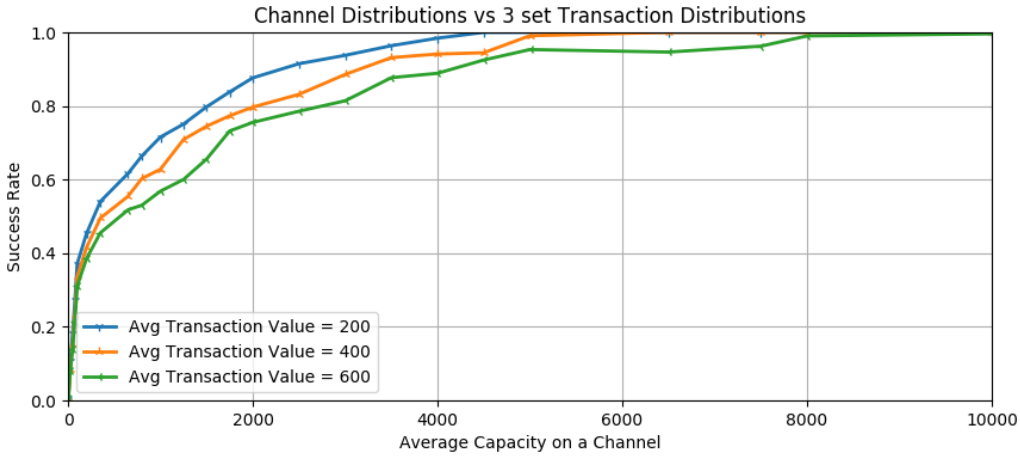

Figure 7.19: Channel distributions vs three sets of transaction distributions with the Ripple Model in a Erdos-Renyi topology

In figure 7.20 the error bars are once again given by the standard deviation of 10 trials run for this experiment. In this graph it can be clearly seen that again with an equal value of capacity and transaction value the simulation is able to achieve almost a 50% success rate. While not the case in all simulations, this set-up of normal distributions and the Erdos-Renyi topology lends its self well to high success rates.

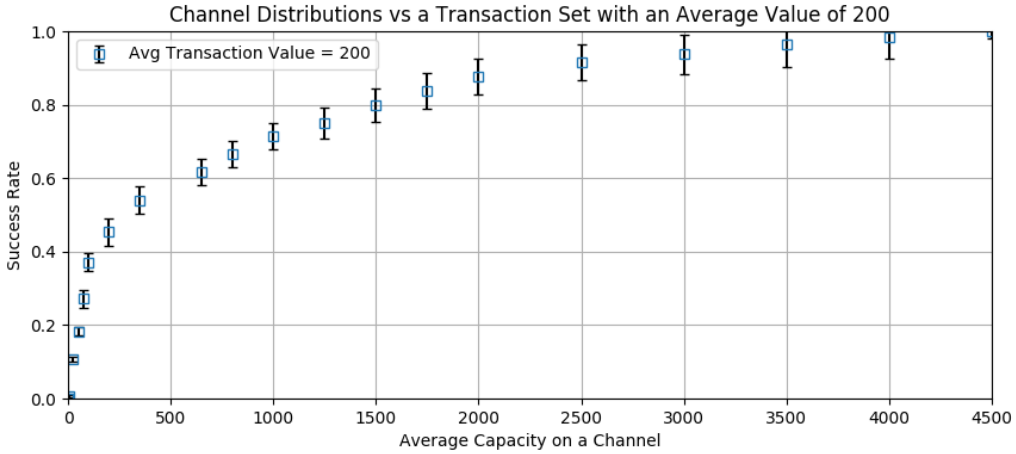

Figure 7.20: Channel distributions vs a transaction value distribution where the average value of the transactions is 200 in a Erdos-Renyi topology using the Ripple model

7.5.6. Discussion Erdos-Renyi

An initial impression of figure 7.17 shows that 100% success rate is possible within the simulation, though the right conditions need to be met. The connectivity of Erdos-Renyi graph and the normal distribution of values generate enough capacity on the lines for micro-transactions to successfully go through the network. As the transaction value increases the success rate decreases. As one would except.

Looking at figure 7.18, with $cP = 600$ and $tP = 600$ the success rate is around 50%. Indicating a balance between the two values. Such a result is a little surprising as SpeedyMurmurs makes use of multiple paths,

one could conceive that less capacity is needed on average to transact certain values as in an Erdos-Renyi graph, two transaction paths do not have to share any of the same links.

The characteristics from changing the channel distribution vs a set of transactions in figure 7.19 show very promising results for graphs with a tp/cp ratio of around 600/2000 as 80%. Changing the channel distribution also influences the rate the success ratio goes to 100%. This may be attributed to the more equal spread of degree and nodes in the generated topology, meaning that most nodes will have capacity going over both sides of a link during initial setup. Having initial bidirectional links, increases the sum of capacity in the network compared to that of a Scale-Free topology, also the Erdos-Renyi topology has double the amount of links as the Scale-Free topology so more capacity can be found in the network in general.

7.5.7. General Discussion and Conclusion

As mentioned in Section 7.4.7 comparing these results to implemented PCN is not possible. The different Synthetic data-set models show different success rates within the simulation. Where Ripple Model slightly outperforms the Lightning Model in the success rates of transactions. The Ripple Model was able to achieve 100% success rate. Under certain conditions in an Erdos-Renyi graph it is able to sustain a 100% success rate with a higher value of transaction vs link capacity ratio than in any of the other models and graphs. Helping solidify that within this simulation, the results indicate that an Erdos-Renyi topology may lead to better performance of a network. The topology of a network will be easier to incentives for then the transaction behaviour of people. For that reason both models perform well, but neither of the models can be validated for distribution accuracy.

7.6. Evaluation of the Success Rate with the Merchant Algorithms

Within this section the evaluation of the Merchant algorithms will be presented.

7.6.1. Methodology and Metrics

As for all the evaluations all these experiments were done with 500 nodes. Each experiment was run 10 times. The results in the table below show that the merchant algorithms are able to provide a way to increase success ratio of transactions in a generated PCN with synthetic data-sets.

The evaluation of the merchant algorithms will determine the impact of a set of transaction distributions and channel distributions from both types of synthetic data-sets. Two different topologies will also, these are the Scale-Free and Erdos-Renyi topologies. The evaluation metric will be based on the success ratio. Success Ratio, is a ratio based on the amount of completed transactions vs the amount of total transactions initialized within the system.

7.6.2. Experiment Set-up

In Table 7.1, the three different scenarios can be seen that the merchants are tested against. These three scenarios were chosen to portray three different use cases. The Micro-Payments test is meant to emulate payments such as buying apps, e.g., this test will be effected more by the fees of the network than the other two tests. This is caused by the low average value of the transitions compared to the fees. The Online-Shopping scenario is meant to emulate a scenario of purchases online, more geared at retail. While the fees play less of a role the ratio between TRANS_P and CAP_P is different than the other two tests, allowing for a different success ratio compared to the other two tests. The Large-Payments is scenario that tries to emulate 'larger' payments. Also with a unique ratio.

Name	Average Transaction Value (T)	Average Capacity value (C)	Ratio of T/C
Micro-Payments	5	400	0.0125
Online-Shopping-Payments	200	2000	0.1
Large-Payments	2000	10000	0.2

7.6.3. Results

The results are broken up into three sections. Micro-Payments, Online-Shopping-Payments and Large-Payments.

Micro-Payments

Within this section the results of the Micro-Payments can be seen in Table 7.7. Overall the Success ratio is the highest of all three experiments as expected. The Merchant algorithms outperform in the Success Ratio compared to Partial-Blocking.

Simulation Properties	Graph-Type	Data-Set-Type	TRANS_P	CAP_P	Total Trans	Success Rate
Partial-Blocking	SC	LN	5	400	2000	72% ±9
Passive Merchant	SC	LN	5	400	2000	76% ±8
Active Merchant	SC	LN	5	400	2000	77% ±9
Partial-Blocking	SC	XRP	5	400	2000	76% ±6
Passive Merchant	SC	XRP	5	400	2000	80% ±7
Active Merchant	SC	XRP	5	400	2000	80% ±9
Partial-Blocking	ER	LN	5	400	2000	79% ±7
Passive Merchant	ER	LN	5	400	2000	84% ±5
Active Merchant	ER	LN	5	400	2000	85% ±6
Partial-Blocking	ER	XRP	5	400	2000	93% ±4
Passive Merchant	ER	XRP	5	400	2000	93% ±4
Active Merchant	ER	XRP	5	400	2000	93% ±3

Table 7.7: Results of the Micro-Payments Simulation Set

Online-Shopping-Payments

Within this section the results of the Online-Shopping-Payments can be seen in Table 7.8. The Merchant algorithms outperform in the Success Ratio compared to Partial-Blocking.

Simulation Properties	Graph-Type	Data-Set-Type	TRANS_P	CAP_P	Total Trans	Success Rate
Partial-Blocking	SC	LN	200	2000	2000	47% ±10
Passive Merchant	SC	LN	200	2000	2000	51% ±12
Active Merchant	SC	LN	200	2000	2000	52% ±11
Partial-Blocking	SC	XRP	200	2000	2000	51% ±7
Passive Merchant	SC	XRP	200	2000	2000	54% ±10
Active Merchant	SC	XRP	200	2000	2000	55% ±10
Partial-Blocking	ER	LN	200	2000	2000	57% ±9
Passive Merchant	ER	LN	200	2000	2000	63% ±7
Active Merchant	ER	LN	200	2000	2000	63% ±12
Partial-Blocking	ER	XRP	200	2000	2000	59% ±9
Passive Merchant	ER	XRP	200	2000	2000	62% ±9
Active Merchant	ER	XRP	200	2000	2000	65% ±10

Table 7.8: Results of the Online-Shopping-Payments Simulation Set

Large-Payments
Within this section the results of the Large-Payments can be seen in Table 7.10. The Merchant algorithms outperform in the Success Ratio compared to Partial-Blocking.

Simulation Properties	Graph-Type	Data-Set-Type	TRANS_P	CAP_P	Total Trans	Success Rate
Partial-Blocking	SC	LN	2000	10000	2000	23% ±8
Passive Merchant	SC	LN	2000	10000	2000	27% ±9
Active Merchant	SC	LN	2000	10000	2000	29% ±8
Partial-Blocking	SC	XRP	2000	10000	2000	27% ±7
Passive Merchant	SC	XRP	2000	10000	2000	30% ±6
Active Merchant	SC	XRP	2000	10000	2000	31% ±9
Partial-Blocking	ER	LN	2000	10000	2000	34% ±8
Passive Merchant	ER	LN	2000	10000	2000	39% ±8
Active Merchant	ER	LN	2000	10000	2000	41% ±6
Partial-Blocking	ER	XRP	2000	10000	2000	39% ±8
Passive Merchant	ER	XRP	2000	10000	2000	45% ±9
Active Merchant	ER	XRP	2000	10000	2000	47% ±7

Table 7.9: Results of the Large-Payments Simulation Set

7.6.4. Discussion and Conclusion

As mentioned, limited research exists in this area. The Merchant algorithms are state of the art, trying to keep channels balanced without adding an extra protocol layer, i.e. the work done on Acyclic Payment Networks[70], allowing for real-time channel balancing. In the section 3.2.1, Revive is presented, a rebalancing strategy for PCNs. Comparison against this rebalancing strategy has been excluded due to the mechanism behind the strategy. Re-balancing occurs on a separate mechanism, without directing transactions and a high-level of trust is needed within the system. While not impossible such a strategy would require stricter security and privacy guarantees. Due to the discrepancy in strategies, and having no clear metric in their results to compare to, the evaluation of the Merchants will exclude revive.

Looking at the results the Active Merchant may be able to have a constant higher success ratio than the Passive Merchant due to it broadcasting out discounts. Taking out some of the randomization done during the initial set-up of starting a transaction. As spanning-trees are selected at random unless the node has received a coupon for a certain edge combined with a spanning-tree. Thus the Active Merchant can take out some of the randomization as the node has knowledge before hand which path may be cheapest.

The Erdos-Renyi topologies out-perform the Scale-Free topologies. Attributing this to difference in path counts and possible paths between two nodes allows more flexibility. If certain links are dead/directional then a route can be found around it more easily in an Erdos-Renyi graph. The success may also come from the fact the diameter of the networks tested are almost identical. Allowing nodes to be highly inter-connected for the Erdos-Renyi graph.

7.7. Evaluation of the Overhead with the Merchant Algorithms

Within this section the evaluation of the overhead of the Merchant algorithms is presented.

7.7.1. Methodology and Metrics

To evaluate the overhead of the Merchant Algorithms on SpeedyMurmurs the total message count will be recorded during each trial simulation. The total amount of coupon messages sent as an Active Merchant will also be recorded. The Passive Merchant does not create extra messages, however the total message count may still increase due to the extra dynamics in the network brought on by fluctuating fees.

7.7.2. Results

Within this section the messages being sent over the network can be seen. The Active Merchant has a large over-head compared to that of SpeedyMurmurs without the Active Merchant.

Rebalancing	Average Total Message Count	Standard Deviation	Average Active-Merchant Messages
Partial-Blocking	47.2e10	12.2e10	N/A
Passive Merchant	45.7e10	13.9e10	N/A
Active Merchant	62.8e10	16.8e10	17.0e10

Table 7.10: Message Overhead

7.7.3. Discussion and Conclusion

The overhead that is created by the Merchant algorithms is about 30%. A way to decrease the message overhead would be by backpacking coupons onto other messages a node forwards. This would decrease the amount of nodes getting the knowledge, however if the node is very active then it would have a higher chance of letting nodes know of state of the channel. This would add complexity to canceling coupons, but that may be fixed with a broadcast down stream. Overall the Active Merchant algorithm is not a heavy message overload, it does inquire a bit of overhead in the SpeedyMurmurs implementation. This can be attributed to the light-weight nature of SpeedyMurmurs.

8

Conclusion and Future Work

8.1. Conclusion

Within this thesis, the current state of Payment Channel Networks(PCN) has been addressed. From the exploration of the current state of research in this field two different contributions arose. The first being a pair of rebalancing algorithms, the Merchants, that have been incorporated in SpeedyMurmurs a state-of-the-art path-based-transaction algorithm secondly two synthetic data-set models have been proposed based on current knowledge of the Lightning and Ripple networks.

The synthetic data-sets are an essential tool that is currently missing in the research field of PCNs. One issue that researchers face is that collecting data-sets is not trivial nor are data-sets made readily available. Current data-sets, are without concurrent transactions and are limited to an early phase of Ripples network. As the technology matures and evolves this data-set will quickly become obsolete and out-dated. While no direct validation can be done on the synthetic data-set models on how well they accurately emulate transaction behavior in PCNs, the data-sets allow for concurrent transactions to take place and generate transactions and capacities that can be regulated by the researcher. Giving researchers the ability to control the type of scenarios being simulated. As research progresses in this field it is not unthinkable that data-sets will be made available, these synthetic data-sets can still serve a purpose.

The novel idea of the Merchant algorithms is that they do not need an additional layer of protocol to be implemented into a PCN, as the algorithms work by adjusting fees to incentives nodes to make use of a certain paths. Other researchers have suggested using a protocol that is not embedded in the transaction process as presented in Section 3.2.1 and 3.2.2. As both approaches of dealing with rebalancing are state-of-the-art, and not much is known about how PCN operate currently no clear advantage in the operation of these approaches can be described.

An advantage to rebalancing a PCN with a protocol that works on another layer of a PCN is that it may be able to rebalance links more effectively, as it can optimize re-balances with more precision due to it not being limited by the transaction amount. While REVIVE is limited to only cyclic graphs, Subramanian *et al.* showed that such an approach can be used in directional and acyclic bi-drectional graphs[70]. Such a protocol has some apparent disadvantages in a dynamic network, as rebalancing links will be done as transactions are processed and the values of the links may be different before the protocol is able to complete a rebalance. Rebalancing with financial incentives in this case allows for the rebalancing to happen real-time, though may be limited in the extend it can optimize a links rebalance. The Merchant algorithms are adaptive to change in the network and can be implemented on any topology as cycles are mitigated. Some ideas about the optimization of the the Merchant algorithms is discussed in the next section.

8.2. Future Work

Further research into the synthetic data-set and the Merchant algorithms could help optimize both contributions. The Merchant algorithms have been shown initial success by improving the success rate of a PCNs in this thesis's evaluation. Further exploration into these algorithms may lead to better results, one idea to optimize the Merchant algorithms is to create a bartering system for the amount of that is getting sent over the path. The bartering system would allow nodes to request a certain amount of the transaction for a certain fee. An implementation might entail a single round as the initial path search is done, or multi-rounds of the

bartering could take place. A single round can be seen as the sender sending information along the path and the forwarder sending a single reply. Such an optimization could however lead to security issues, thus the resulting security effects of such an implementation would need to be studied. Some of the security issues may stem from the fact malicious nodes will be able to win transactions to take there link, and could result in transactions to fail. Another issue this may bring is in privacy, if a node can see the value of the whole transaction the transaction value privacy is lost, but methods can be thought of to ensure that nodes in a path are unable to known the transaction value with certainty.

As for the synthetic data-set models research can be continued in this area as well. Without validation against actual data-sets, it is difficult to establish how accurate these models can emulate the behaviour of a PCN. Being able to compare the models to data-set also allows for a further understanding of the mechanisms that make up a PCN. An idea that has been discussed with other researchers in this field is to create transactions that go in a certain direction of the graph. A reason for why this may occur can be seen in online-retail. Shoppers tend to buy from one or multiple-stores, this creates a direction from the shoppers to the stores in payments.

This would mean that sub-graphs would be needed to be established and that the transaction volume moves heavier from one sub-graph to another. Such an implementation would allow for stress testing of rebalancing algorithms. As it may be impossible to balance a PCN without going to the blockchain if transactions have a higher transaction volume in one direction of the graph. Whilst it is unknown if transactions have direction over a graph, it may lead to better insights into the limits of rebalancing algorithms.

The network in this thesis has taken a static approach, the network is established and kept the same throughout the duration of the simulation. In actual PCN nodes are able to join and leave the network, this is a more dynamic environment than that was simulated. While studies have been done into the Lightning Network, little is known about the dynamics of a PCN. Due to the little information, these dynamic elements were not added to the simulation. Research into this field will allow for a better understanding of the impact a PCN has on a blockchain, if rebalancing links is needed via the blockchain periodically, and how to simulate this dynamic environment better.

As closing words, this thesis has presented two contributions to the research of PCN. By formalizing synthetic data-sets and examining rebalancing strategies for path-based transactions interlaced within the transaction protocol. The work has been evaluated and shows promise; as does this field of research.

Bibliography

[1] Real-time lightning network statistics. URL https://1ml.com/statistics.

[2] Payment channels, . URL https://en.bitcoin.it/wiki/Payment_channels.

[3] Contract, . URL https://en.bitcoin.it/wiki/Contract#Example_7:_Rapidly-adjusted_.28micro.29payments_to_a_pre-determined_party.

[4] Cryptocurrency market capitalizations. URL https://coinmarketcap.com/.

[5] Lightning network explorer. URL https://explorer.acinq.co/.

[6] Currencyrate, oct 2019. URL https://btc.currencyrate.today/eur/819.

[7] Lightning stats, oct 2019. URL https://1ml.com/statistics.

[8] Ripple stats, oct 2019. URL https://xrpcharts.ripple.com/#/topology.

[9] Latency statistics, Jan 2019. URL https://enterprise.verizon.com/terms/latency/.

[10] Xrp charts, oct 2019. URL https://xrpcharts.ripple.com/#/xrp-markets.

[11] Ittai Abraham, Dahlia Malkhi, et al. The blockchain consensus layer and bft. *Bulletin of EATCS*, 3(123), 2017.

[12] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.

[13] Hagit Attiya and Jennifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. John Wiley & Sons, 2004.

[14] Albert-László Barabási, Erzsébet Ravasz, and Tamas Vicsek. Deterministic scale-free networks. *Physica A: Statistical Mechanics and its Applications*, 299(3-4):559–564, 2001.

[15] Charlie Barton. Why aren't we all buying cryptocurrency?, oct 2019. URL https://www.finder.com/uk/why-people-arent-buying-cryptocurrency.

[16] Roman Beck. Beyond bitcoin: The rise of blockchain world. *Computer*, 51(2):54–58, 2018.

[17] BitcoinWorldWode. How many bitcoins are there?, oct 2019. URL https://www.buybitcoinworldwide.com/how-many-bitcoins-are-there/.

[18] Béla Bollobás, Christian Borgs, Jennifer Chayes, and Oliver Riordan. Directed scale-free graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 132–139. Society for Industrial and Applied Mathematics, 2003.

[19] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[20] K Mani Chandy, Jayadev Misra, and Laura M Haas. Distributed deadlock detection. *ACM Transactions on Computer Systems (TOCS)*, 1(2):144–156, 1983.

[21] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography*, pages 319–327. Springer, 1988.

[22] Usman W Chohan. The limits to blockchain? scaling vs. decentralization. 2019.

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

[24] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[25] My Accounting Course. What is a ledger, oct 2019. URL `https://www.myaccountingcourse.com/accounting-dictionary/ledger`.

[26] CryptoCompare. Coins list, oct 2019. URL `https://www.cryptocompare.com/coins/list/USD/20`.

[27] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[28] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1): 17–60, 1960.

[29] Ittay Eyal and Emin Gün Sirer. How to disincentivize large bitcoin mining pools. *Blog post: http://hackingdistributed. com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools*, 2014.

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

[31] Finder.com. Why haven't we all bought cryptocurrency yet?, oct 2019. URL `https://www.finder.com/why-people-arent-buying-cryptocurrency`.

[32] William Foxley. Ripple extends banking network with finastra partnership, Oct 2019. URL `https://www.coindesk.com/ripple-extends-banking-network-through-finastra-partnership`.

[33] Robert G Gallager, Pierre A Humblet, and Philip M Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and systems (TOPLAS)*, 5(1):66–77, 1983.

[34] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16. ACM, 2016.

[35] Omkar Godbole. Bitcoin's total share of crypto market now highest since march 2017, oct 2019. URL `https://www.coindesk.com/bitcoin-price-bounces-to-10-5k-as-dominance-rate-passes-70`.

[36] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. pages 258–272, 1999.

[37] Rajashekar Kailar. Accountability in electronic commerce protocols. *IEEE Transactions on software engineering*, 22(5):313–328, 1996.

[38] Ghassan O Karame and Elli Androulaki. *Bitcoin and blockchain security*. Artech House, 2016.

[39] Rami Khalil and Arthur Gervais. Revive: Rebalancing off-blockchain payment networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 439–453. ACM, 2017.

[40] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 279–296, 2016.

[41] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, 2014.

[42] Joshua A Kroll, Ian C Davey, and Edward W Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, page 11, 2013.

[43] Victor Lewis, Kenneth D Kay, Chandrika Kelso, and James Larson. Was the 2008 financial crisis caused by a lack of corporate ethics? *Global Journal of Business Research*, 4(2):77–84, 2010.

[44] Lightning, dec 2019.

[45] 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.

[46] M Molloy and B Reed. Random struct. alg. 6, 161 (1995). *Combin. Probab. Comput*, 7:295, 1998.

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

[48] Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Silentwhispers: Enforcing security and privacy in decentralized credit networks. In *24th Network and Distributed System Security Symposium (NDSS 2018)*, 2017.

[49] Walid Najjar and J-L Gaudiot. Network resilience: A measure of network fault tolerance. *IEEE Transactions on Computers*, 39(2):174–181, 1990.

[50] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008. URL `https://bitcoin.org/bitcoin.pdf`. Accessed: 2015-07-01.

[51] Satoshi Nakamoto. Re: Bitcoin p2p e-cash paper. *Email posted to listserv*, 9:04, 2008.

[52] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 33–43, New York, NY, USA, 1989. ACM. ISBN 0-89791-307-8. doi: 10.1145/73007.73011. URL `http://doi.acm.org/10.1145/73007.73011`.

[53] Lightning Network, nov 2019.

[54] Raiden Network, nov 2019.

[55] Christos H Papadimitriou. Serializability of concurrent database updates. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1979.

[56] Christos H Papadimitriou and David Ratajczak. On a conjecture related to geometric routing (algorithmic aspects of wireless sensor networks), volume 3121/2004, chapter on a conjecture related to geometric routing, 2004.

[57] Christos H Papadimitriou and David Ratajczak. On a conjecture related to geometric routing. *Theoretical Computer Science*, 344(1):3–14, 2005.

[58] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[59] Defi Plus. Lightning network, oct 2019. URL `https://defipulse.com/lightning-network`.

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

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

[62] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[63] Helena Rifa-Pous and Jordi Herrera-Joancomartí. Computational and energy costs of cryptographic algorithms on handheld devices. *Future internet*, 3(1):31–48, 2011.

[64] Stefanie Roos, Martin Beck, and Thorsten Strufe. Anonymous addresses for efficient and resilient routing in f2f overlays. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[65] 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.

[66] David Schwartz, Noah Youngs, Arthur Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5:8, 2014.

[67] István András Seres, László Gulyás, Dániel A Nagy, and Péter Burcsi. Topological analysis of bitcoin's lightning network. *arXiv preprint arXiv:1901.04972*, 2019.

[68] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia Fanti, and Pramod Viswanath. Routing cryptocurrency with the spider network. *arXiv preprint arXiv:1809.05088*, 2018.

[69] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security*, pages 457–468. Springer, 2014.

[70] Lalitha Muthu Subramanian, Guruprasad Eswaraiah, and Roopa Vishwanathan. Rebalancing in acyclic payment networks.

[71] Olivier Denecker Madhav Goparaju Marc Niederkorn Sukriti Bansal, Philip Bruno. Global payments 2018: A dynamic industry continues to break new ground. Online, Mckinsey&Company, oct 2018.

[72] Manny Trillo. Stress test prepares visanet for the most wonderful time of the year, 2013. URL `https://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html`.

[73] Paul F Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 35–42. ACM, 1988.

[74] András Varga. The omnet++ discrete event simulation system. In *In ESM'01*, 2001.

[75] Visa. Visa inc. at a glance, June 2015. URL `https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf`.

[76] Bimal Viswanath, Mainack Mondal, Krishna P Gummadi, Alan Mislove, and Ansley Post. Canal: Scaling social network-based sybil tolerance schemes. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 309–322. ACM, 2012.

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

[78] Shira Werman and Aviv Zohar. Avoiding deadlocks in payment channel networks. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 175–187. Springer, 2018.

[79] B. Wiki. Hash time locked contracts. URL `https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts`.

[80] Wikipedia. Ledger — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Ledger`, 2019. [Online; accessed 12-October-2019].

[81] Wikipedia contributors. Smart contract — Wikipedia, the free encyclopedia, 2019. URL `https://en.wikipedia.org/w/index.php?title=Smart_contract&oldid=923105708`. [Online; accessed 15-November-2019].

[82] Robert S Winternitz. A secure one-way hash function built from des. In *1984 IEEE Symposium on Security and Privacy*, pages 88–88. IEEE, 1984.

[83] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[84] Rui Zhang, Rui Xue, and Ling Liu. Security and privacy on blockchain. *arXiv preprint arXiv:1903.07602*, 2019.

A

Ripple Dataset Analysis

Development and research in the are of PCNs is still in early development, a consequence of this is that there are no load models that have been verified. Not enough research has been done in trying to emulate how PCNs behave within the real-world with models. This is partially due to the lack of data-sets and partially because of the current state of this research area. Roos benchmarked speedymurmurs against a ripple dataset in her work on speedymurmurs[65]. The ripple dataset was taken over the course of three years, in the time window of 2013 – 2016. Such a vast dataset can aid in the creation of load models that can emulate real-world interaction. To get the best picture of how PCNs work multiple data-sets would give a better results, though due to the lack of access to different data-sets only the ripple dataset will be used to create a representative model of a PCN.

Taking the dataset directly from Roos's work, means that the dataset has already been filtered and all data has been removed that created inconsistencies. The dataset was split into a dynamic and static graph topologies and transactions. For the case in this work, only the static dataset will be analyzed as during the running of the simulation, no nodes will fail, be added or have any dynamic properties.

To generate a representative model a couple of characteristics need to be examined from the dataset. These characteristics are **Channel Capacity**, **Value of transactions** and **Node Connection Distribution**. These characteristics will form the foundation of different load models. Different load models will be created that generate transactions either randomly, or give the transactions a specific direction within the model. These load models are as follows:

A.0.1. Node Connection Distribution

Node connection distribution is the measurement of how many nodes are connected to one another and how these connections are distributed throughout the network. From the filtered ripple data-set it can be seen in table A.1 that the average node has a very low connectivity. For the static data-set it is just under 3 connections and for the dynamic dataset it is a little more than 3.5.

Table A.1: Nodes, Edges and Average Edge per Node from Ripple Dataset

Dataset	Amount of Nodes	Amount of Edges	Average
Static	67149	199547	2.97
Dynamic	93502	331096	3.54

Such low averages indicate that most nodes are not highly connected. Examining the static data it can be seen that 47842 nodes have a single connection, that is 71%. For the dynamic dataset there seems to be less single connection connectivity, 58037 nodes have a single connection. Making the amount of single connected nodes 62%. Having more such large fraction of the network connected with a singular other node demonstrates that the distribution of connections for non-singular connected nodes in the network must be much higher.

For nodes connected to less then 12 other nodes the distribution can be seen in fig A.1. Within the histogram range of maximum 12 connected nodes more than 99% of the nodes fall into this category. This

indicates that the connectivity of the ripple PCN during 2013 – 2016 was rather low. A graph having such low connectivity means that to create a graph curtain nodes must have a very high connectivity.

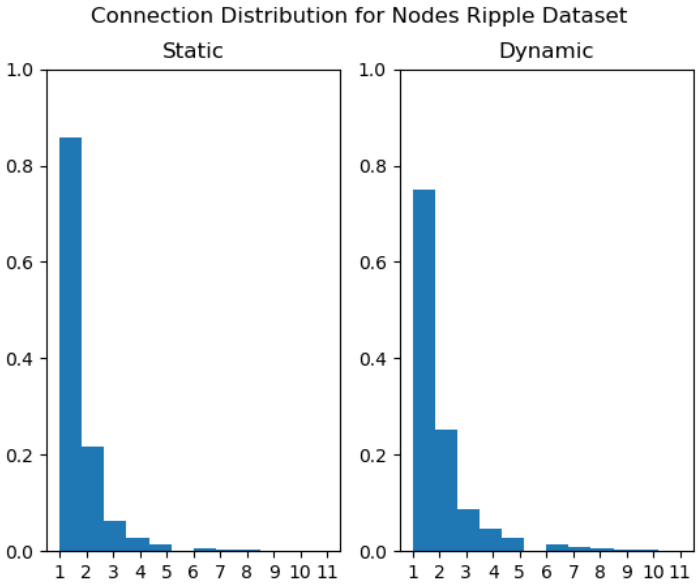

Figure A.1: Node connectivity distribution pulled from the static and dynamic dataset used by Roos

Table A.2: 100 Most Connected Nodes

Dataset	Amount of Edges	Percentage of Edges Total	Max	Min
Static	96639	48.43%	12506	28
Dynamic	153542	46.37%	21363	74

Modeling the node connectivity distribution one will need to consider the low amount of connectivity found in the ripple dataset. With such low connectivity between the nodes, the network becomes vulnerable to attacks and failures if the top tier connected nodes were to fail. This paper does not look into the effects of this on the stability of the node network, however modeling only a low connectivity graph may not fully be indicative of how PCNs should and will be created in the future. Creating graphs with super high connectivity would also not be indicative of how PCNs may work, seeing as currently the data shows that when users connect to a PCN, they will connect to the network via a highly connected node.

A.0.2. Channel Capacity

The amount and type of payments that can be made on a network are dependent on the channel capacity. If a network has a low channel capacity on most links then high capacity payments cannot be made. Analyzing the channel capacity graph from the ripple dataset it becomes apparent that not all edges in the network have a capacity. From table A.1 it was shown that the static data set had a total of 199547 edges, however the capacity dataset has only managed to establish 98625 edges with a capacity that is above 0. For the dynamic dataset for only 8627 nodes a capacity was found, though 165548 data points were in the file and in the network graph 331096 edges exist. This could be that more capacity is added dynamically over time.

Table A.3: Link Capacity found in Ripple dataset

Dataset	Average Capacity of Link	Max Capacity	Min Capacity	Edges with Capacity
Static	$1.1e36$	$1.1e41$	$8.39e-18$	98625
Dynamic	$8.39e18$	$7.23e22$	$1e-18$	8627

Taking this data at face value and making assumptions from this data to translate to running simulations does not seem fully plausible. The reliability of this part of the data seems questionable. The max capacity

of the static dataset is $1.1e41$, according to coinmarketcap the max supply of ripple is $1e11$. Meaning that the max capacity of this data set far exceeds the maximum amount of ripple available on the market. Assuming these data points are errors it is easy enough to filter out these values. The data is filtered twice separately one at the max supply of $1e11$ and once again at $1e9$, this last value was taken due to it representing roughly $\$10,000,000$. Seeing as such technology is relatively new, most nodes would not set-up such high amount of capacity between one another.

Table A.4: Link Capacity found in Ripple dataset filtered 1e11

Dataset	Average Capacity of Link	Max Capacity	Edges with Capacity	Total Sum Ripple
Static	$640e6$	$9.6e10$	83633	$53e12$
Dynamic	$19.8e6$	$7.2e10$	8534	$169e9$

Table A.5: Link Capacity found in Ripple dataset filtered 1e9

Dataset	Average Capacity of Link	Max Capacity	Edges with Capacity	Total Sum Ripple
Static	$65e6$	$960e6$	73578	$4.8e12$
Dynamic	$495e3$	$723e6$	8522	$4.2e9$

After extreme values have been filtered out of the graph the values in table A.5 still shows that the sum of capacity on the links exceeds the maximum amount of ripple available. The dynamic graph stays below the maximum capacity however it has significantly less edges accounted for. Filtering out just the values that go above the maximum amount of available ripple as in table A.4, these values also are well above what is expected. Due to these values seemingly being so off, no distribution nor conclusions for real world capacity links can be gathered.

A.0.3. Value of Transactions

An important aspect of PCN is that users can exchange currency for services or goods. Examining the ripple dataset for transaction amounts should give a good indication of the amount of currency that is being exchanged and in what volumes this usually occurs. From that data one could construct a value transaction distribution, allowing simulations to run synthetic transactions that reflect transaction amounts from users of a PCN.

Table A.6: Transactions from Ripple Dataset

Dataset	Average Transaction	Median Transaction	Max Transaction	Number of Transactions
Static	$2.2e11$	3.0	$7.232e17$	$1e6$
Dynamic	$1.8e17$	0.0	$7.23e22$	800819

The table A.6 represents the raw data from the data sets. Averages for both static and the dynamic data set are above the limit of ripples available, $1e11$. Secondly the median of these datasets is very low. For the dynamic dataset the median is 0, thus no currency is being transacted and for the static dataset it is 3. This suggests that the dataset has transactions that are trying to transact more than possible and transactions that do not transact anything. Assuming the users of the ripple PCN, are unable to do either of these things. One could filter out values above plausible limits and remove transactions that do not transfer anything.

Table A.7: Transactions from Filtered Ripple Dataset

Dataset	Average Transaction	Median Transaction	Max Transaction	Number of Transactions
Static	13562	3	$7.7e8$	999199
Dynamic	26834	0	$.9e8$	800229

- Talk about many small transactions
- exponential distribution

- transactions with super small payments

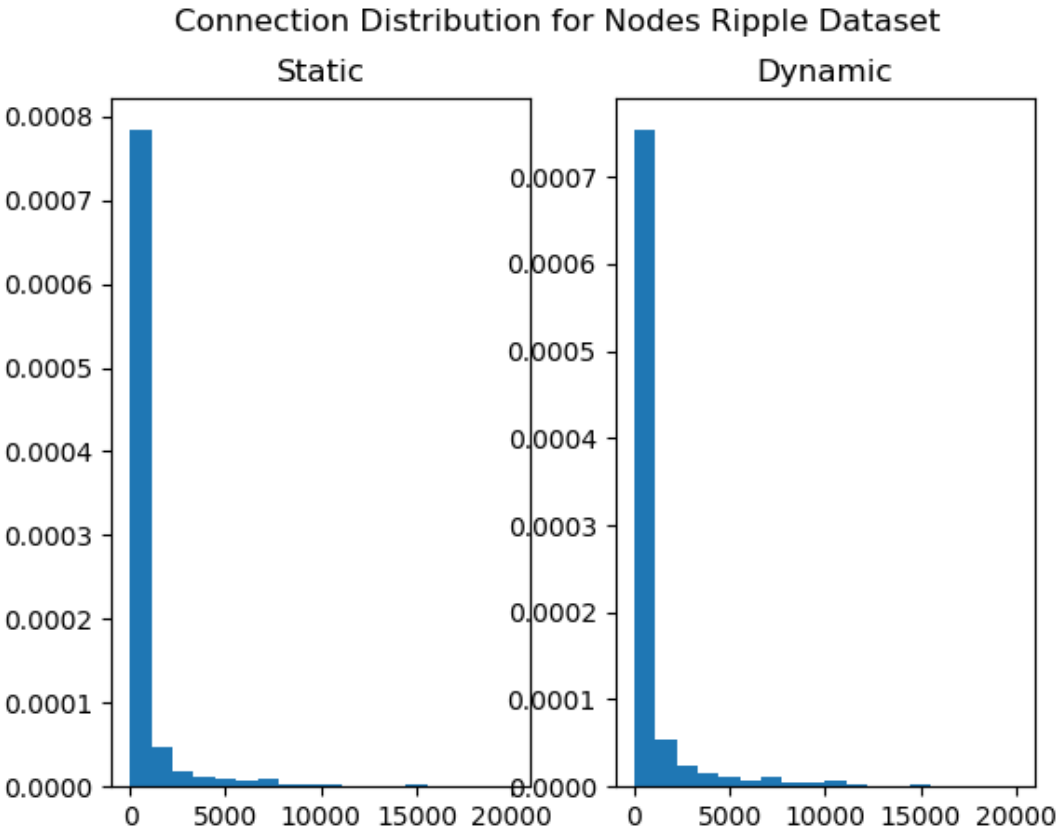

Figure A.2: Transaction distribution