Blockchain Layer2 Based Mass E-Commerce

by

Sijia Zhao

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfilment

of the requirements

for the Degree of

Doctor of Philosophy

University of Dublin, Trinity College

March 2023

Declaration

Sijia Zhao

January 10, 2023

Acknowledgments

First and foremost, I would like to thank my supervisor Professor Donal O'Mahony for his guidance, expertise, and support throughout my PhD. During the last two years, most of my PhD studies were spent in lockdown, much of our interactions were online, but he still tried his best to help me for my research and gave me many insightful suggestions. I am grateful to him for the many helpful and invaluable comments on previous versions of this thesis. Professor Donal has often made me feel privileged to be his PhD student.

I would like to thank the School of Computer Science and Statistics for being my second home for the past few years. I would also like to thank Trinity College and my country for giving me the opportunity to do this work.

Finally, thanks must go to my parents, Zhilin and Mengqiu, for their understanding, support, encouragement, and love during my study and research over many years.

Sijia Zhao

University of Dublin, Trinity College
March 2023

Abstract

Blockchain technology has become a new driving force for technological innovation and has the potential to revolutionize the e-commerce industry. Hundreds of millions of users, merchants and platforms can be connected to form a blockchain-based network, enabling mass e-commerce. In mass e-commerce, transactions are settled on a distributed shared ledger. Payment issues, security issues and transaction transparency issues in traditional e-commerce will be improved. We find that blockchain Layer2 technology which moves transactions off the blockchain, has great potential to become a fast and cheap solution for mass e-commerce.

We analyzed the existing Lightning Network. We found that the evolved topology leads to long path lengths and congested nodes. The routing algorithms are slow and will cause high latency for payments as well as a lot of routing-related traffic. We show that the lightning network, as it exists today, will be unable to cope with the demands of the mass e-commerce environment.

This dissertation presents a blockchain Layer2 based architecture to address the payment problems of the current e-commerce industry. A tree-based topology is adapted to provide the e-commerce network among customers, merchants, and platforms. A Kademlia-based routing algorithm is incorporated to find the minimum cost path from customers to merchants.

We conducted experiments on the performance of the topology and routing algorithm. Simulation results show that the network we propose can effectively shorten the path length compared with the existing unstructured Lightning Network. The K-routing algorithm we put forward can find paths faster and reduce transaction delay. The 2-Level platform node topology we design can effectively reduce the total number of payment channels required by the network. The maximum path length does not increase as the size of the network increases allowing the network to scale effectively. Furthermore, transactions can be distributed more evenly across the network of payment nodes which helps to avoid the trends of centralization exhibited by the Lightning Network. Our network also tolerates multiple node failures maintaining high availability at all times. Constructing a payment network of this scale takes some time. We show that scaling up our design to Amazon's scale involves on-chain transactions that can be completed within days or weeks. We show that our network can be realistically deployed at a global scale.

Many of the changes we propose to the Lightning Network provide a solution for future blockchain scenarios with a large number of participants. The system can scale with the demand of the network in the future. Although building the payment network involves overhead, the rewards in terms of scalability make this a worthwhile trade-off.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, people deposit money in a bank and buy products on e-commerce platforms. The payment will travel from the bank account directly or through a third-party payment service, such as PayPal or Alipay, to merchants. A successful transaction needs good behaviors from all participants during this process. Trust is needed for this to work. Anyone can register an account on an e-commerce platform without an ID card. Most platforms, including Amazon, do not verify the identification of users. Buyers can randomly select sellers according to their preferences. This increases risk in e-commerce transactions, like fraud and theft, because we cannot guarantee that every merchant is trustworthy. The platform can compensate buyers within a specific range to keep its reputation. Amazon reported that millions of dollars are involved in fraudulent transactions every year [1]. Lack of trust has become a significant obstacle to the further development of e-commerce. In addition, high transaction or processing fees are charged by a third-party payment system. Table 1.1 [2] shows the fees charged by Amazon to process a transaction. As the transaction amount increases, the transaction fees also increase gradually. Therefore, better solutions are sought for this industry.

Table 1.1: Amazon Transaction fees (Online)

Type	Processing Fee	Authorization Fee
Domestic Transaction	2.9%	$0.30
Cross-border Transaction	3.9%	$0.30

The emergence of blockchain technology may bring new solutions for improving trust in the e-commerce environment. Blockchain is regarded as the "next generation of innovative technology", and its core advantage is using decentralization, data encryption, time stamps and distributed consensus to build a new trust mechanism and efficient coordination mechanism. However, it is not perfect. Some obstacles make blockchain technology a bad choice and unusable for large-scale applications, like e-commerce. Take Bitcoin as an example. For the Bitcoin blockchain, it takes around 10 minutes to confirm a transaction, and if there is congestion, it takes longer. The maximum number of transactions per second is 7, and the average is 4. Even though Ethereum is ten times faster than Bitcoin. It still cannot support transaction volumes like Visa peaking at 65,000 transactions per second, with an average of 2,000 transactions per second [3]. From these comparisons, we can see the current limitations of blockchain. Because of this, scalability is a significant research topic in the blockchain community.

There are two basic categories of scalability solutions, Layer 1 and Layer 2. Layer 1 solutions take measures on the blockchain itself, including block size extension [4], block interval shortening [5], and sharding [6]. A plan to increase the block size referred to as SegWit2x, failed due to a lack of consensus in 2017 [7]. If the block interval is decreased, only a few high-performance nodes can verify all data in a short interval. This means a smaller percentage of nodes can contribute to the verification process in the network. Sharding [8] aims to divide the blockchain data into small parts and process transactions in parallel to increase the total output. However, sharding faces more challenges in communicating between different shards. The complexity of this solution has caused delays for Ethereum 2.0 [9].

Layer2 solutions attempt to move transactions off the main blockchain to reduce the congestion and computation requirements on the main chain. They require an additional communication layer or structures which allow traders to make hundreds of payments or micropayments outside the blockchain. There are three main directions of layer 2 solutions: Lightning Network for the Bitcoin blockchain, Plasma for the Ethereum blockchain and Rollups for the Ethereum blockchain. State channels are the basis of the Lightning Network, which are basically attached channels that perform blockchain operations and commit them to the blockchain Layer 1. State channels are mainly used as payment channels. Two user nodes are at two sides of a payment channel. Participant nodes and payment channels form a payment channel network over blockchain Layer 1. The payment channel network is arranged in a topology that allows payments to be routed from a source node to a destination node. Our research will more focus on this payment channel network based solution. Plasma is based on sidechains which is a tree-like blockchain structure. Transactions generated in Layer2 can take place independently from Layer 1, which can significantly increase the transaction speed and reduce transaction fees. Rollups essentially transfer the calculation process from the main chain to a separate chain called the “Rollup chain”. Unlike Plasma, security can be guaranteed by the blockchain itself. These Layer2 solutions are considered viable solutions to scale the existing blockchain.

1.1 Objectives

The open issues in blockchain scalability motivate our research into whether Blockchain Layer2 Technology can be used to meet e-commerce requirements. In this dissertation, we will propose a design for a payment network that links customers and merchants through a network of platform nodes. Our objectives and research questions can be listed as follows.

What Layer2-based topology may suit a customer-platform-merchant structure of the scale of Amazon.com?

In general, nodes in Layer2 are unstructured peer-to-peer nodes. They can send transactions to others and can also receive transactions. However, for an e-commerce scenario, participants take on roles as the customer, platform node, or merchant. Customers send transactions in most cases, and merchants receive payments. The platform nodes act as intermediaries between customers and merchants. In addition, establishing a connection like a payment channel will generate fee and time costs on the blockchain. Therefore, finding a suitable topology is the first step of our research.

How many platform nodes are required for our proposed topology for different sizes of e-commerce platforms?

We assume e-commerce companies maintain platform nodes. If the total number of platform nodes is exceptionally high, companies will face extra costs to maintain these nodes. The second question is designing a payment network with a reasonable number of platform nodes.

Which type of routing algorithm can have a better performance in our proposed topology? What factors are necessary to find the optimal path? What metrics are used to analyze performance?

The routing algorithm aims to find a good path from the source to destination node. The payment channel network is a topology consisting of nodes and channels. The routing algorithm we propose needs to minimize the overhead caused by routing from customer nodes to merchant nodes, including time and fee cost.

Compare the performance of our proposed system with the existing e-commerce platform, like Amazon or Taobao.

Amazon’s online shopping system can support hundreds of millions of users’ purchase operations. Whether our proposed system can achieve this metric will be an essential criterion to measure whether the system can be applied in practical scenarios.

1.2 Research Methodology

In order to study whether the blockchain layer2 technology is suitable for e-commerce, we must first understand the features and current problems of e-commerce systems. Therefore, we conducted an observational study on the challenges faced by e-commerce. We learned about the payment methods and processes used by each e-commerce platform, and obtained data such as the user scale and transaction flow of each platform. In addition, we read papers, collected layer2 solutions and understood the current status of the layer2 technology. This thesis presents the classification of layer2 technology that we constructed during the question investigation process. At the same time, a comprehensive analysis of each classification is used as the basis for the network design.

To determine the feasibility of our proposed scheme, we constructed a small network topology of layer2 based e-commerce to simulate user transaction behaviour. Statistical analysis was carried out on the experimental data to determine the relationship among various influencing factors.

1.3 Thesis Contribution

In this section, we summarise the main contributions of the thesis.

We propose a blockchain layer2 based e-commerce architecture. This architecture can support a large number of users like that of traditional e-commerce, with high transaction frequency and small payment amount. Compared with traditional e-commerce and its payment methods, it also shortens transaction time, reduces transaction fees, and enhances the trust of all parties.

We propose a network topology for the blockchain layer2 based e-commerce. This reduces the total number of payment channels required, that is, to reduce the number of interactions with the blockchain and allows the transactions to be passed from a customer node to a merchant node within five intermediaries, even with an Amazon-sized network.

We propose a routing algorithm for the blockchain layer2 based e-commerce. This can find paths faster and reduce transaction delay.

1.4 Overview of Thesis

The thesis is laid out as follows.

Chapter 2 surveys the current e-commerce payment landscape and gives an overview of blockchain technology. We have classified layer2 technology and analysed the advantages and disadvantages of various layer2 solutions in detail. This chapter also explains why we chose the State Channel based solution as the research direction.

Chapter 3 presents our analysis of applying layer2 to e-commerce. This chapter begins with an introduction to the scale of Alibaba and Amazon e-commerce network which are the largest in the world of e-commerce. We compare and analyse the benefits and drawbacks of the popular topology and routing algorithms applied to layer2 based e-commerce. This provides the basis for Chapter 4. The concept of channel rebalancing is introduced, and we list four channel rebalance mechanisms.

Chapter 4 proposes a blockchain layer2 based e-commerce architecture, including the design of network topology, routing algorithm, and channel rebalance mechanism.

Chapter 5 reports on the design and conduct of the experiments.

Chapter 6 summarizes and discusses the finding of the research program in the previous chapters and gives our outlook on possible future research directions.

Chapter 2

Background

2.1 Current E-Commerce Payment Situation

An e-commerce platform is an application which acts as a trading bridge between merchants and purchasers. With the rapid development of the Internet and e-commerce technology, online shopping has become an essential part of people's daily life. In recent years, the number of e-commerce online customers in various regions of the world has increased significantly. More than three million companies worldwide are engaged in e-commerce, including Amazon, eBay, Alibaba and Walmart. There are also many festivals worldwide which cause peaks in e-commerce, such as Singles' Day, Black Friday, Cyber Monday, or the Christmas Sales. For example, Alipay handled a peak of 120,000 transactions per second on Singles' Day, 2020 [10]. Table 2.1 shows the global online shopping customers for different regions and worldwide in 2013 and 2018. In addition, the number of e-commerce users worldwide in 2021 reached 2.14 billion [11].

Table 2.1: Global Online Shopping Customers for Regions in 2013 and 2018 (million)

Year	Asia	Europe	America	Africa	World Wide
2013	460.3	268.7	257	93.6	1079.6
2018	782.4	327.6	343.1	170.6	1623.7

Internet electronic payment is an indispensable part of e-commerce transactions. Customers select products from the merchant via the Internet and purchase with an electronic payment. Most e-commerce platforms accept credit or debit cards. We take a MasterCard payment as an example. Several steps must be undertaken to complete a payment, including authorization, clearing and settlement. The processes involve a customer, an e-commerce platform, a customer's bank, the MasterCard company or MasterCard system, a MasterCard settlement bank, and a merchant's bank, resulting in nearly 20 messages and transactions being exchanged among these participants.

E-Commerce has the following three main features, and we take Amazon.com as an example.

(1) High transaction frequency. Amazon had an average of 66 sales/second in 2019 [12].

(2) The average payment amount is small.

(3) There are many products and diverse trading entities. There are 200 million active products on Amazon and 244 million active user accounts.

However, the existing e-commerce system structure has several problems. Payment methods of e-commerce mainly include bank remittance, professional remittance company payment, credit card payment and third-party payment. Regardless of the payment methods, payment must be coordinated by multiple transaction entities and completed through the settlement method of hierarchical agents.These methods have the following two disadvantages: low efficiency and long payment cycle; high cost. Blockchain technology has great potential to impact the e-commerce industry. Blockchain Technology is best known as the underlying technology of the Bitcoin cryptocurrency. The blockchain is a sequence of blocks which acts as an unchangeable public ledger combining technology and incentive mechanisms to ensure data correctness and trustworthiness without a central authority. Satoshi Nakamoto proposed Bitcoin in 2008 and Vitalik Buterin proposed Ethereum in 2013. Blockchain is a powerful technology and can make organizations or transactions more transparent, democratic, decentralized, effective, and secure. However, compared with traditional solutions, like Visa, blockchain's Transactions per Second (TPS) is significantly low. The Visa Credit Card processing system theoretically can process more than 65,000 transactions per second in 2018. The actual average transaction rate is around 1,700 per second. Bitcoin can handle 3-7 transactions per second. Ethereum can support 29 transactions per second, but it usually supports around 8.

Blockchain Layer2 approaches have the potential to help the e-commerce industry, enabling the blockchain to service high traffic volumes. The layer2 solution is to move many transactions off the blockchain. Only the final status is submitted to the Layer1 chain (blockchain or On-Chain) for verification. This can significantly improve the current limitation on the number of transactions per second. At the same time, the first layer can guarantee the security and trust of the second layer. The following are some advantages blockchain Layer2(Off-Chain) approaches can give e-commerce.

(1) Instant payments. Unlike traditional payments where multiple participants and processes are involved, payments through Layer2 are just between a customer and a merchant. If participants agree on their balance status, that status can be confirmed locally.
(2) No third party. Status updates are propagated between peers. They do not require a consensus mechanism which saves time. Compared with the traditional e-commerce network, customers and merchants do not need to interact with banks or intermediaries for every transaction.
(3) Protect privacy. Status updates are off the chain. Peer-to-peer communication guarantees privacy, and only the final status is submitted to the blockchain, which can better protect users' privacy than the traditional e-commerce platforms.
(4) Low transaction fees. Only on-chain transactions consume significant transaction fees, and status updates are free. Hundreds of thousands of transactions can be generated between a merchant and a customer with very low or no transaction fees. Applying Layer2 approaches can significantly reduce transactions fee for e-commerce users.

2.2 Blockchain Technology

The blockchain [13] is a ledger used to record transactions around the world, providing an alternative payment method without requiring trust in any third party. The word "Blockchain" consists of "block" and "chain". A block is made up of many digital transaction records. A transaction in blockchain technology is digitally signed using cryptography and included in the block for verification. The transactions waiting to be processed are placed in an "unconfirmed transaction pool" [14]. Figure 2.1 shows an example of one of these pools. Each transaction must be confirmed for validity by other computers, which we call "miners". Miners pick up transactions from the pool, process valid transactions, and reject invalid ones at execution time. If the transaction is eligible to be executed, it can be added to a block with the customer's signature. Miners can select hundreds or thousands of transactions to be placed in a block as long as this number does not reach the block size limitation of Bitcoin or the gas limitation of Ethereum. Miners get transaction fees as a reward when they produce a successful block. Therefore, offering a higher transaction fee can help a customer have his transaction quickly chosen for adding to the blockchain.

A block consists of a block header and a block body. Figure 2.2 shows the structure of a block in the Bitcoin blockchain. The block body contains transactions. Transactions are messages that detail funds transmissions from one user to another. Each block has a unique hash code that allows others to distinguish it from other blocks. Miners have

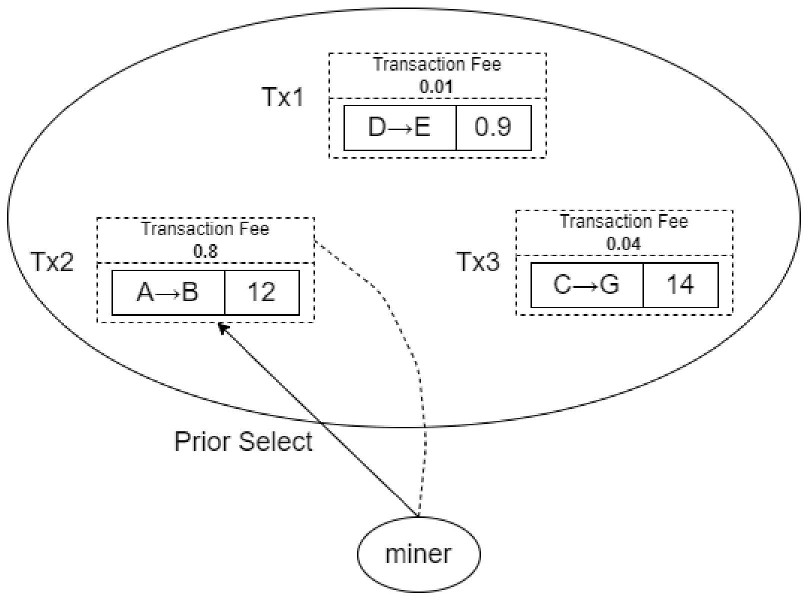


Figure 2.1: Transaction Selection for a Miner(Tx1, Tx2, Tx3 are in the pool. The miner selects Tx2 which has the highest transaction fee.)

to solve a complex mathematical problem to find a hash for a block they want to add to the blockchain. This process is called ”mining”, and the problem-solving algorithm is called ”Proof-of-Work” [15]. Proof-of-work is a way for transactions to be verified on the blockchain. Miners compete to be the first to solve a math puzzle. The winner can get rewards. The metadata of a block header is an input of the hash function. When a hash function is applied, even a one-bit input change causes an entirely different output. This feature makes the puzzle hard to solve but easy to check. Figure 2.2 shows the block header structure. Miners repeatably change the Nonce value in the block header to impact the hash result to match the target output. The algorithm can automatically readjust the difficulty value in the block header to ensure a stable block generating speed. The block can be added to the blockchain once a miner finds a valid hash. The public can then view transactions in this block.

The most popular public blockchain is the Bitcoin blockchain. The Bitcoin blockchain was proposed by Santoshi Nakamoto in 2008 [16]. In Bitcoin, Unspent Transaction Outputs(UTXOs) are used to track funds. UTXOs are coins available to be spent. UTXO's accounting method divides each transaction into inputs and outputs. Each transaction consumes the UTXO generated by previous transactions and generates a new UTXO. The UTXO model is stateless. The balance of an account is all the unspent UTXO collections belonging to the address. Figure 2.3 shows how UTXOs work.

The Bitcoin blockchain uses scripts to execute transactions. Locking scripts are placed on outputs which specify the conditions that must be true to spend the UTXO. Unlocking

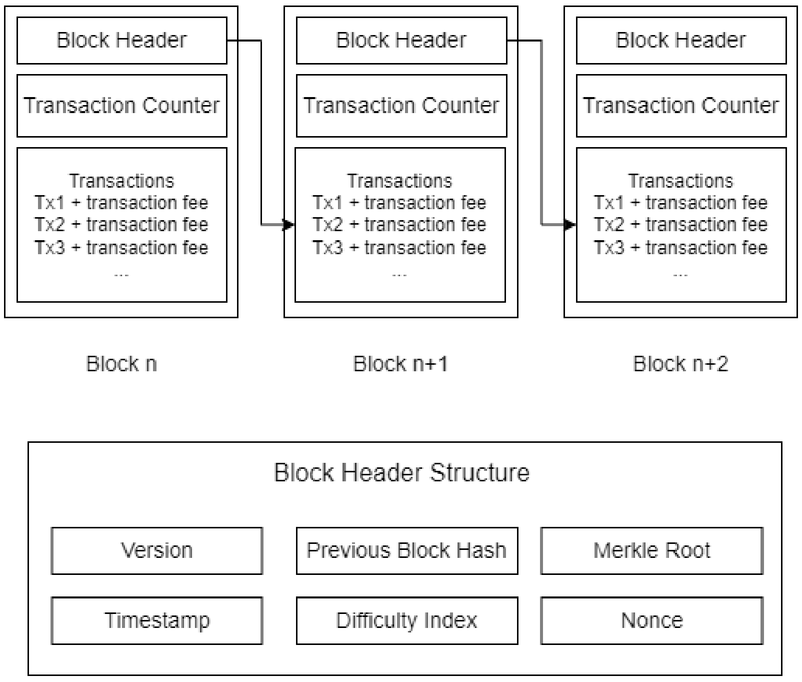


Figure 2.2: The block structure of Bitcoin blockchain

scripts are placed on inputs which prove inputs fulfil the conditions specified in the UTXO. These unlocking scripts contain the user's digital signatures. When a node receives a transaction, it will execute the locking and unlocking scripts to see whether the digital signature matches the address to which the output is locked.

In the Bitcoin blockchain, the sequence order can be easily verified. Inputs link to current outputs. A node can check whether an output has been consumed or not. In addition, transactions are both the result of the current transaction and the proof the previous transactions. There is no extra computation and state storage required.

Blockchain technology has great potential to impact the e-commerce industry. Some researchers and companies have already made contributions to applying blockchain to the e-commerce industry. C. Liu, et al. proposed a transaction settlement system based on blockchain technology to manage transactions automatically using smart contracts [17]. W. Xie, et al. introduced a consensus algorithm to ensure trusted trading in the e-commerce based on blockchain technology [18]. This allowed simple purchases and refunds to be made between participants. Y. Li, et al. proposed a balanced e-commerce model that allows direct payment based on a blockchain between customers and suppliers without involving an e-commerce service provider like Amazon [19]. This method creates many transactions on the blockchain, which may cause congestion if the active user group is large enough. These approaches use the blockchain to improve trust and contracts that execute automatically. However, they use existing unmodified blockchain networks, like

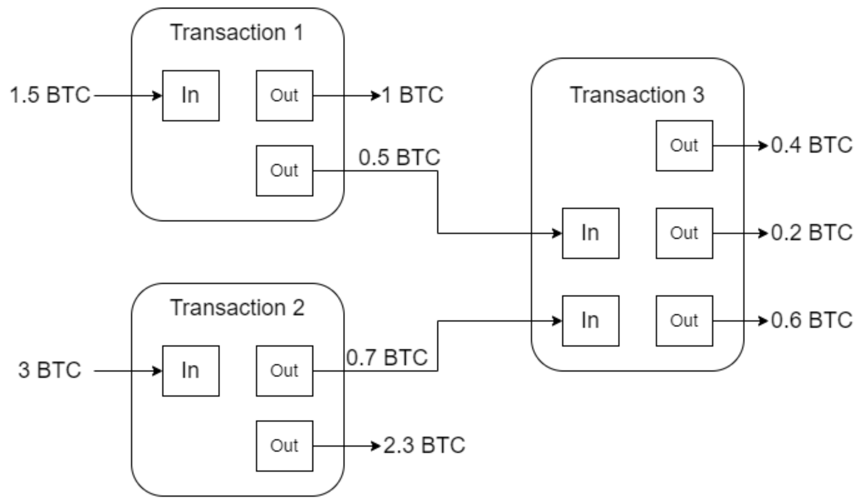


Figure 2.3: UTXO

Bitcoin or Ethereum, and do not deal with scalability issues.

2.3 Blockchain Layer2 Technology

Blockchain Layer2 solutions give another option for the E-Commerce industry. These solutions attempt to decrease the number of transactions on the blockchain network. Unlike Layer1 computation and storage, a Layer2 scaling method is an approach to increase the network's speed without modifying the blockchain itself. In our research, we will focus on the second-layer solutions.

2.3.1 State Channel Based Approaches

State channel based approaches establish an exchange layer above the blockchain, enabling customers and merchants/platforms to update balance states without interacting with the blockchain.

State Channels

The state channel is an agreement between a fixed number of participants, usually two, to implement secure off-chain transactions. State channels not only support payments but also support general state updates. Payment channels are a specific type of state channel exclusively used for payments.

The state channel agreement is as follows. Participants lock a deposit on the chain using a 2 of 2 multi-sig address. A 2 of 2 multisig uses 2 separate wallets, requiring two private keys to sign. Both signatures are required to spend the funds which prevent one participant from spending the money without the approval of the other. Once the deposit is completed, both participants can send state updates, including round number, amount to send and signature to each other to realize the transfer without interacting with the Layer 1 blockchain if the balance of both parties is still positive.

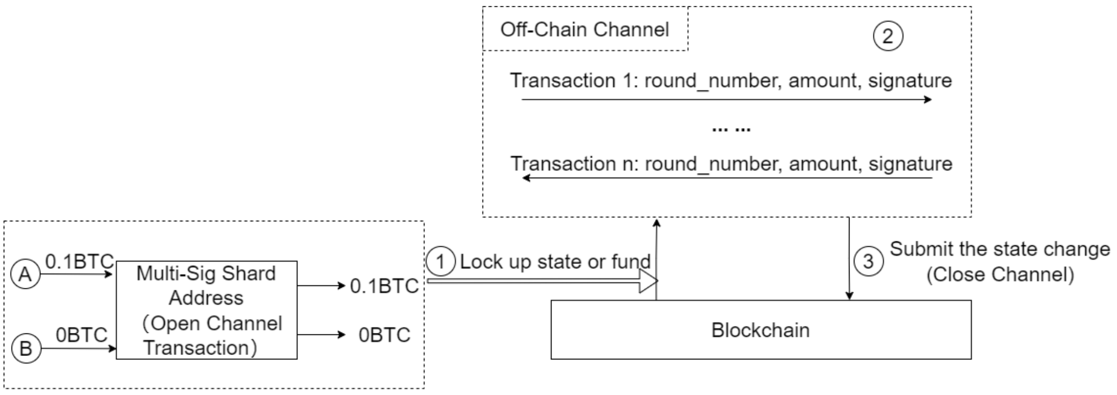

Off-Chain Channel
2
Transaction 1: round_number, amount, signature
... ...
Transaction n: round_number, amount, signature
A
0.1BTC
B
0BTC
Multi-Sig Shard Address (Open Channel Transaction)
0.1BTC
0BTC
1 Lock up state or fund
3 Submit the state change (Close Channel)
Blockchain

Figure 2.4: State Channel Workflow

When two participants want to stop the channel together, they can sign a closing proof which means they all agree to close the channel. Once verified by the blockchain, this state channel can be closed immediately, and the settled balance will be refunded to the two participants. If only one participant starts the exit process, he can submit the final status to the blockchain and wait for the Challenge Period [20]. The challenger or another participant can verify the transaction's correctness off-chain and issue a challenge if they disagree. The blockchain will programmatically check the claim and execute it on-chain. Unless both parties agree to exit in the first case, participants may have to wait a relatively long period before they can withdraw. Also, they need to monitor the blockchain frequently to ensure that their counterparties have not used a past state to withdraw. For convenience, the monitoring process is usually handed over to a third party using so-called Watchtower technology [21].

Hash Time Locked Contracts

The Hash Time Locked Contract (HLTC) is a kind of agreement to achieve value transfer. This concept first appeared in the BitcoinTalk forum in 2013 [22]. It was first applied and implemented in Bitcoin Lightning Network [23]. A Hash Time Lock is divided into two parts, 1) time lock and 2) hash lock. A time lock means that both parties agree to submit the transaction within a certain period to make it valid. The transfer beyond that time is invalid. Hash lock means that for a hash value H, if the recipient can provide the preimage R and Hash(R)=H, the transaction is valid, otherwise, it is invalid. If the transaction is unsuccessful for various reasons, the time lock can allow participants to get their funds back, avoiding losses due to fraud or transaction failure.

Users who are not directly connected can establish a payment path formed by a series of end-to-end channels through hash time locked contracts. We use an example to illustrate the HTLC workflow.

We suppose Alice wants to send a transaction with Bob. The transaction amount is 0.01 BTC. However, there is no direct channel between Alice and Bob. Alice needs to go through Carol, a node that both connect with, to establish a path with Bob for transactions.

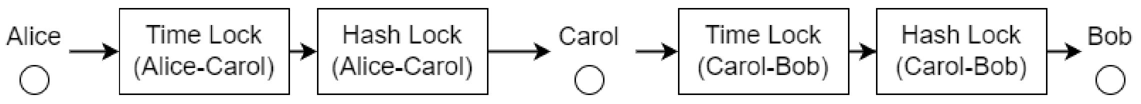


Figure 2.5: HTLC and Payment Path

(1) Bob sets the preimage R and tells Alice the Hash value H. H=Hash(R).
(2) Alice makes a conditional payment to Carol through HTLC. If Carol can provide a correct R within a period, Alice pays 0.01 BTC to Carol. In this case, R between Alice and Carol is a Hash Lock, and the time period is a Time Lock. Only when Carol meets these two conditions can she get money from Alice. Otherwise, 0.01 BTC will not be sent to Carol.
(3) Carol also makes conditional payment to Bob through HTLC. If Bob provides R within a period, Carol pays 0.01 BTC to Bob. Otherwise, 0.01 BTC will not be sent to Bob, and Carol cannot provide a correct R before the period. Then 0.01 BTC that Alice has paid to Carol will be returned to Alice.

From this example, we can see that all conditional payments in an HTLC are either completed or not completed, with all participants getting their funds back. The process is atomic. The preimage R can be regarded as a receipt. R and funds flow in the opposite direction. In an HTLC, the transmission of the preimage and the hash value can all be completed off-chain. Hash Time Locked Contracts are the basis of off-chain payment channels and cross-chain transactions.

Figure 2.6 shows the workflow of state channel based approaches for e-commerce. A customer, Alice, and a merchant, Bob, generate a channel. They share and lock up deposits via a multi-signature wallet in Bitcoin or a smart contract in Ethereum which will be discussed in section 2.3.6. This step gives an initial state for their channel. Participants can then generate and sign transactions to counterparties via the channel in Layer2. Transactions in Layer2 are off-chain messages which cause balance changes between participants. Customer Alice can continually or periodically purchase products if she has enough funds in the channel. When they choose to close the channel, both can sign and commit that final state to the blockchain and withdraw their funds according to the final state. State changes are stored by participants locally. If one of them finds their counterparty did not commit the up-to-date state to the blockchain, he can use a current state to rebut his counterparty.

Compared with the Layer1 mechanism, the Layer2 technology has more advantages in e-commerce payment. Taking Alibaba as an example. Most of Alibaba's e-commerce orders are paid to merchants through Alipay[24]. On average, active customers make 52 purchases per year on Alibaba's e-commerce platforms using Alipay[25]. As a third-party payment system, Alipay can not only process e-commerce transactions but also can carry out micropayments such as grocery purchases, transportation card recharge and other payments. The per capita payment is about 4.0 transactions per day (that is, 1,460 transactions per year) in 2019, with the characteristics of small amount and high frequency[26]. But Alibaba control all the transaction data which is a drawback. Let's take blockchain technology as a possible solution. If all transactions occurred in blockchain Layer1, that would incur huge transaction fees. At the same time, such a transaction frequency will cause many transactions to wait for verification on the blockchain and cause high latency. Moving the transaction to Layer2 can solve these problems very well. If the Layer2 technology like State Channel is used, customers only need to pay the on-chain transaction fee to open the channel, then conduct dozens or hundreds of small transactions, which greatly reduces the cost and latency.

There are several state channel based solutions for Bitcoin and Ethereum.

BitcoinJ [27] is a library which implemented a unidirectional payment channel for Bitcoin client/server application in 2013. The transaction can only be sent from one side to another, and only the sender has to deposit to the channel which suits scenarios like a coffee shop. It was one of the first implementations of micropayment channels.

The Lightning Network is another payment channel based implementation proposed by Joseph Poon and Thaddeus Dryja in 2016 [23]. Unlike BitcoinJ, the Lightning Network supports bidirectional payment channels, which means transactions can be generated from both participants. It was launched on the Bitcoin Mainnet in 2018. Participants need to monitor the blockchain continually or use watchtower services [28] to avoid fraud, checking that their counterparty has not submitted an out-of-date state.

Eltoo [29] is an improvement on the Lightning Network proposed by Lighting Labs in

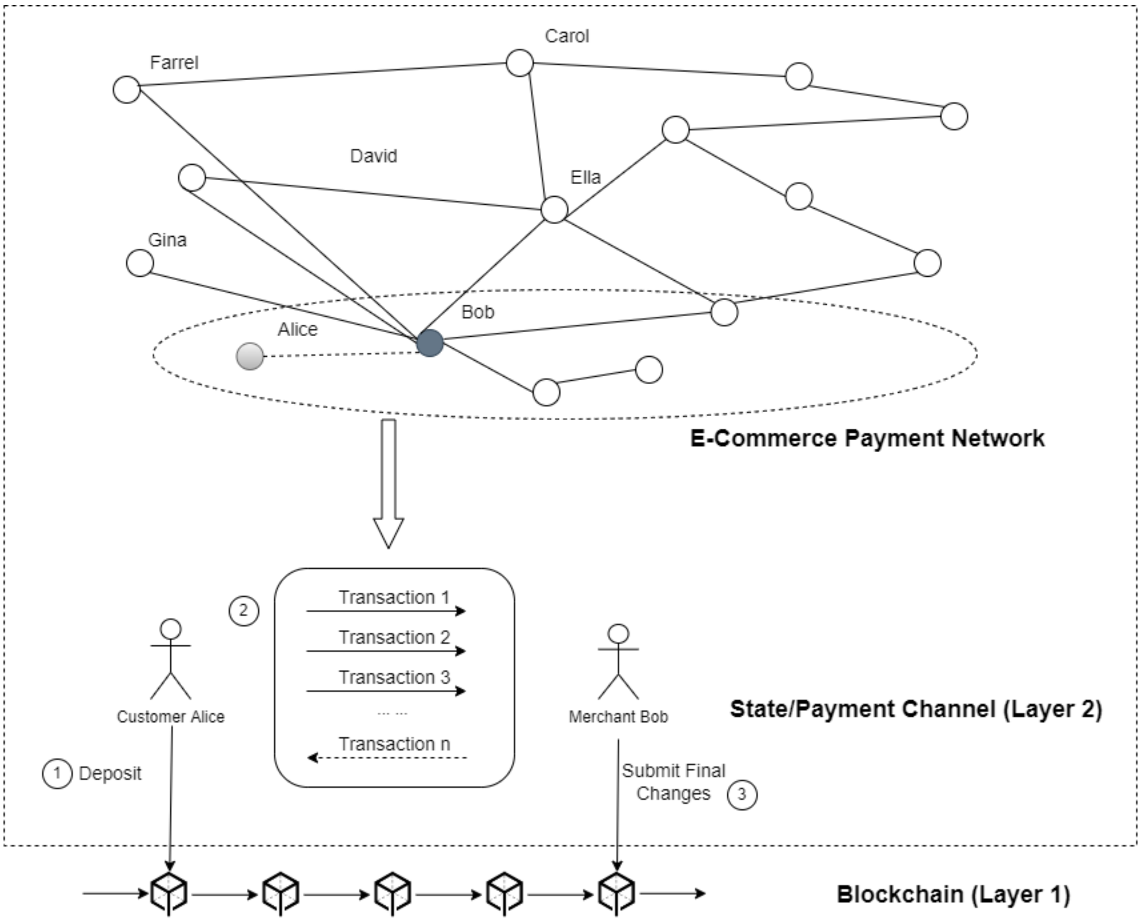


Figure 2.6: State Channel Based Transactions

2017. It uses a flag for the Bitcoin protocol, SIGHASH_NOINPUT, which helps to order state changes and ensure the latest state will be committed to the blockchain instead of the fraudulent old state.

The Raiden Network [30] is a state channel based solution designed for the Ethereum network. It is similar to the Lightning Network but uses smart contracts to implement operations which allow multiple deposits without closing channels. A smart contract is code that runs on the Ethereum blockchain which establishes the terms of agreement agreed by the customer and merchant. It also can allow payment using ERC-20 standard tokens rather than the underlying Ethereum coin, ETH. BTC or ETH are the native assets of an underlying blockchain, whereas tokens are units of value that blockchain-based projects built on an existing blockchain using smart contracts. The first release of the Raiden Network, µRaiden Network [31], which allowed unidirectional many-to-one payments, was launched on the Ethereum mainnet in 2017. The Raiden "Red Eye" version was released in 2018, enabling users to send bidirectional payments.

The Lightning Network and Raiden Network can also support payment channel network transfer. A channel network is formed by nodes and the linked channels. If there is no direct channel between two participants, they can send transactions with the help of other nodes and channels. For example, in Figure 2.6, when Gina wants to send 1BTC to Ella without a direct channel, this payment can pass through Bob to Ella with the help of Hashed TimeLock Contracts (HTLCs). However, if intermediaries become unresponsive, participants along the route have to wait until the timelocks expire.

Sprites [32] is a solution which aims to reduce the waiting time in this situation. Sprites introduced a global smart contract called a "PreimageManager(PM)" to organize the state of each intermediate node. The PM can automatically confirm the payment transfer throughout the route.

Revive [33] is another state channel based solution implemented on Ethereum, which was proposed by Rami Khalil and Arthur Gervais in 2017. It provides a scheme to transfer deposits among users' channels to rebalance the existing payment network without interrupting channels.

Rapido [34] is a solution based on Bitcoin. It splits the payment value into small shares and sends them via different routing paths to the destination, which will be helpful when there are not enough deposits in a single channel.

Duplex Micropayment Channel [35] is based on Bitcoin and was proposed by Christian Decker in 2015. Participants have to set up two unidirectional payment channels to enable bidirectional payments. When a new transaction is generated, the timelock of this transaction will be lower than the previous one, ensuring the latest state will be committed to the blockchain. However, after the timelock is "exhausted", channels must be closed and reopened to enable more transactions.

Channel Factory [36] improves upon the Duplex Micropayment Channel to enable more transactions generated off the blockchain by resetting the timelock. It also enables several participants to lock funds in a shared wallet and generate transactions among these participants.

2.3.2 Applications Using State Channel based Payments

There are several applications based on blockchain technology using or attempting to use state channel based Layer2 approaches to scale their payments system. Connext Network [37], a state channel based p2p micropayment infrastructure, was launched on the Ethereum mainnet in September 2018. Counterfactual [38] proposed to use a state channel protocol to allow more applications to achieve instant payments in November 2018. It provided a test version, Web3Torrent [39], in 2020 and a layer2 library in 2021. Funfair [40], a game application based on Ethereum, implemented payment channels for online game transactions. Magmo [41], another game application using state channels, presented its demo in Devcon4, a famous Ethereum Developer Conference, in 2018. These applications only use state channel technology for instant payment but do not involve a state channel network. Celer Network [42] is a layer 2 scaling platform that supports blockchain applications. It launched Eridanus, the world's first generalized state channel network in 2019. Celer Network posted a cross-chain initial version, cBridge 1.0, in 2021, and the transaction volumes on cBridge grew to $170 million per month [43]. The development of cBridge2.0 is in progress. PyRos [44] is a state channel based access control system for the public blockchain network, which is used for data transactions, and sharing between participants and it was first proposed in 2020.

2.3.3 State Channel Based Approaches Discussion

Table 2.2 shows a comparison of State Channel based approaches which can be classified into unidirectional and bidirectional. BitcoinJ Micropayment Channel and µRaiden Network are unidirectional payment channels which support transactions from only one direction, like in a coffee shop scenario. Compared with unidirectional payment channels, bidirectional payment channels use a shared account to link different participants and enable payment to each other. Unidirectional payment channels are simpler than bidirectional payment channels. However, in reality, in the e-commerce industry, if customers get refunds from a merchant, the unidirectional approaches will cause more on-chain transactions than bidirectional ones. For example, the average return rate of Singles' Day was 27% in 2017 [45]. That means a significant number of merchants have to be able to send transactions to customers.

These approaches can be classified into two categories by how they prevent fraud. One is time-based bidirectional channels, like the Duplex Micropayment Channel and the Channel Factory, which uses two unidirectional channels with a limited lifetime to enable bidirectional transactions. The other kind of bidirectional channels is punishment-based bidirectional payment channels, like the Lightning Network, Raiden Network, Sprites, Revive and Rapido, which use a punishment mechanism to ensure honesty between participants. The dishonest party may lose all the funds if fraud is detected. There are no other costs during the transactions as long as enough funds are in the channel. However, this still needs every participant to monitor each other's behaviour to avoid loss.

Table 2.2: Comparison of State Channel Based Approaches

Approach	**BitcoinJ**	**Lightning**	**Raiden**	**μRaiden**	**Sprites**	**REVIVE**	**Rapido**	**Duplex**	**ChaneelFactory**
Unidirectional	✓	×	×	✓	×	×	×	✓	✓
Bidirectional	×	✓	✓	×	✓	✓	✓	×	×
Timelock	×	×	×	×	×	×	×	✓	✓
Punishment	✓	✓	✓	✓	✓	✓	×	×	×
Bitcoin	✓	✓	×	×	×	×	×	✓	✓
Ethereum	×	×	✓	✓	✓	✓	✓	×	×
Cryptocurrency/ Token	Bitcoin	Bitcoin	RDN & ERC-20	RDN & ERC-20	ERC-20	ERC-20	ERC-20	Bitcoin	Bitcoin
Multi-hop Network	×	✓	✓	×	✓	✓	✓	×	×
Multiple Deposits	×	×	✓	×	✓	✓	✓	×	×
Multiple Withdrawals	×	×	✓	✓	✓	✓	✓	×	×

A Payment Network here means transactions can be sent from a customer to a merchant via intermediaries, which could be provided by an e-commerce platform. The BitcoinJ Micropayment Channel and µRaiden Network are unidirectional payment channels which do not support a payment network. Duplex Micropayment Channel and the Channel Factory can support a group of participants. Even though they allow multiple participants, there are no payment networks in the Duplex Micropayment Channel and the Channel Factory. However, they may be suitable for "Group Buying" e-commerce models. The Lightning Network, Raiden Network, Sprites, Rapido and Revive can transfer payments via intermediaries. Compared with the Lightning Network and the Raiden Network, Sprites decreases the time cost of unlocking funds from the channels using a PreimageManager contract. Rapido allows payments to be transferred via multiple channels in parallel. However, if participants frequently generate transactions in one direction, deposits can be easily exhausted, and participants have to redeposit to this channel. This causes more on-chain transactions. Revive can help intermediaries rebalance their channels to support more transactions without closing and reopening them. An e-commerce platform can use this rebalancing feature to support more transactions in one-directional frequently used channels. However, all participants' signatures need to be collected to rebalance channels. If one or more participants are unresponsive during the rebalancing process, all others must wait for a time-out to roll back the status. During this period, the state of these channels is frozen, and no one can send transactions. If this happens frequently, it may result in a denial of service attack, providing a bad user experience for customers. The settings of timeouts or relative thresholds will seriously affect the efficiency of this off-chain layer.

From the comparison, the Lightning Network and Raiden Network are the two most essential implementations of state channel protocols. They can support transactions via intermediaries. They also have drawbacks, such as the time cost of unlocking funds from intermediaries. Sprites achieves the simultaneous release of funds from intermediaries but increases the centralized control risk. Rapido and Revive improve on the Lightning Network by continually sending transactions without immediately closing channels, but they increase the communication costs.

In the Lightning Network, Raiden Network, Sprites, and Revive, intermediaries need to lock the same amount of funds as the end-points to support a transfer. Rapido divides the total value into small amounts, which helps more intermediaries to participate in the network. However, this still leaves the problem of intermediary failure. Intermediaries can be other merchants or nodes operated by e-commerce platforms. If an intermediate node fails during transmission, other participants have to wait for a period to close channels and withdraw their funds. The more intermediaries, the longer participants need to wait. Sprites can relieve this issue using the PreimageManager contract. The unlock request is controlled by a contract. Once the failing node has recovered, funds can be unlocked simultaneously.

Participants can deposit only once using BitcoinJ, Lightning Network, Rapido, and Duplex Micropayment Channel. The Raiden Network, µRaiden Network, Sprites, and Revive allow multiple deposits and withdrawals using smart contracts. Due to the use of smart contracts, Ethereum-based approaches have a more ability to support continuous deposits and withdrawals with fewer on-chain transactions.

2.3.4 Advantages Compared with On-Chain Payments

(1) Privacy. Everything happens inside the channel, not on the blockchain. The public only sees the open and close transactions.
(2) Low transaction fees. For frequent transactions, the transaction costs can be effectively reduced. Creating a channel or deploying a contract will have initial costs. After deployment, the cost of each status update within the channel is small.
(3) Instant payment. Every transaction on the blockchain needs to wait to be included

in a block and receive confirmation from multiple blocks afterwards. However, in state channels, participants can receive their status updates immediately.

2.3.5 Issues of State Channels

State channels require the full availability of all participants. Since the final closure of the channel and the final status submission may be submitted by a malicious party, there is a risk of losing funds if participants ignore completing the transaction.

2.3.6 Sidechain Based Approaches

Sidechain based approaches are another option for Blockchain Layer2 based e-commerce industry. A sidechain is a secondary blockchain that runs independent of the main blockchain (like Bitcoin or Ethereum) and connects to the main chain to allow assets or tokens from one blockchain to be securely used in a separate blockchain and submit the final status to the main chain. This approach was first proposed in 2014 [46]. Figure 2.7 shows the workflow involved.

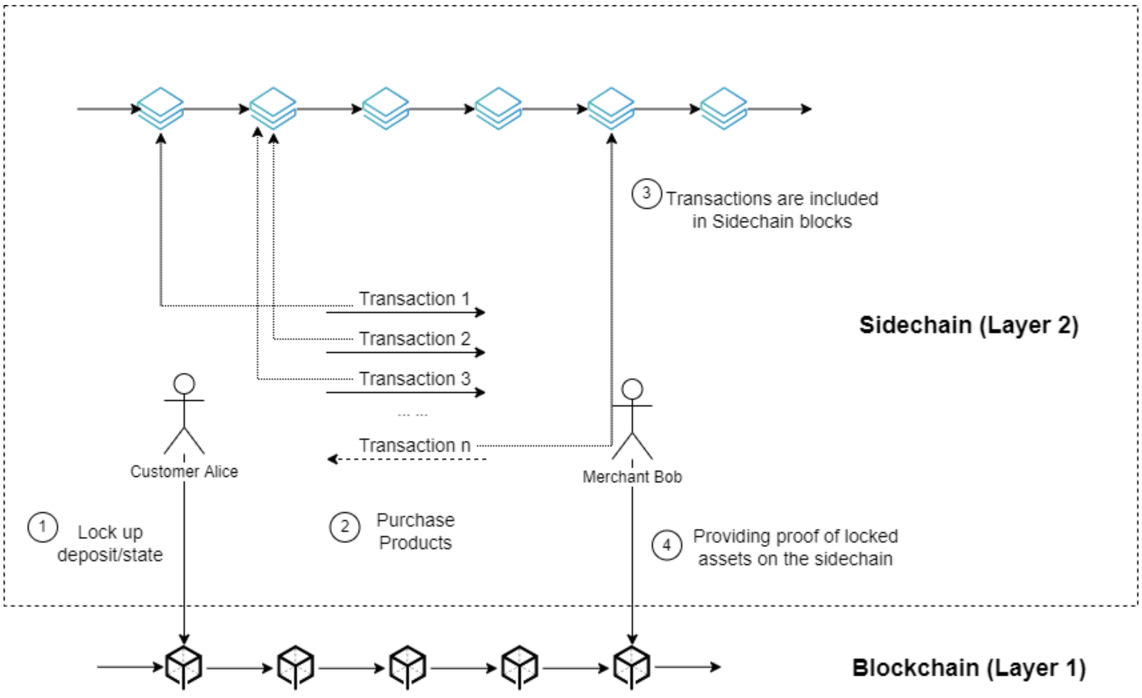


Figure 2.7: Sidechain based Transactions

Customers and merchants/platforms lock funds on the mainchain via scripts in Bitcoin or smart contracts in Ethereum. The exact value of alternative tokens(tokens used in sidechains) will be released on the sidechain. Customers can use these tokens to generate transactions on the sidechain. These transactions can be included in sidechain blocks.

Participants can provide a proof of locked assets on the sidechain to the mainchain and withdraw their funds in the mainchain. Participants in the blockchains can lock their funds with high transaction volume and low transaction speed, and unlock other coins in chains with low transaction volume and high transaction speed. The main chain can ensure the funds' security while the sidechains allow more micropayments and frequent e-commerce trading.

There are several sidechain based solutions for Bitcoin and Ethereum. Rootstock(RSK) [47] is one of the most well-known two-way pegged Bitcoin sidechains, which was created by the Bitcoin core team in 2013. Users lock up their BTC and get an equivalent amount of RBTC in the sidechain, rootstock. Rootstock introduces Federations, a set of semi-trusted third-parties (STTPs), to help lock and release bitcoins on the Bitcoin blockchain. Rootstock requires the Federations to have the ability to check the correctness of the release of BTC funds. There are 25 leading bitcoin industry companies in the RSK Federation list, including Bitfinex, Bitbulls and Huobi. These companies act as notaries in the RSK Federation. When STTPs receive commands from the RSK blockchain, only the majority of STTPs need to approve the transaction, and BTC funds can be released on the Bitcoin blockchain.

Plasma [48] is a sidechain based solution for the Ethereum blockchain which Joseph Poon proposed in 2017. It is a framework that uses a Proof-of-Stake protocol to maintain the sidechain and uses a UTXO model, like Bitcoin, to handle payments. Proof-of-stake is another way for transactions to be verified on the blockchain. Unlike Proof-of-work, miners should pledge a stake with coins of their own before validating transactions under the proof-of-stake. If blocks that miners validated can be added to the blockchain, miners can receive rewards; otherwise, the miners' stake can be lost which keeps the blockchain secure and address energy usage. Smart contracts are used to lock funds in the Ethereum blockchain. The Plasma team published their first implementation, "Minimal Viable Plasma"(MVP) [49], in 2018, a simplified version of Plasma. The plasma contract has been deployed to the Ethereum blockchain mainnet. There are three entities involved in Plasma MVP, the Ethereum blockchain with Plasma contracts, Operators, and clients. The Ethereum blockchain, together with Plasma contracts, are used to lock funds from clients and verify the commitments. A Plasma Operator is a centralized authority which can generate plasma blocks with specific rules. The operator is responsible for collecting transactions into blocks and will send the hash of blocks which include transactions, to the root Ethereum blockchain and let the root chain make the confirmation. Clients are users who want to transfer their funds. Participants use signatures to verify or confirm a transaction. For example, when Alice sends a transaction, she needs to sign a transaction, meaning she saw it and passed it to Bob. Bob then signs it, which means he also saw it, and now the transaction is valid.

Plasma Cash [50] is another implementation of Plasma proposed by the Plasma team in 2018. It uses Non-Fungible-Tokens(NFTs) to enable less user data checking and Sparse Merkle Trees to check the ownership of coins. NFTs are a kind of token which are not interchangeable. A fungible token is like gasoline. One unit is replaceable by any other. But a non-fungible token is like an airplane ticket with a customer name, departure time and seat number and is unique. Some other implementations include More Viable Plasma (MoreVP),Plasma Snapp, Plasma Debit and Plasma Bridge. MoreVP reduces the number of the exchanged messages compared with Minimal Viable Plasma. In MoreVP, the latest transaction has the highest priority, ensuring the new transaction can be committed to the blockchain without exchanging signatures like in Minimal Viable Plasma. Plasma Snapp uses a double-signed mechanism which ensures all owners for this block can prove transactions are valid and up to date, preventing fraud. In Plasma Snapp, the operator sends a hash of transactions to their owners. When participants confirm transactions are included and valid, both need to sign the hash. Otherwise, the hash of that block is invalid. Plasma Bridge proposes using the plasma chain to connect two layer1 chains, enabling coin exchange between two main chains. In Plasma Debit, when a user wants to pay another user, they pay the operator and have the operator pay the other user.

2.3.7 Applications Using Sidechain based Payments

OmiseGo, a Decentralized Exchange platform, is building Minimal Viable Plasma to increase the number of transactions per second they can support. Loom Network is a multichain interoperability platform which used Plasma Cash to support its transactions in 2018. Liquid, a commercial sidechain implemented by Blockstream, moves funds between exchanges. POA is an Ethereum sidechain which enables a 5 second block interval in their sidechain.

2.3.8 Sidechain Based Approaches Discussion

Table 2.3 presents the comparison of Sidechain based approaches. Sidechains or Plasma can help to deal with more transactions than the main blockchains. Unlike state channels which can be opened and closed by participants, once a sidechain is established, it becomes permanent. Participants can only lock and unlock funds from the sidechain instead of closing the sidechain. Transaction records in sidechains are also permanent.

Table 2.3: Comparison for Sidechain Based Approaches

Approach	**Third Party**	**Native Token**	**Bitcoin or Ethereum**
Pegged Sidechain	Federation	BTC,sidechain token	Bitcoin
Rootstock	Federation	BTC,RBTC	Bitcoin
MVP	Operator	ETH,ERC-20	Ethereum
Plasma Cash	Operator	ETH,NFT	Ethereum
MoreVP	Operator	ETH,ERC-20	Ethereum
Plasma Snapp	Operator	ETH,ERC-20	Ethereum
Plasma Debit	Operator	ETH,ERC-20	Ethereum
PlasmaBridge	Operator	ETH,ERC-20	Ethereum

However, participants need to trust a group of third parties, such as a Federation (Cryptocurrency Exchange Platforms) or Operator, to ensure security tokens are locked. This federation or operator structure adds another layer between the main chain(Layer1) and "sidechains"(layer2) which increases risk. In addition, "sidechains" have to maintain their security. For example, if there are not enough miners working on the sidechain, it may be easy to subvert.

2.3.9 Advantages Compared with On-Chain Payments

(1) Low transaction fees.

(2) Permanent. As long as a sidechain is created, it can be used by anyone on the main-chain.

2.3.10 Issues of Sidechain based Approaches

(1) Less privacy. Compared with state channel based approaches, transactions on sidechains are not private. All participants involved in the same sidechain can view these transactions.

(2) High initial and maintenance costs. Sidechains have their consensus mechanism and security level. It means miners also need to work on sidechains. If there is not enough mining power to ensure the security of a sidechain, it could be attacked.

(3) Third-party or intermediary. Sidechain based approaches require federations or operators to connect the original chain and sidechain. These intermediaries are easy to be attacked.

(4) Long withdrawal time. When participants request withdrawals in the plasma blockchain,

they need to wait for a challenge period, about 7-14 days, to allow other participants in the chain to submit proofs to challenge this withdrawal.
(5) Transaction capacity limitation. The transaction capacity describes the maximum number of transactions the network can achieve in a given period. Although the transaction capacity of the sidechain is higher than that of the main chain, there is still an upper limit on the number of transactions per second that can be processed. Take Liquid as an example. The block interval is 1 minute is fixed, and is 10 times faster than Bitcoin [51]. The transactions per second of Liquid is around 7 to 10. Compared to Bitcoin with tps is 3-6, the transaction capacity increase is insignificant.

2.3.11 Rollups Approaches

Rollups are a type of layer2 construction which work by executing transactions outside the blockchain but publishing transaction data on layer1. They operate on top of Ethereum to improve scalability and were first proposed in 2018 [52]. Developers can create decentralized applications, as well as to execute and deploy smart contracts on the Ethereum network by Ethereum Virtual Machine(EVM), a stack-based smart contract operating environment which help to solve the system differences of nodes and realize the results of smart contracts. Rollup technology essentially transfers the calculation process from the main chain to a separate chain called the "Rollup chain". The "Rollup chain" is an independent blockchain. On these rollup chains, after transactions are executed, data is aggregated and then transmitted to the main chain for verification. Unlike sidechain, security can be guaranteed by the blockchain itself. A Rollup Contract acts as an operator to connect the Rollup Chain to the main chain. Figure 2.8 shows an example of the Rollup workflow.

The Rollup Contract is a smart contract which maintains the state root as a Merkle root in the rollup tree. The primary function of the smart contract is to facilitate transfer and verify that everything that happens on the Rollup chain is carried out by the rules. The Rollup Merkle Tree is not stored in the blockchain but can be recomputed from on-chain data. Anyone can publish a batch of aggregated transactions, a highly compressed transaction set containing the previous state root and the new state root. The contract will check whether the old state root in the batch matches its current state root. If it matches, the contract will update the state root.

However, we need to avoid someone submitting a state root to transfer all the tokens in the Rollup to themselves. Two different solutions were proposed, and these two solutions

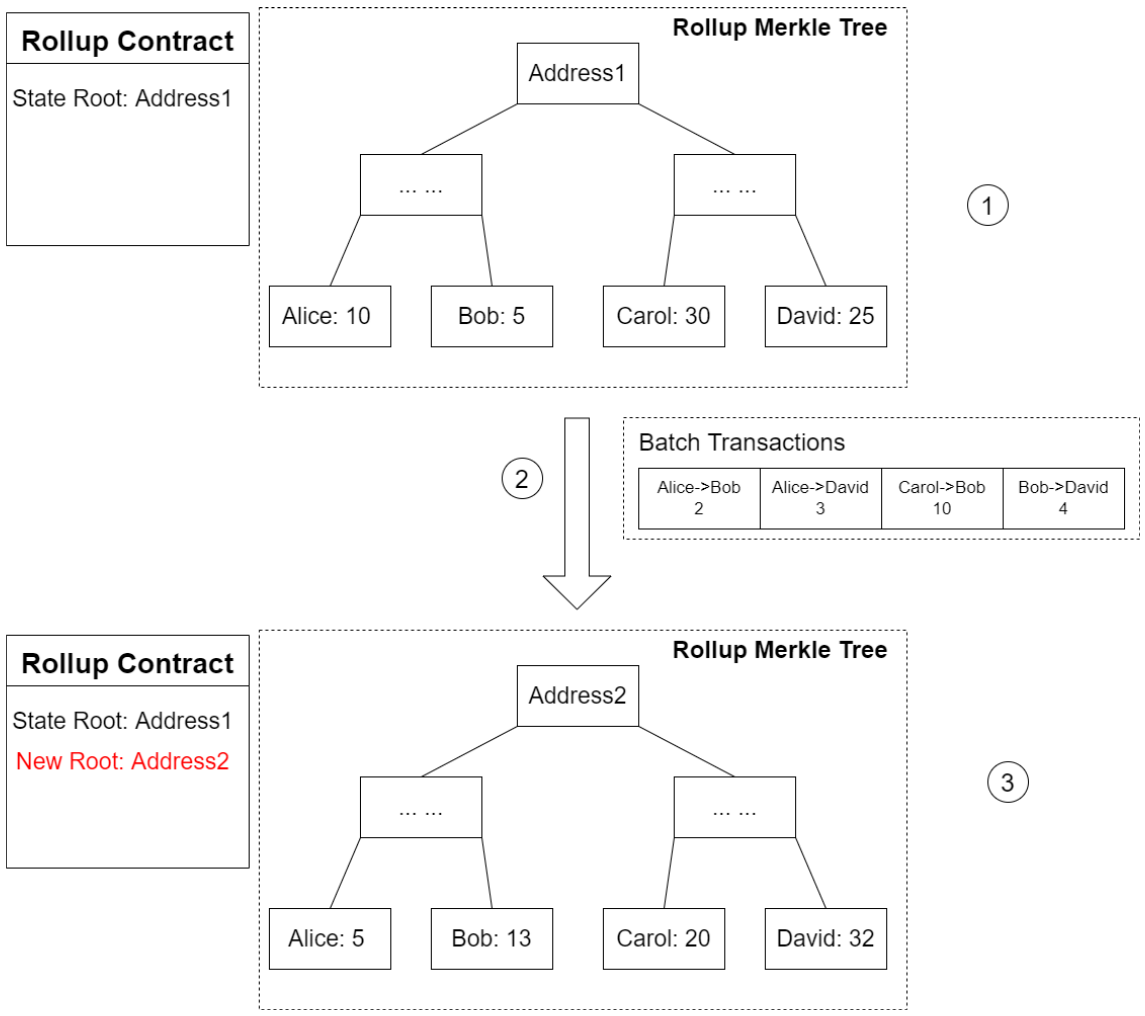


Figure 2.8: Rollups Workflow

gave rise to two kinds of Rollups, 1) ZK Rollup and 2) Optimistic Rollup.

ZK Rollup

A ZK Rollup [53] is realized using zero-knowledge proof technology [54]. For a precise description assertion, a prover interacts with a verifier to convince them that an assertion is correct. The so-called Zero-Knowledge means that the verifier cannot obtain any additional information other than the result of the assertion judgment, wrong or right.

ZK Rollups use validity proofs. In ZK Rollups, each batch published to blockchain layer1 contains a cryptographic proof called a ZK-SNARK. After submitting the transaction batch to layer1, the contract on layer1 can quickly verify the ZK-SNARK proof, and invalid batches will be rejected immediately.

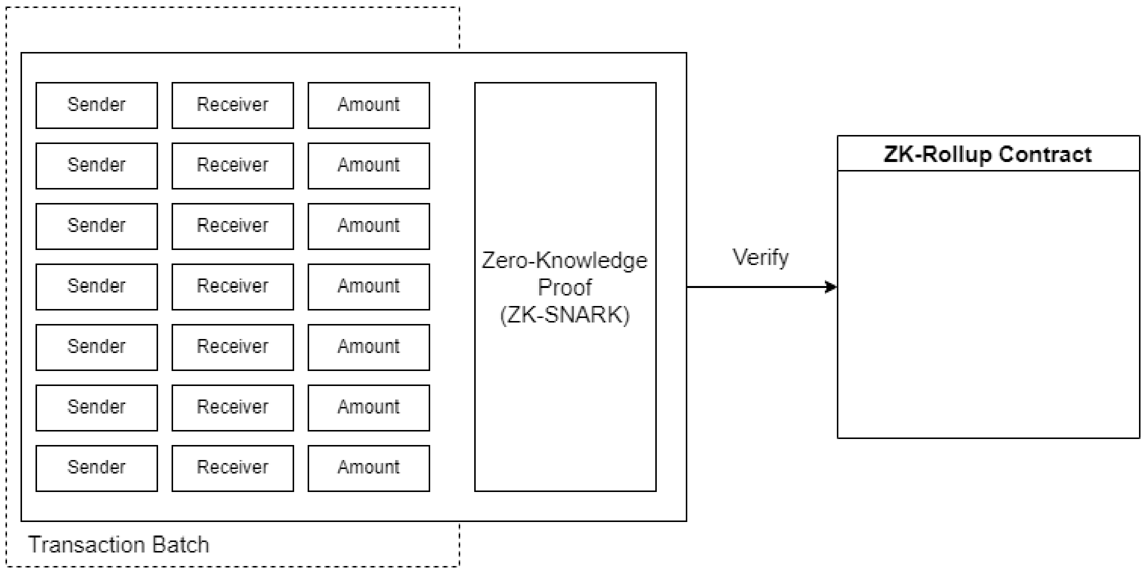


Figure 2.9: ZK-Rollup

Optimistic Rollup

Optimistic Rollups [55] use Fraud Proofs like those in Plasma to verify the transaction batches. The new state root is published by the operator and it will not be checked by the rollup smart contract every time. Validation is only done if someone thinks the new state root is wrong. Because silence will be considered consent, the transaction parties must be always online. Compared to ZK Rollups, Optimistic Rollups is less secure and requires additional measures to prevent “acquiescence” caused by DDoS attacks.

2.3.12 Applications Using Rollups based Payments

Rollups already have some excellent implementations. zkTube [56] uses layer2 ZK-Rollups to provide low transaction fees and high throughput transactions. It ensures security by tracking the history of the state root and the hash of each batch. Loopring [57] is an extension of the Ethereum Layer2 protocol that allows using ZK proof to build high throughput, low-cost payment applications on Ethereum. Loopring supports payments through ZK-Rollups batch processing of off-chain requests. It reduces gas consumption and overall transaction costs. It can process up to 2025 transactions per second. StarkWare [58] uses ZK-STARK encryption certification technology, allowing users to share data and perform calculations with third parties without leaking data. DeversiFi [59] integrates ZK-STARK, allowing more than 9,000 transactions per second. Aztec [60] provides ZK-SNARK services for the main network. Each Rollup aggregates 112 transactions, and by sharing the cost of a single proof, the gas fee paid by users is much lower. Aztec can scale up to 300 transactions per second on demand. Hermez [61] is an open source ZK-Rollup solution that can realize secure, economical and usable token transfer based on Ethereum. By using ZK-Rollups, Hermez can reduce transfer costs by 90%.

Optimism [62] implements Optimistic Rollups which provides instant transactions and scalable smart contracts. Fuel Network [63] supports fast, secure, and cost-effective transmission and exchange of ERC-20 tokens. It has a throughput capacity of up to 10,000 TPS. Cartesi [64] is a Layer 2 platform composed of blockchain and off-chain components. DApps running inside Cartesi can handle almost unlimited amounts of data. OMGX [65] is a Layer 2 solution maintained by the OMG and Enya teams, which provides economic smart contract support and equity rewards. It runs in an EVM-compatible virtual machine, which allows it to process any smart contract running on Ethereum.

2.3.13 Comparison of ZK_Rollup and Optimistic Rollup

Zero-knowledge proofs are different from signature mechanism. Generating zero-knowledge proofs requires extensive computing resources, and different solutions may be developed for different hardware. Therefore, the matter of computing resources may require a relatively significant investment. In addition, the complexity of the technology makes it much more challenging to create EVM-compatible ZK-Rollups, which makes it more difficult to extend general-purpose applications without rewriting the application logic. Compared with Plasma, its advantage is that every transaction data exists on the Ethereum blockchain. It can provide almost the same security level as layer1 transactions. When users join and

Table 2.4: Comparison for Sidechain Based Approaches

	ZK-Rollup	**Optimistic Rollup**
Advantages	1. Transaction data is stored on the blockchain, which can provide almost the same security level as a Layer1 transaction.2. Instant withdrawal	1. It is easier to support general smart contracts 2. Each transaction data is stored on the main chain.
Disadvantages	1.Generating zero-knowledge proof requires a lot of computing resources. 2. It is more difficult to support general smart contracts.	Using fraud proof, users need to wait a long challenge time to exit the network.

exit the network, the speed can be bottleneck. Unlike Plasma or Optimistic Rollup, where the exit process may take several days. This can happen in just a few minutes.

Optimistic Rollup uses fraud proofs. The challenge period may take several days, so the users' exit process is relatively long, which gives a bad user experience. Like ZK-Rollups, compared to Plasma, every transaction data is stored on the main chain, providing a high security level. Unlike ZK-Rollup, it is easier to support a general smart contract. For example, Optimism's OVM technology [66] can be compatible with Ethereum smart contracts. So for developers, its migration cost may be lower.

2.3.14 Advantages Compared with on-chain payment

(1) Scale the blockchain. Rollups can execute a batch of transactions through one on-chain transaction. The main chain does not need to verify the validity of each transaction by executing each transaction. The transaction data is published as function parameters so it will not occupy storage space on the main chain.
(2) Lower transmission cost. An ERC20 token transmission on the Ethereum blockchain consumes about 45000 gas, while the same action in rollup costs less than 300 gas [67].

2.3.15 Issues of Rollups

Rollups can significantly improve the scalability of the main chain. However, this improvement is not without limitations. Although there is only one transaction on the chain, it will be constrained by the gas cost of the transaction data itself. ZK Rollups compress the transaction data, reducing the size of the address data from the original 20 bytes to only 3 to 4 bytes. Each transaction is compressed into 16 bytes, plus a SNARK of about 100-300 bytes, which can theoretically increase Ethereum throughput to about 680 TPS.

Since fraud proofs require signatures, the throughput of Optimistic Rollups is theoretically only 100 TPS.

2.3.16 On-Chain vs State Channel vs Sidechain vs Rollups

These Layer 2 solutions have great potential to increase the number of transactions processed per second, improving the blockchain's scalability. We will now discuss these methods. Table 2.5 shows the comparison of these three categories of layer2 technology.

On-chain transactions take 6 to 10 blocks to be fully confirmed as valid. State Channels can allow users to confirm the transaction instantly. There is a challenge period in Plasma and Optimistic Rollups, so participants need to wait about 7 to 14 days to allow other participants to submit fraud proofs. Compared with Sidechains and Optimistic Rollups, the delay of ZK Rollups is relatively tiny. If there are 1000 transactions in a batch, it takes about 20 minutes to construct a proof on an ordinary server. For time cost and user experience consideration, State Channel is a good choice.

Table 2.5: Comparison of State Channel, Sidechain and Rollups

	On-Chain	**State Channel**	**Sidechain/Plasma**	**Rollups**	
				ZK-Rollup	**Optimistic Rollup**
Delay for Finality	6-10 blocks	Instant	7 days	20 minutes	1-2 weeks
Privacy	Medium	High	Medium	High	Medium
Throughput	Low	Very High	High	High	Medium
Gas Fee/ Transaction fee	High	Very Low	Low	Medium	Low
Monitor	no	yes(but can have watchtower)	yes(Operator)	yes(Sequencer)	yes(Sequencer)

Users can use the wallet or contract address to track the transactions on the blockchain, but the address has been hashed, which improves the privacy. State Channels can only perform a transfer with another party in the channel already. The channel is only visible to participants, not the public. Only the final status will be committed to the blockchain, which protects privacy. All participants on the sidechain/plasma can view transactions generated on this chain. For Optimistic Rollup, not only participants in this rollup chain but also all other Ethereum nodes can verify transactions on the rollup chain. ZK-Rollup uses zero-knowledge technology, which can allow verification without knowing the content of transactions, which protects privacy. For privacy considerations, State Channel and ZK-Rollup have suitable privacy protection mechanism.

The Bitcoin blockchain can support 3-7 transactions per second, while the Ethereum blockchain can support 10-20 transactions per second. There are no limits to the number of transactions per second in the State Channel protocol. The TPS of Plasma depends on the consensus mechanism that the sidechain use. The transactions per second of Liquid is around 7 to 10. Plasma can deal with thousands of transactions per second theoretically. The throughput of ZK Rollups is about 680 TPS, and that of Optimistic Rollups is about 100 TPS. For throughput consideration, State Channels can provide better results.

An on-chain transaction fee, involving Bitcoin or Ethereum, is high. There is no cost if two parties have a direct payment channel under the State Channel protocol. Even if a transaction needs to pass through other nodes, the customer node only needs to pay small fees to these nodes, like 1 msatoshi. Different sidechains have different fee costs. For example, the lowest possible transaction fee on Liquid is 0.1 satoshi/vbyte [51]. Each transaction in Rollups is compressed into 16 bytes, costing less than 300 gas. For transaction fees consideration, State Channel is a better choice.

There is no monitor on the blockchain for on-chain transactions. In State Channel, participants must stay online to keep the channel alive and monitor the transactions within the channel to avoid fraud. Participants can pay a Watchtower to help monitor transactions. Operators on the sidechain should stay online to monitor traffic and send the hash to the blockchain at regular intervals. There are also sequencers on the Rollup chains to help package batches. Operators' or sequencers' election mechanisms are still a work in progress.

In this chapter, we listed the existing blockchain layer2 solutions and performed a taxonomic comparison of these solutions. We find that State Channel based solutions have almost no transaction delay while guaranteeing a high level of privacy. They can provide high throughput and minimize transaction fees. Because transactions are dealt with off the blockchain, when the user cannot monitor the transaction status, a watchtower is provided to help ensure the accuracy of the transaction. Therefore, we decide to take State Channel based approaches as our research direction for layer2-based e-commerce.

Chapter 3

Applying Blockchain Layer2 to Mass E-Commerce

In this chapter, we will discuss considerations on topology, routing algorithms and re-balancing mechanisms when we attempt to apply blockchain Layer 2 technology to an e-commerce scenario involving many millions of customers and merchants.

3.1 Scale of Alibaba and Amazon Networks

Table 3.1: Average Daily Active Users of Amazon.com and Taobao.com

E-Commerce Platform	Customers	Merchants
Amazon	310 million	2.5 million
Taobao	299 million	7 million

In order to apply blockchain Layer2 technology to the e-commerce industry, we first need to know the scale of the existing e-commerce platform. For Amazon, the number of active customers and merchants is 310 million and 2.5 million annually in 2021 [68]. There were 757 million annual active customers of Taobao.com on the China marketplace in 2020 [69]. The average daily active customers are 299 million [70]. The number of daily active users(DAU) on Double-Singles day is 475 million [71]. Alibaba's Singles Day saw record peaks of 583,000 orders per second in 2020 [72]. Table 3.2 shows the average and peak transactions per second of Visa card and Alipay. 24,000 transactions per second are the maximum Visa card can handle [73]. The peak transactions per second of Alipay and Bitcoin are the max they handle.

Table 3.2: Average and Peak Transactions Per Second (TPS) of Visa and Alipay

Payment Types	**Average TPS**	**Peak TPS**
Visa Card	1,750	24,000
Alipay	256,000	583,000
Bitcoin	7	4

For blockchain layer2 based mass e-commerce to support such a number of users and transaction volumes, we need to explore a suitable network topology.

3.2 Topology

A Network's topology is the arrangement of the elements of a communication network composed of nodes, links, and paths.
(1) Node: A node is also known as a network unit. There are various data processing equipment, data communication control equipment and data terminal equipment in the network system. Common nodes include servers, workstations, switches and other equipment.
(2) Link: A link is a connection between two nodes. There are two types, physical links and logical links. The former refers to the actual existence of a communication line, and the latter refers to a logically available network path. In our network, a link refers to a logical link.
(3) Path: Path refers to a series of nodes and links from a node that sends information to another node that receives information. That is, a series of node-to-node links established through the communication network.

Topology affects routing, reliability, throughput, and latency. The lightning network operates at layer2 above the blockchain. The topologies we discuss in this chapter are all for the lightning network. Nodes represent customers, merchants and platform nodes that run the lightning node software. Links or connections are payment channels established between two nodes. In the lightning network, the main reasons for a failed transaction related to the topology are (1) no route is found from source to destination, (2) not enough balance is present along the route to make the payment and (3) a node is unresponsive.

Therefore, the topology design objectives are:
(1) Reliability. Improve the reliability of the network as much as possible to ensure that all data can be received accurately; also consider the maintainability of the system to make

fault detection and fault isolation more convenient. An e-commerce platform is a high-availability and high-reliability system which supports a massive amount of transactions every second. The system should avoid any single points of failure which may cause an overall failure. The network also needs to provide users backup with channels and ensure a smooth flow within the network.
(2) Low cost. When constructing a network, minimising the number of nodes and links is necessary. More channels mean more on-chain transactions. Each channel also needs to be funded with an initial balance.
(3) Response time. The network can provide users with the shortest possible response time between customers and merchants. The path length is an important criterion for measuring response time. The longer the path, the higher the response time takes.
(4) Extendibility. It needs to be considered that when the system is expanded or changed in the future, the network topology can be easily reconfigured. There is no limitation on the number of payment channels that a lightning node can establish theoretically. However, a server could run out of resources to maintain open connections. So the system should operate well under the limit of the number of payment channels. When we look at 1ml.com, the most connected node in the public Lightning network has 1491 payment channels on 1st March 2021.

Amazon has 300 million customers, and the number of merchants was 2.5 million in 2021. We consider applying the following topologies to this size of the system. The topological structures of computer networks mainly include star topology, line/bus topology, ring topology, mesh topology, and tree topology. These are the most common types of network topologies. Many topologies are based on these designs. So, we intend to start with these topologies and analyse the problems and feasibility of applying them to Layer2 based e-commerce.

3.2.1 Star Topology

Each node is directly connected to the central node in a star topology. In Figure 3.1, a platform node is in the centre of the topology to allow merchants and customers to connect to it.

However, we cannot rely on a single node. If the central node fails, the entire network will be paralyzed. In addition, the requirements for the central node are pretty high to support a network with millions of participants daily. Therefore, the star topology is not suited for our e-commerce scenario.

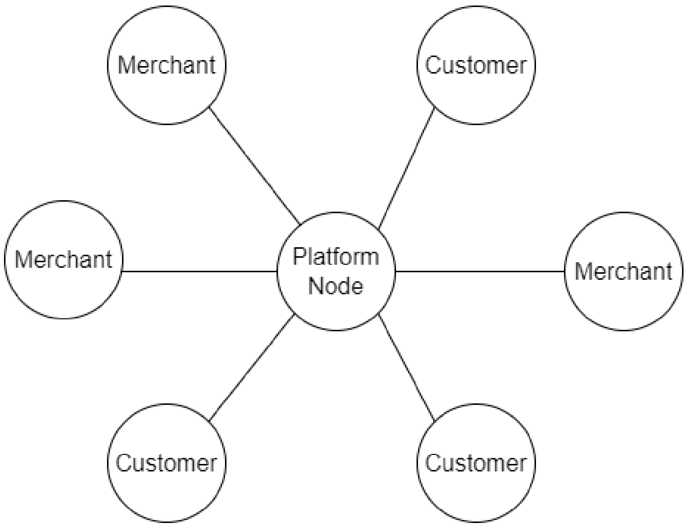


Figure 3.1: Star Topology of E-Commerce

3.2.2 Line/Bus Topology

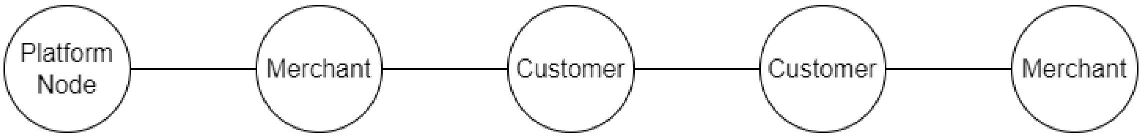


Figure 3.2: Line Topology of E-Commerce

In a line topology, a node communicates directly with all nodes through one bus line. We consider this an extreme case. However, this topology offers no redundant links. When one node is unresponsive, the network is partitioned. In addition, as the network expands, a transaction must go through more intermediate nodes to get to the destination. This type of topology cannot scale to a large number of nodes and cannot meet our requirements.

3.2.3 Ring Topology

In a ring topology, each node connects to another two nodes until all nodes are connected in a ring. Data transmitted between nodes along rings can be bi-directional in the ring. This structure eliminates the dependence of users on the central system when communicating. It reduces the path length compared with the line topology. However, a node failure can divide the network into islands.

3.2.4 Mesh Topology

In another extreme case, we consider the mesh topology where every node connects to every other. It is also called full connected topology. It offers low latency. However, this topology does not suit an extensive scale network. Even though there are no limitations on the number of channels that a node can generate theoretically, it depends on the performance of the device. In practice, a node will have a limit on the maximum number of channels it can support.

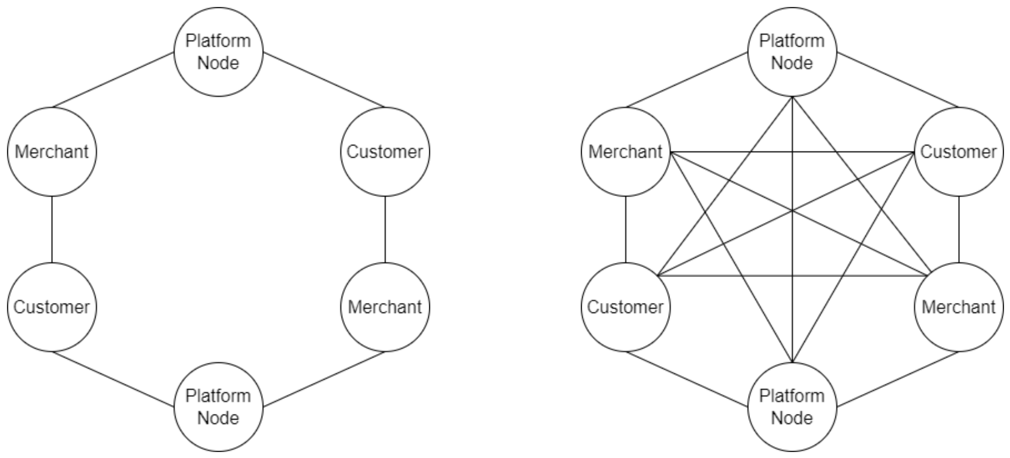


Figure 3.3: Ring and Mesh Topology of E-Commerce (Left: Ring Topology, Right: Mesh Topology)

3.2.5 Tree Topology

A tree is a topology with a tree structure in which all nodes are connected like branches. Each branch can contain multiple nodes. Compared with other topologies, the tree topology has some advantages.

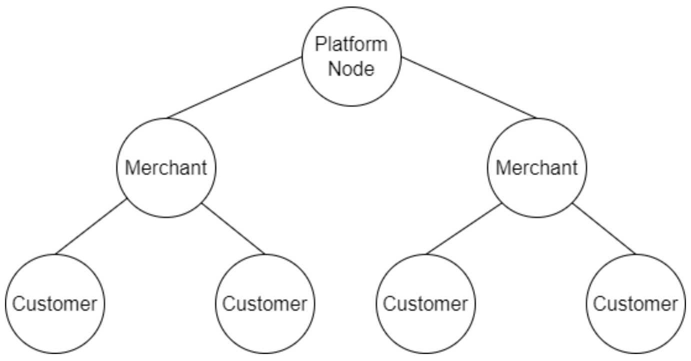


Figure 3.4: Tree Topology for E-Commerce

(1) Easy to expand. A tree can extend many branches and sub-branches. So it is easy to add new nodes to the network.
(2) Easy to isolate a fault. If a specific branch or a particular node fails, it mainly affects a limited sub set of nodes. So, the faulty part can be isolated from the entire system efficiently.
However, if the root node fails, it will cause the entire network to fail.

The Fat Tree topology is a type of tree topology for data centres proposed by Al-Fares, et al. in 2008 [74]. Large-scale data centers also face similar problems to those experienced by the Lightning Network, such as cost, communication performance, and large-scale extension. Fat-Tree can use inexpensive equipment with uniform capacity to build a network and has robust scalability. Fat-Tree has been widely used in scientific research in recent years and can provide a possible solution for our layer2 topology. There are 3 layers in the fat-tree structure, Core, Aggregation, and Edge layer. The participants in each layer are switches. Switches in each layer are core switches, aggregation switches and edge switches. There are links connecting edge and aggregation switches, aggregation and core switches. A set of aggregation switches and edge switches are combined together into small groups, which are called pods.

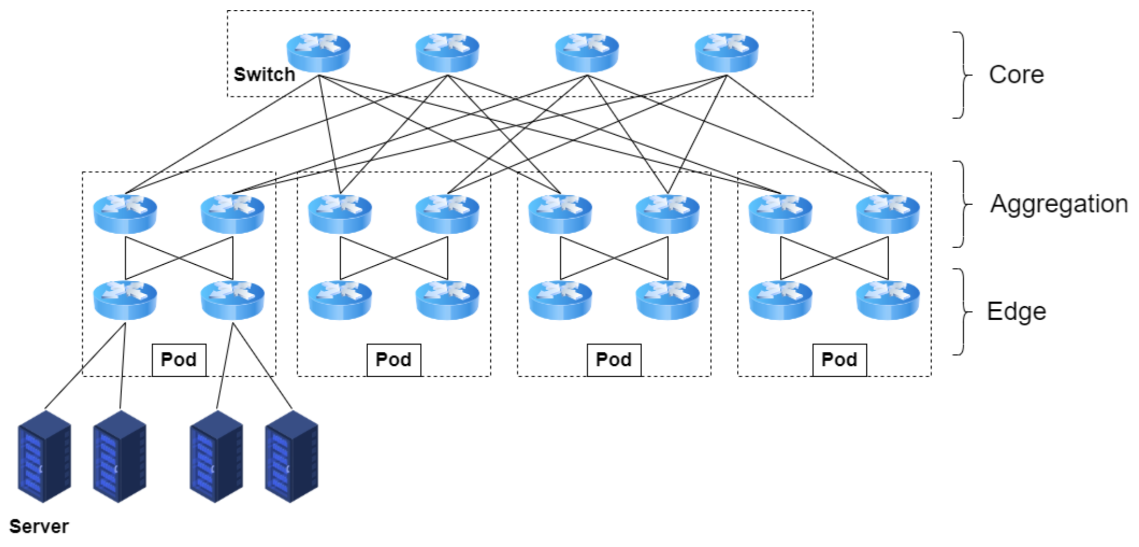


Figure 3.5: Fat-Tree Topology with k=4

In Fat-Tree, k/2 edge switches and k/2 aggregation switches are gathered into pods. k is the number of pods. Each pod has a link to each core switch via aggregation switches. Each switch has k ports. Each edge switch is directly connected to k/2 servers, and the other k/2 ports are connected to k/2 of the k ports in the aggregation layer. There are $(k/2)^2$ core switches in the fat-tree topology. The network can support $k^3/4$ servers. Figure 3.5 shows an example Fat-Tree topology with k=4.

Fat Tree topology has many advantages.

(1) The network adopts a horizontal expansion method. The network can support more servers by increasing the pods in the network horizontally. An increase in the number of pods and switch ports does not cause an increase in the number of intermediate hops between any two edge nodes.

(2) There are multiple links to achieve load balancing. There are multiple parallel paths

between different pods. Traffic can be loaded through multiple links via core switches to avoid hotspots.
(3) Different paths among aggregation and edge switches in each pod provide good fault tolerance and avoid overload problems.
(4) The network is small, meaning the theoretical network delay is small and the real-time performance is good.

However, the fat-tree topology was designed to work with ethernet switches. These were composed of switches with a fixed max number of ports. The size of the fat-tree topology is limited by the number of ports on a single switch which is not conducive to the long-term development requirements of the data center. Moreover, each pod has aggregation switches and edge switches, but only edge switches connect with servers which increases the number of switches required.

If we apply the Fat-Tree topology to layer2 based e-commerce systems, switches are replaced by platform nodes. There is no difference in the Aggregation and Edge layers. All platform nodes in a pod can be connected to customers or merchants. Each platform in Edge Layer connects to at least one node in Core Layer. Besides the interconnection of each pod and connections with Core Layer nodes, all other payment channels can be used to connect with customers and merchants. Multiple paths can be chosen. Moreover, transactions can still go through via other nodes if one node is unresponsive.

3.2.6 Comparison of Topologies for Layer2 based E-Commerce

Table 3.3: Comparison of Different Types of Topology Applying to E-Commerce (✓is acceptable, ×is unacceptable)

Topology	**Star**	**Line**	**Ring**	**Mesh**	**Tree**
Reliability	×	×	×	✓	✓
Response Time	✓	×	✓	✓	✓
Low Cost	×	✓	✓	×	✓
Extendibility	×	×	×	×	✓

Reliability means that even in the event of a failure, the network can provide service capability. A single-point failure is the most common type of failure. It may destroy the most of network in the star, line and ring structures. Mesh topology and tree topology have good reliability. With a fully connected mesh topology, a single failure only affects the failing node itself.

For response time, the longer the path, the higher the response time takes. The path length is the distance between two edge nodes. In the layer2 based e-commerce scenario, the path length is the number of nodes when a transaction is sent from a customer to a merchant. In star topology, each transaction has only one intermediate node, a platform node, on the path. The path length is only 3. However, in an extreme case, a transaction may pass through millions of nodes in the line topology. The path length under the ring topology is not fixed but much shorter than the line topology. However, the extreme case is to pass half of the nodes to find the destination. In a mesh topology, every node has direct connections with each other. In a tree structure, when we limit the height of the tree and expand the tree to a horizontal level, like in a Fat Tree, the path length can be reduced significantly.

Considering the cost of the network, in Star, Line, Ring or Mesh topology, with millions of users, the number of channels is massive and will exceed what a device can support. For the tree topology, leaf nodes do not need to be connected to other leaf nodes. They only need to connect to the parent node. The number of upper nodes in the fat-tree topology is far less than the number of leaf nodes. Even if the upper nodes were mesh-connected, the number of channels generated is still within an acceptable range.

Extendability is the ability to handle increasing scale. When more users join the network, the central node in the star topology and all nodes of mesh topology need to generate more channels beyond their capability. A node may need a longer path to send a transaction in the line topology. When too many nodes are in the ring topology, it will inevitably affect the data transmission rate and prolong the network's response time. In addition, the ring is closed, and it is not easy to expand. Unlike these topologies, creating a sub-tree in the tree structure is easy without interrupting the other part of the network.

The above comparison shows that the tree topology can be applied in Layer2 based e-commerce systems.

3.3 Routing Algorithm

A routing algorithm drives the pathfinding process determining how a message gets from source to destination. In the lightning network, we must consider (1) channel balance, (2) route direction and (3) whether the peer is active or not. These factors will affect the pathfinding result in the lightning network.

Therefore, the routing algorithm design objectives are:

(1) Correctness: to ensure that the data is transmitted from the source node to the destination node.

(2) Simplicity: easy to implement.

(3) Low Overhead: minimize the cost in time and space.

(4) Adaptability: The algorithm can adapt to changes in traffic and network topology.

(5) Stability: changes in intermediate nodes will not affect the path discovery.

The routing algorithm is not a part of the lightning network protocol, but it is an important enabler of a successful transaction. In this section, routing algorithms are divided into two main categories: 1) Single-path routing and 2) multi-path routing. In single-path routing, the transaction will be transferred via one path. In multi-path routing, the transaction will be split into small amounts and transferred via more than one path.

Table 3.4: Popular Layer2 Routing Algorithm

Category	**Technology**	**Algorithm**
Single-Path Routing	Source Route	Dijkstra
	Ants Behaviour	Ant
	Distributed Hash Table	Kademlia
	On-Demand Route	AODV-based Routing
Multi-Path Routing	Network Embeddings	SpeedyMurmurs
	Landmark route	SilentWhisphers
	Network Flow	Flash
	Packet Forwarding	Spider

Most lightning network implementations such as c-lightning [75], Eclair [76], lnd [77], Rust-Lightning [78], Electrum [79], use Dijkstra's algorithms to find the optimal path.

3.3.1 Dijkstra's Algorithm

Dijkstra's algorithm is a routing algorithm to find the shortest path across a weighted graph and was first proposed by Edsger W. Dijkstra in 1956. The main feature is to start from the source node to find the shortest path to the destination by comparing the distance to all nodes in the graph. The algorithm has two sets S and U. The S set contains the points of the shortest path that have been found and the corresponding shortest length. The U set contains the points that have not been visited.

In Figure 3.6, source node A tries to find the shortest path to D. The distance from the source node to itself is 0. The distance from the source node to all other nodes has not been determined yet, so we initially use the infinity symbol $\propto$ to represent this. Since we are choosing the start point A, we can mark A as visited and update the set U. Then we

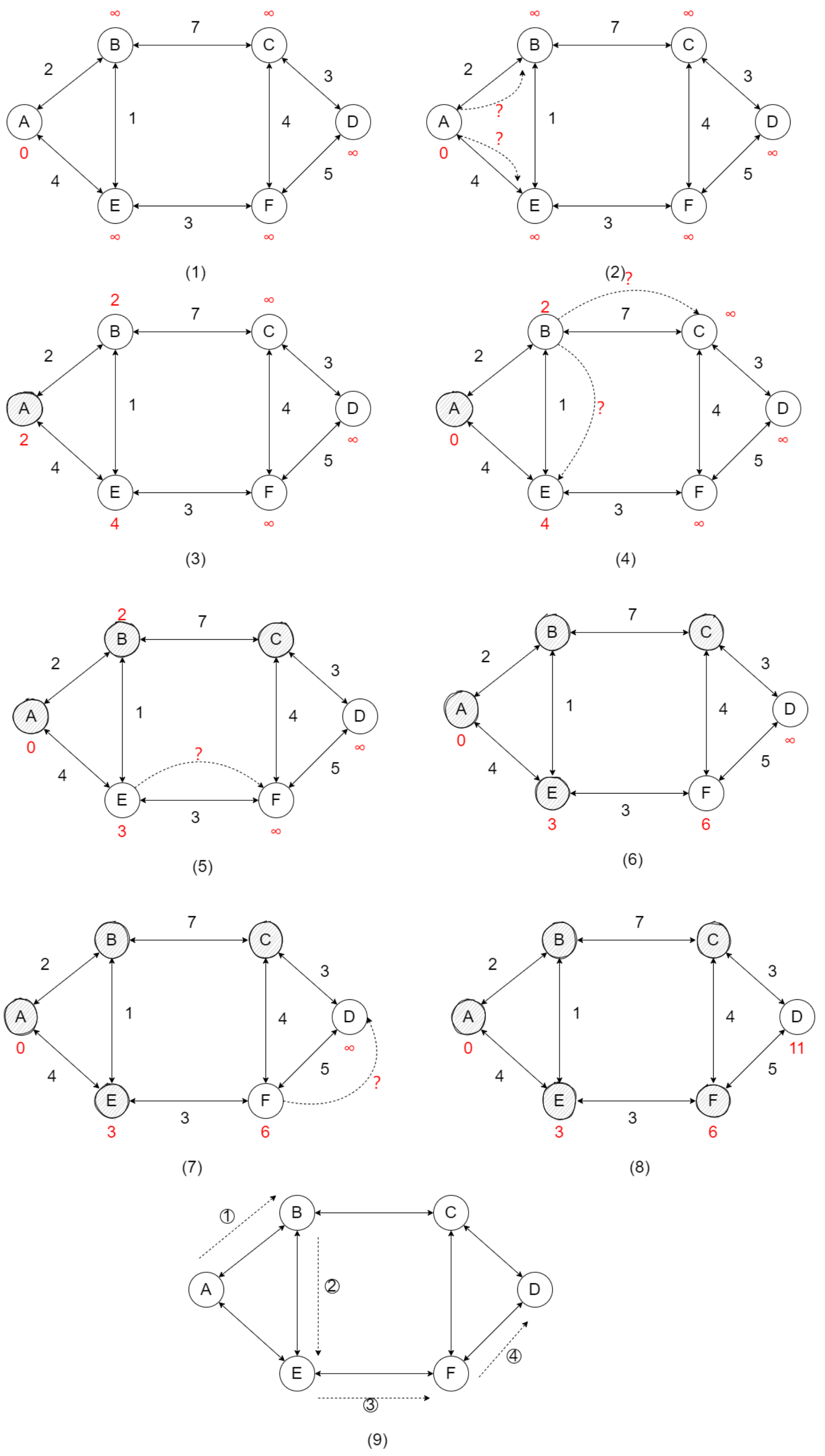


Figure 3.6: Dijkstra Algorithm Shortest Path Finding Process

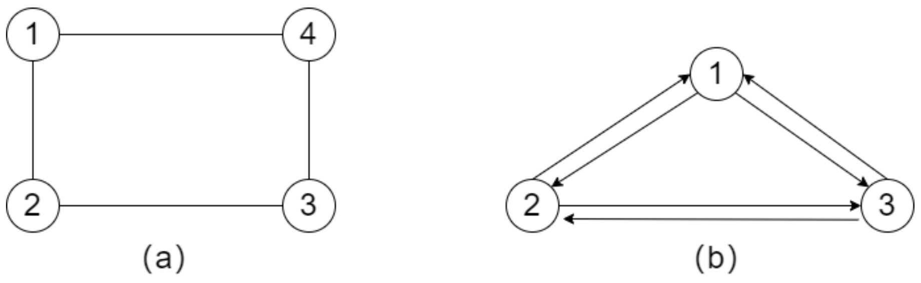


Figure 3.7: Undirected Graph and Directed Graph (Left is Undirected Graph and Right is Directed Graph)

check the distance from A to its adjacent nodes, B and E. After comparing the weights of edges of A to B and A to E, we can see that node B has the shortest distance to the source node, so we mark B as visited, add B to the path and update the distance of B and E to A. Now we need to analyse the new adjacent nodes to find the shortest path to reach them. We will only analyse the nodes that are adjacent to the nodes that are already part of the shortest path. There are two nodes in the set S, A and B. Node C and E are adjacent to nodes that are directly connected to A and B. So C and E are the nodes that we will analyse in the next step. We only update the distance if the new path is shorter. Then we add the corresponding nodes to visited set and repeat the process until all nodes have been added to the path. Finally, we can get a result with the shortest path from node 0 to each node in the graph. In this example, node A wants to reach node D, we just need to follow the shortest path $A \rightarrow B \rightarrow E \rightarrow F \rightarrow D$.

Dijkstra's Algorithm finds the shortest path between the source node and all other nodes in a graph. This algorithm uses the weights of the edges representing the distance to find the path that minimizes the total distance between the source node and all other nodes.

Dijkstra's algorithm does a blind search. The only thing that it can do is to distinguish a non-goal state from a goal state. So it use a lot of time to compare and find the next step. One of the reasons why most Lightning Network implementations use Dijkstra's algorithm is that it is easy to deploy for a small graph. The Lightning Network mainnet has only 22,378 active nodes and 53,111 channels on 6th July, 2021. But with the graph size increasing, a more suitable routing algorithm is necessary.

3.3.2 Ant

The Ant routing protocol [80] was proposed in 2018 for searching paths in the Lightning Network. It implements a decentralized routing algorithm that draws on a "pheromone" to mark paths when ants look for food. While studying the foraging behavior of ants, researchers discovered that ants release a substance called a “pheromone” along the path they pass. The ants in the ant colony can perceive this “pheromone”. The higher the concentration of “pheromone” in a path, the greater the probability of it being selected by the ants. Every ant passing by will leave an additional “pheromone” on the road. In the Ant routing algorithm, start nodes are like ants, and destination nodes are like food in the network. The topology is the entire map. Channels connect to form a path. Nodes store parts of routing information and maintain payment channels with direct neighboring nodes. The process is shown in Figure 3.8.

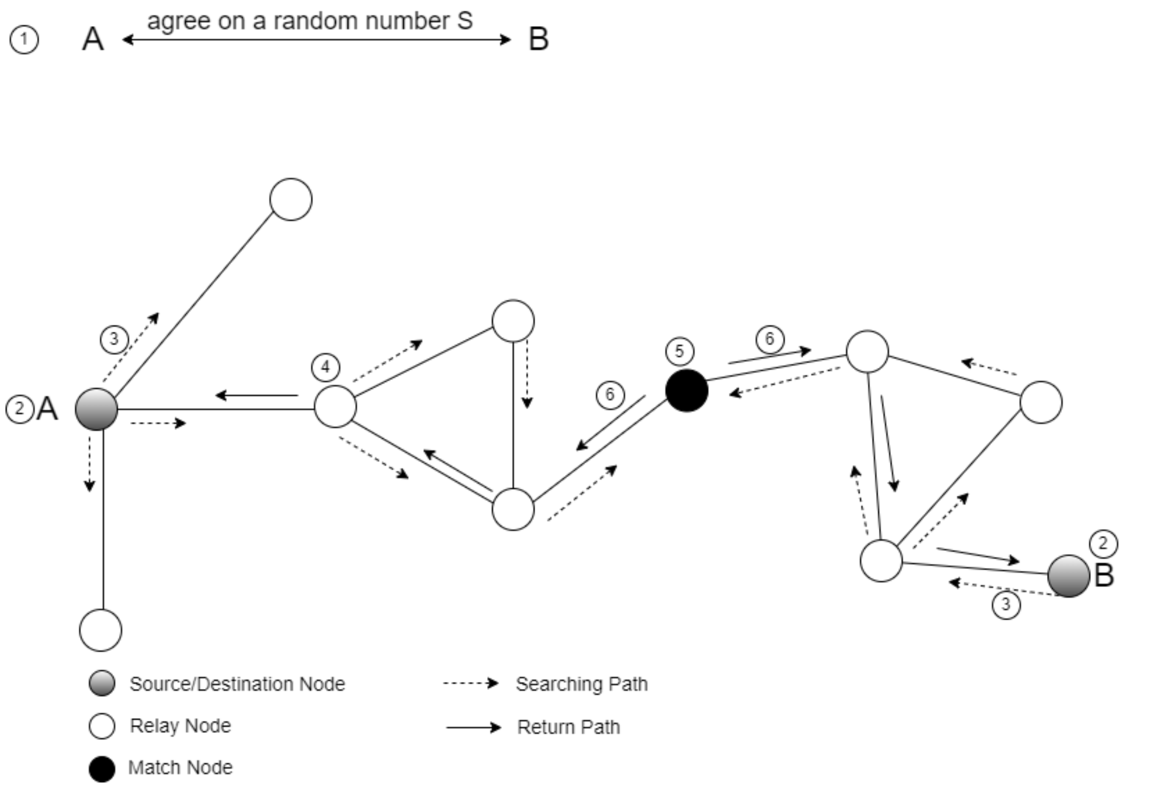


Figure 3.8: Ant Routing Algorithm

(1) The start node (A) and the destination node (B) agree on a large random number S. A creates the number $S(0) = 0 \frown S$, and B creates the number $S(1) = 1 \frown S$. These two numbers are “pheromone seeds”. A and B exchange them in a secure way. We have a match when these two pheromone seeds meet at one node.

(2) They broadcast their seeds to neighboring nodes.

(3) When the neighboring node receives the seed, it will verify whether the seed already exists in its memory.

(4) If the node finds that it has received a previously un-seen seed, it stores the seed in the memory and hands it over to its neighboring nodes. These intermediate nodes are called Relay Nodes.

(5) A match is found when two seeds meet at some node what we call Match Node. A counter is used to count the number of times the same seed has been received.
(6) The Match Node creates two matched seeds and sends these matched seeds back to the source and destination nodes according to the same route. Then the source node and destination node can be connected via a path.

The source node may receive several matched seeds, and it can select one path according to a selection algorithm. Then the source node can process a payment.

Besides the pheromone seed, each node also stores ids of the node from which it received the seed. The id is used to trace the way back to the start node when a matched seed is found. The Ant algorithm would allow finding routes in a payment channel network. Nodes only store data from their neighbors, significantly reducing the stored data size. However, the pathfinding process needs a considerable amount of message exchange to propagate the pheromone seeds. Ant routing algorithm has not been deployed in the Lightning Network. It still needs more simulations to evaluate its performance in a large-scale network.

3.3.3 Kademlia Algorithm

Kademlia is a peer-to-peer Distributed Hash Table (DHT) which uses a node ID to locate resources. It was proposed by Petar Mamounkov and David Mazieres in 2002 [81]. Each

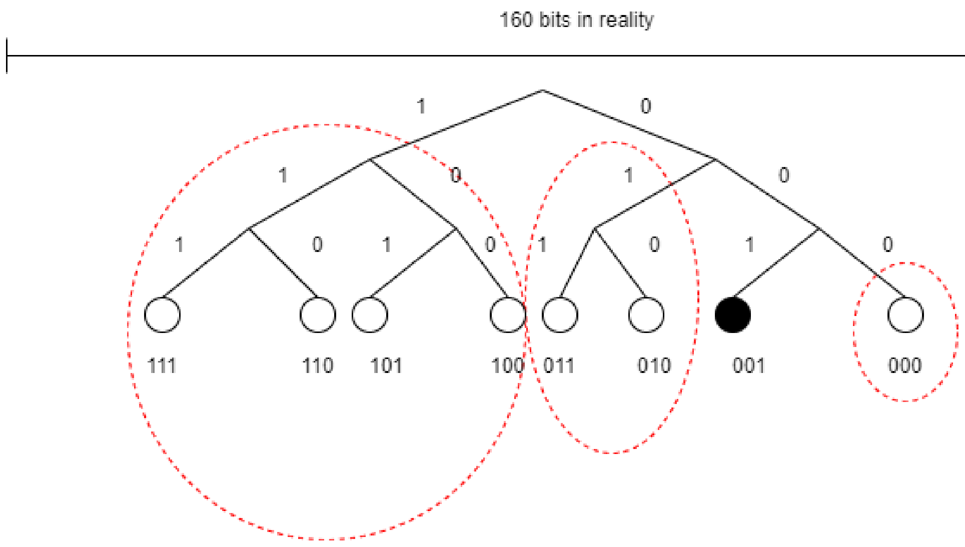


Figure 3.9: Kademlia Structure

node in the Kademlia network has a 160-bit ID value as an identifier. Each node that joins the Kademlia network will be assigned a 160-bit node ID. Usually, this ID value is randomly generated. In the Kademlia algorithm, all nodes are treated as leaves of a binary tree, and the position of each node in the tree is uniquely determined by the shortest prefix of its ID value. In Figure 3.9, we provide a 3-bit structure as an example. If the first digit of the prefix is 1, the ID value is on the left side of the tree. If the first digit of the prefix is 0, the ID value is on the right side of the tree. If the first two digits of the prefix are 00, we can further narrow the search range in the tree structure. We need to notice that this tree structure does not represent the actual connection of nodes. The binary tree can be decomposed for any node into a series of continuous subtrees. From the point of view of a node, the split rule is to start from the root level, split the subtree that does not contain the node, and then split the next subtree that does not contain this node, and so on until finally, only this node is left. After each node splits the subtree according to its perspective, N subtrees can be obtained, and then N routing tables need to be maintained by the node. For example, in Figure 3.10, from the root level, the first subtree without node 001 is with the prefix 1xx (including 111, 110, 101, 100). Next, find the subtree that does not contain node 001 from the other half of the tree. The next subtree is 01x (including 011, 010). The final subtree without node 001 is node 000. Each node knows its neighbourhood well and has contact with a few nodes far away, which can help locate destinations that are far away.

The Kademlia routing table is constructed using lists called K-buckets, which are shown in Figure 3.10. Different k-buckets store node information with different distances from the node. For example, in Figure 3.10, a node's k-bucket[0] stores the node information whose distance is 1 from this node. K-bucket[1] stores the node information whose distance is 2 and 3 from this node.

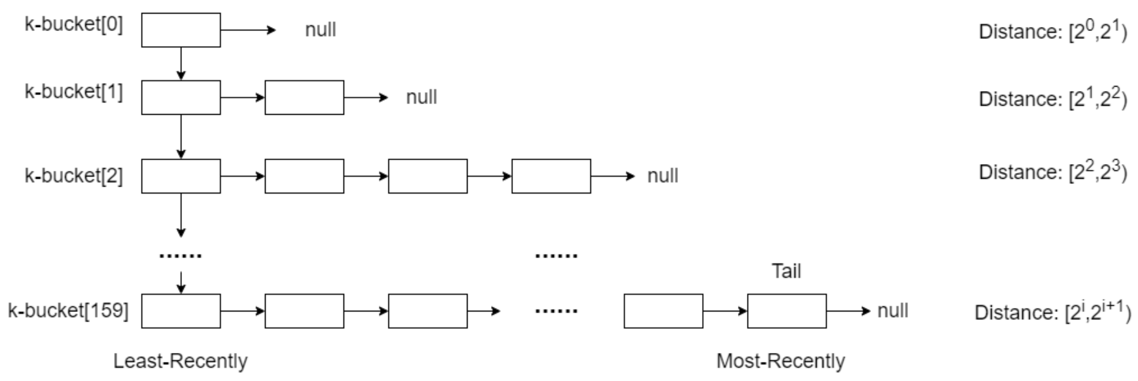


Figure 3.10: Kademlia's Routing Table - KBucket Structure

Table 3.5: Distance Range of each K-Bucket

K-Bucket Number	**Distance Interval**	**Distance Range**
0	$[2^0 - 2^1)$	1
1	$[2^1 - 2^2)$	2-3
2	$[2^2 - 2^3)$	4-7
3	$[2^3 - 2^4)$	8-15
...	...	...

The XOR operation is used to calculate the logical distance between two nodes.

$$d(x, y) = x \oplus y$$

The result is 0 when the corresponding bits are the same and 1 when they are not. The distance range of K-bucket[i] is $[2^i, 2^(i + 1))$. If a node with ID A wants to find a node with an ID value of B, Kademlia performs a route search according to the following recursive operation steps:

(1) Calculate the distance to B, $d(A, B) = A \oplus B$. and see that the result belongs to a particular distance range. We then choose the K-bucket corresponding to this range.

(2) Filter out α nodes closest to the target ID from the corresponding K-bucket and initiate a $FIND\_NODE$ query request to each of these nodes in parallel. If the number of nodes in this K-bucket is less than α, select a total of α nodes with the closest distance to B from multiple nearby buckets.

(3) For each node that receives the query operation, if it finds that it is B, it will answer that it is the closest to B; otherwise, it measures the distance between itself and B, and select α nodes from its corresponding K-bucket.

(4) A executes the $FIND\_NODE$ operation again for each newly received node that has not already been sent the request, and this process is repeated continuously until a node closest to B is found.

(5) Through the above search operation, A obtains replies from the α nodes closest to B.

During the query process, nodes that do not respond in time are eliminated immediately, and the querier must ensure that the α nodes finally obtained are all online.

Here is an example from Figure 3.9. The current node is 001, and the target node is 101. The routing table of 001 is

$$001 \oplus 101 = 100$$

Table 3.6: 001 Routing Table

001	k-bucket[0]	000			
	k-bucket[1]	010	011		
	k-bucket[2]	110	100	111	

the XOR distance between 001 and 101 is 100 $= 2^2 = 4$, which is in [4,8) distance range of K-bucket[2]. Look up 101 in that k-bucket, not found. Next, we send requests to 110, 100, and 111. Take 110 as an example. The following is the routing table of 110.

Table 3.7: 110 Routing Table

110	k-bucket[0]	111			
	k-bucket[1]	100			
	k-bucket[2]	011	010	001	000

Calculate $110 \oplus 101$, the XOR distance between 110 and 101 is 011=3, which belongs to K-bucket[1]. Look up 101 in that k-bucket, but not found. Next, send a request to 100,

Table 3.8: 100 Routing Table

110	k-bucket[0]	101			
	k-bucket[1]	111	110		
	k-bucket[2]	001	100	010	011

Calculate $100 \oplus 101$, the XOR distance between 100 and 101 is 011=1, which belongs to K-bucket[0]. Look up 101, it is found in the k-bucket.

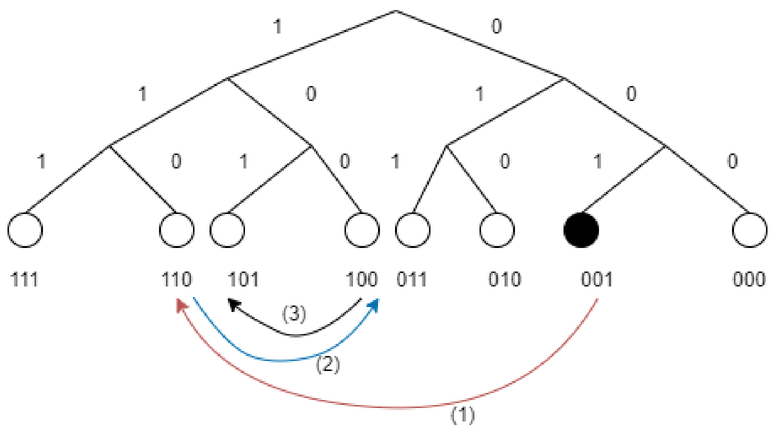


Figure 3.11: Nodes Visited During Kademlia Routing Algorithm (001 to 101)

In the Kademlia algorithm, each node maintains the routing/address of other nodes so that when any node in the network changes, including joins or leaves, the impact on the entire network is minimal. Every time α nodes are taken out of the same k-bucket, a query request is issued. The requesting node will always get the response from the fastest peer. For security reasons, the longer a node is online, the more likely it will be added to a k-bucket. Therefore, even if an attacker constructs a batch of malicious nodes, it is challenging for these malicious nodes to be added to a k-bucket by normal nodes.

Flare [82] uses the key ideas of the Kademlia Distributed Hash Table for finding routes in payment channel networks. Flare is a hybrid routing algorithm which consists of two stages 1) proactive and 2) reactive stages. A node proactively discovers the neighbours and beacon nodes. The reactive stage is for selecting and ranking candidate routes when it receives paths from other nodes. Flare is similar to Kademlia in constructing the routing table and route discovery. It uses 256 bits address instead of 160 bits address in Kademlia. It uses an XOR operation to calculate the distance between two addresses to determine who is the next hop. Flare simulated a 2000 nodes' topology and found this routing algorithm has the potential to support a scale of network with millions of users.

3.3.4 AODV-Based Routing

AODV-based routing algorithm was proposed for use in the lightning network in 2018 [83]. It applies the Ad-hoc On-Demand Distance Vector Routing protocol (AODV) to blockchain payment channel networks. The node does not store the routing information of all nodes in the network. It checks the routing table only when it needs to generate payments to the destination node. If there is no route, it broadcasts REQ requests to neighboring nodes with direct payment channels to discover a path. If the node receiving the REQ message is the destination node or there is a route to the destination node in the routing table, the REP message will be unicast to the source node and stop broadcasting the corresponding REQ message. If the node receiving the REQ message is not the destination node or if there is an intermediate node routing to the destination node, the REQ message will continue to be broadcast to other direct neighboring nodes except the current node. When the source node needs to communicate with the new destination node, the REQ is repeatedly sent to discover the route to the new destination node. Figure 3.12 shows the route discovery and route selection process.

Both REQ and REP messages contain a sequence number field. The sequence number allows the node to distinguish between old and new messages. It allows the node to update

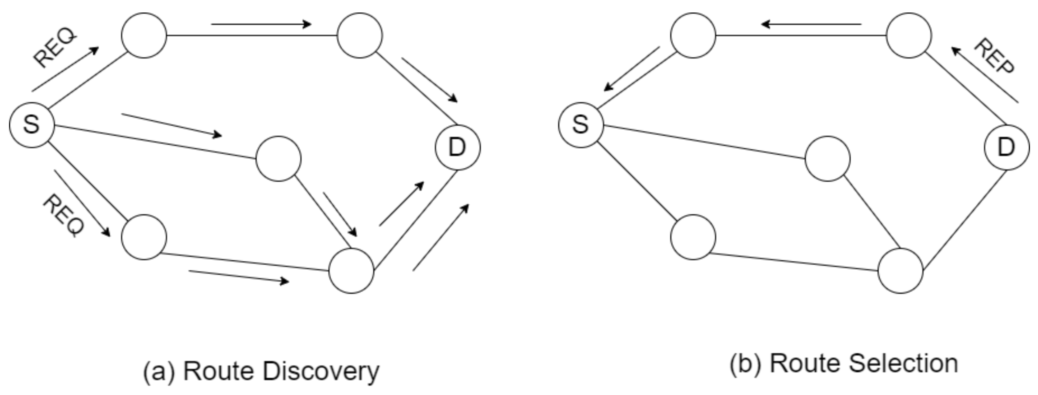


Figure 3.12: AODV-based Routing Algorithm

the old information in the routing table with the new information.

The AODV-based routing protocol is an on-demand routing protocol that only operates when the source node needs it. Nodes only need to store the required routing data, which reduces memory requirements. However, the route discovery process needs to broadcast messages that consume considerable network resources. At the same time, broadcast messages will incur delays which cause routes to become outdated. This routing algorithm is suitable for networks whose topology is changing rapidly but expensive for stable ones.

The routing algorithms above are all single-path routing algorithms. Multi-path routing algorithms use the splittable characteristics of off-chain transactions to split an off-chain transaction into multiple small transactions to reach the destination through different payment channels. Multi-path routing has high concurrency when the network status changes dynamically.

3.3.5 SilentWhispers

SilentWhispers is a Landmark-based routing algorithm proposed by Giulio Malavolta et al. in 2016 [84]. In the Landmark algorithm, a landmark maintains a tree structure with itself as the root node. Each edge of the tree is a payment channel. Each node is eligible to be a landmark node, and each node has the right to choose which landmark nodes to assist it in the routing process. In the SilentWhispers algorithm, a highly connected node in the network will be selected as a landmark. The path from the sender to a landmark and the path from the landmark to the receiver will be combined together to generate a complete payment path.

There is only one landmark in SilentWhispers. It means that only one search tree is generated. In this case, SilentWhispers acts as a single-path algorithm. A transaction goes from the source node to the destination node via only one path.

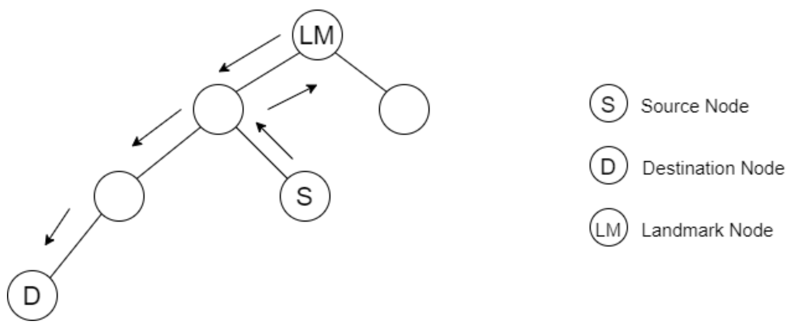


Figure 3.13: SilentWhispers Routing Algorithm with One Landmark Node

In the SilentWhispers algorithm, the landmark node is fixed. The information on these landmark nodes is shared across the entire network. In the network, each node only maintains its next hop to all landmark nodes. When receiving a path discovery request, a node will judge the available balance of its payment channels and decide where to forward the message based on the local routing table. SilentWhispers uses digital signatures to construct a digital signature chain to achieve node identity authentication.

SilentWhispers used real transaction sets to simulate the network and analyze the result. A transaction takes roughly 1.3 seconds using the SilentWhispers routing algorithm, which is less than Ripple network's roughly 5 seconds per transaction [84]. It has the potential to scale to an extensive network.

The fact that there is no public ledger of the entire network in SilentWhispers reduces the storage requirements. However, the signature exchange significantly increases the computing and communication overhead. In addition, in each tree structure, all paths go through the landmark node, and this causes low channel utilization and resource wastage. Even the route between two nodes very close to each other must pass through the Landmark node. This algorithm periodically updates routing tables, which may cause payment failure due to out-of-date channel information.

3.3.6 SpeedyMurmurs

SpeedyMurmurs [85] proposed in 2017, solves some of the problems of SilentWhispers, but has not gone into production. SpeedyMurmurs adopted the tree-based graph embedding routing algorithm to realize path discovery among nodes. It can split a transaction amount into small amounts and send these small transactions through multiple landmark trees. Each landmark maintains a k-tree as the root node based on its local payment channels information. Assume that a payment from u to v needs to be processed, the amount is c, there are m Landmark nodes $(L_1, L_2, \ldots, L_m)$. The sender divides the payment amount into m parts $(c_1, c_2, \ldots, c_m)$. The routing of c_i is handed over to L_i.

There are two main differences from SilentWhispers. The first is that multiple landmarks are used in SpeedyMurmurs. Multiple paths in different spanning trees provide routing services. The second one is that each node will set a “coordinate”. This coordinate contains its parent’s coordinate as a prefix and its own position in the spanning tree. Nodes on the candidate path anonymously exchange information about neighbors. Therefore, if a node knows a path closer to the recipient, it will forward the payment in this direction, called the ”fast path”. In the fast path, the intermediate node does not necessarily know the recipient but knows the neighbors near the recipient. Figure 3.14 shows the coordinates setup process of the SpeedyMurmurs routing algorithm.

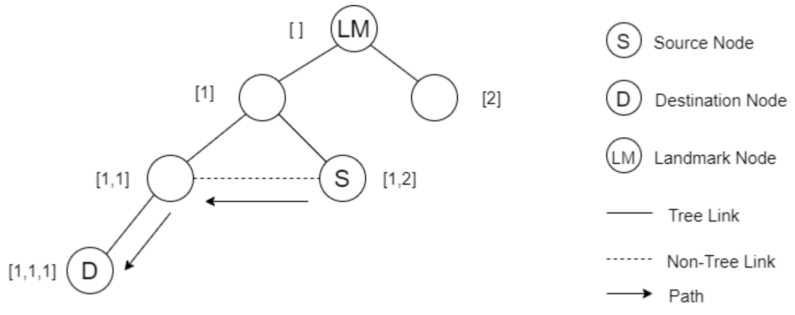


Figure 3.14: Coordinates Setup for SpeedyMurmurs

The landmark node is assigned an empty coordinate value. The child node’s coordinate value is the combination of its parent node’s coordinate and the identifier of this child node. For example, in Figure 3.14, the closest node to the destination D [1,1,1] is the node with coordinates [1,1]. The closest node to the node [1,1] is the source node with coordinates [1,2]. The path is from the source node S to the node with the coordinate [1,1] to the destination D.

The source node randomly splits the payment amount. In the search process, the node needs to determine whether the balance of the relevant channel is sufficient. If part of the path discovery fails, the source node will split the payment amount again according to the failure information until all paths meet the balance requirement. If the shortest path can be found through the coordinates, the route does not need to pass through the landmark node.

The adoption of coordinates effectively shortens the path length and improves channel utilization. In SpeedyMurmurs, the node coordinates update only occurs when a channel’s balance becomes extremely low or zero. Compared with periodic updates, the overhead of this algorithm is lower. However, when the balance of the landmark node’s child node becomes zero, the tree needs to be reconstructed, which creates additional overhead. Compared with SilentWhispers, SpeedyMurmurs performs better on landmark and routing selection under both static and normal dynamic networks. However, when the network changes frequently, SilentWhispers provides better performance than SpeedyMurmurs.

3.3.7 Flash

Flash [86], CoinExpress [87] and Concurrent and Distributed Route Selection [88] are flow-based routing algorithms for payment channel networks. We take Flash as an example. Flash proposed by P.Wang, et al. in 2019. It divides transactions into two categories according to the payment amount. Nodes have the network topology without capacity information, and each node has its routing table. The node calculates the shortest path for transactions with small payment amounts according to its routing table. Then the node directly pays along this path. If it succeeds, the payment ends. If it fails, the transaction amount that this path can support will be calculated, and the nodes can pay part of the transaction amount along this path. The node recalculates the available path for the remaining transaction amount and keeps trying according to the above steps until the transaction is completed. If all feasible paths are exhausted, and the transaction demand cannot be met, the transaction fails. In the worst case, the time cost is very high. It is hard for most nodes to meet the balance requirement for transactions with large payment amounts. It will cause channel congestion and transaction delay due to too many attempts, affecting the network's overall utilisation. Flash uses a breadth-first algorithm to find several paths and calculates the maximum payment amount that these paths can support. The source node also compares transaction fees required by different paths to find low-cost and high-efficiency paths. However, nodes need to maintain the routing table of the whole network. When the number of users increases significantly, the storage requirements for nodes will become very high. The team who proposed Flash provided a simulation [89], but this routing algorithm has not been implemented in the lightning network mainnet.

3.3.8 Spider

Spider [90] is a packet-forwarding-based routing algorithm which was proposed in 2018. It splits transactions and forwards them in parallel through multiple paths. In Spider, spider routes are used to forward transactions to the receiver. When the channel is blocked, or the balance of the channel is insufficient, the corresponding transactions will be placed in a spider router waiting queue. At this time, the router can increase the transaction fee in the blocking direction of the channel locally to avoid more transactions passing through this direction. Through such a mechanism, nodes are encouraged to explore different routes, so the channel network is rebalanced. When the channel recovers, the waiting transactions are processed according to their sequence in the queue. Figure 3.15 shows an example where Spider is facing channel congestion.

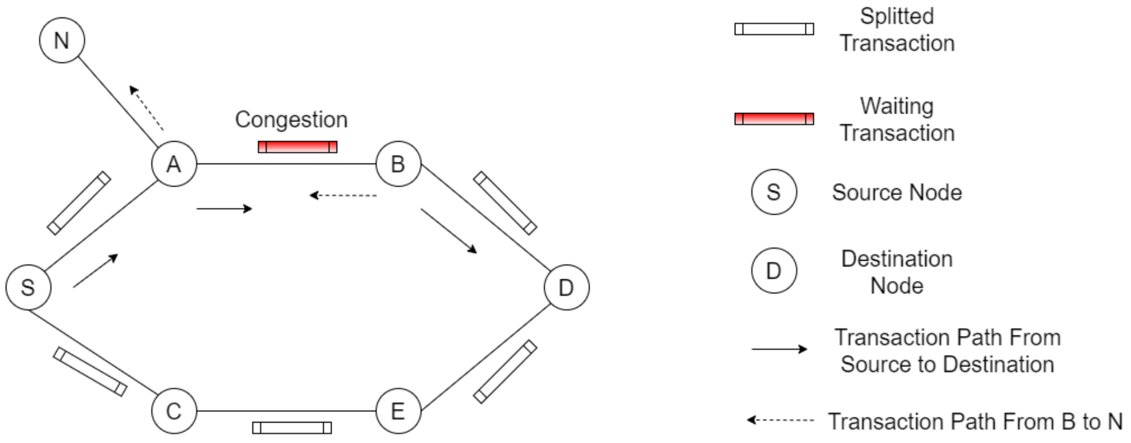


Figure 3.15: Spider Routing Algorithm

In Figure 3.15, S is the source node, D is the destination node, and A,B,C,E are spider nodes. The transaction from Source Node (S) to Destination Node (D) is split to pass through two paths, 1) $S \to A \to B \to D$, 2) $S \to C \to E \to D$. Because the $A \to B$ direction of the channel is frequently used, the balance in this direction is exhausted and most funds are locked in the $B \to A$ direction which causes this channel congestion. The payment through this path waits in the queue. Router A then sets a high transaction fee for $A \to B$ direction, and router B sets a low transaction fee for $B \to A$ direction. After that, other nodes are more likely to send transactions from B to A instead of A to B because of the low cost. It will cause funds to move back to the A side. The channel will be rebalanced. If the funds are enough in $A \to B$ direction, this transaction $S \to A \to B \to D$ will be sent instantly.

Spider improves the utilization of the channel by dynamically adjusting the routing fee and reduces the cost of closing and reopening the channel. However, the nodes need to maintain information on the entire network and control the fee level at nodes which increases the storage requirements of nodes and reduces the scalability of the network. In addition, the state of blocked transactions depends on whether the channel is rebalanced, increasing transaction latency.

3.3.9 Comparison of Routing Algorithms

Table 3.9: Comparison of Routing Algorithm Under Different Evaluation Criteria

Evaluation Criteria	**Dijkstra**	**Ant**	**Kademlia**	**AODV-based Routing**	**Speedy Mur-murs**	**Silent Whisph ers**	**Flash**	**Spider**
Efficiency	Low	High	High	Low	High	High	High/Low	Low
Rebalancing	×	×	×	×	×	×	×	✓
Routing Table Size	High	Low	Low	Low	Low	Low	High	Low
Path Opti-mality	✓	×	✓	✓	✓	×	✓	✓
Concurrency	×	×	×	×	✓	✓	×/✓	✓
Latency	Small payment amount: low Large payment amount: high				Small payment amount: high Large payment amount: low			
Small Transac-tion Cost	Low	Low	Low	Low	High	High	High	High

Efficiency: Since the e-commerce network has the characteristics of 1) a large number of participants, 2) high transaction frequency, and 3) scattered transaction locations, a routing algorithm that can reduce unnecessary searches in order to improve the efficiency of pathfinding is a vital evaluation criterion. Dijkstra's algorithm is strictly a breadth-first algorithm, which calculates the cost from all nodes to the starting node. It is not an efficient choice for a large-scale network. There is no need for the Ant algorithm to search all nodes to find a path which is much better than the Dijkstra algorithm. However, it tends to choose the next node randomly, which affects the search efficiency. The Kademlia algorithm selects a limited number of closest nodes as a next step. reducing the number of searches. In AODV-based routing, the route discovery process requires broadcast messages which consumes a lot of network resources and has low efficiency. SpeedyMurmurs and SilentWhisphers use landmark nodes to reduce the number of searches. In Flash, the pathfinding process for small value payments is more efficient than that for large payment amounts. The transaction goes through multiple paths, and this increases uncertainty just like the Spider routing algorithm.

Rebalancing: Frequent transactions will cause imbalanced channels. So, whether the routing algorithm has a rebalancing mechanism affects the stability and efficiency of the channel network. Only the Spider routing algorithm considers rebalancing mechanisms.

Routing Table Size: This indicates whether a node has a global view of the network Nodes in Dijkstra and Flash algorithms need to store global information to calculate a path. For Ant, Kademlia/Flare, AODV-based routing, SpeedyMurmurs, SilentWhisphers, and Spider algorithms, nodes only need to maintain limited information, including neighbor' information and direct connected channels information. The routing table size and storage requirements are consequently reduced.

Path Optimality: It shows whether the routing algorithm considers taking the optimal path or first come first adopted. Dijkstra is a shortest path first algorithm. In an e-commerce topology, the optimal solution for the shortest path can be obtained. The Ant algorithm seeks a satisfactory solution rather than an optimal solution. When the Kademlia algorithm locates a node, it requests the k nodes closest to search in each k-bucket. The result returned at each step is the local optimal solution. AODV-based routing algorithm can find an optimal path based on the existing routing table. SilentWhisphers finds a path through a landmark. There is a high probability that this path is not optimal. SpeedyMurmurs can find a shorter path based on the neighbors routing table. Flash compares all available paths to find multiple lowest cost paths. Spider uses rebalance mechanism to ensure the optimal path decision.

Concurrency: Single-route algorithms, including Dijkstra, Ant, Kademlia/Flare, AODV-based routing, and Flash for small payment amounts, have low channel utilization. Multi-route algorithms, including SpeedyMurmurs, SilentWhisphers, Flash for the large payment amount, and Spider, can split a transaction to multiple channels, increasing channel utilisation and having high concurrency.

Latency: Transaction processing latency. In a scenario where the payment amount of single-route algorithms is small, and the channel capacity is sufficient, the transaction latency is significantly lower than that of the multi-route algorithms. But if there are large transactions, the single-route algorithm channels may become congested for a long time, resulting in increased transaction processing time.

Small Transactions Cost: The average transaction fee required to transfer a small payment from the source node to the destination node should be considered. A transaction is going through a single path under single-route algorithms. However, multiple paths are used to send a transaction in multi-route algorithms. With more intermediate nodes involved, the total cost will increase. Therefore, with the same topology, single-route algorithms have lower costs than multi-route algorithms.

After the above comparison, the Kademlia algorithm has high efficiency and low routing table size. It can find the optimal path during the path discovery process. The Kademlia algorithm has low latency and low costs for small transaction amounts. Using these criteria, the Kademlia routing algorithm is a good choice for our e-commerce network.

3.4 Channel Rebalancing

As we mentioned, when two participants do not have a direct payment channel, they need to establish a payment path by connecting intermediate nodes. Most routing algorithms do not consider the rebalancing problem. Frequent transactions in one direction often result in an imbalanced state of the channel. In an extreme case, the balance on one side of the channel becomes 0. A bidirectional channel becomes a unidirectional channel.

Figure 3.16 shows the imbalance process of payment channels among node A, B, and C. The original capacity of the channel between A and B is 200 and between B and C is 200. 100 for each side. When many transactions go through $A \rightarrow B \rightarrow C$, the A-side balance of channel AB and B-side balance of channel BC are quickly exhausted. At this time, transactions can no longer pass via $A \rightarrow B \rightarrow C$. Path discovery attempts will fail if this is the only path from A to C. This situation is an extreme case of channel imbalance.

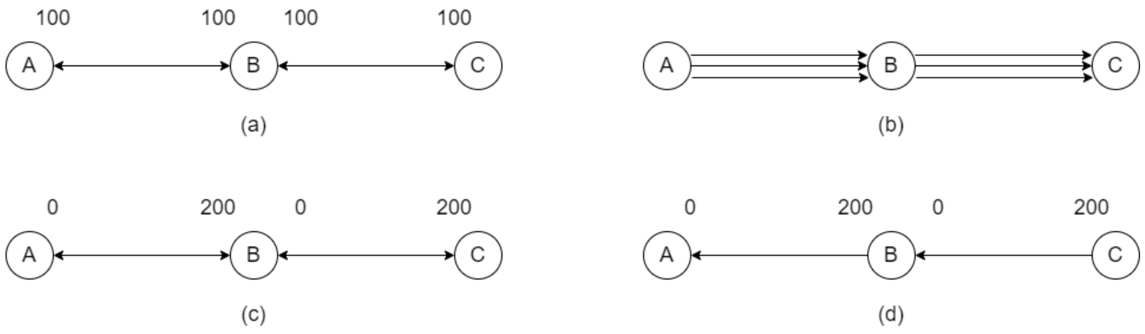


Figure 3.16: Imbalance Process of Channels

The general solution for this situation is to close the channel and then reopen it and deposit more funds to the channel. However, for a scenario like e-commerce, this operation will generate a considerable number of on-chain transactions which cannot be processed quickly. Rebalancing is a technology that attempts to deal with this issue.

The purpose of rebalancing is to enhance the liquidity of the network while reducing interaction with the blockchain. There are several methods to achieve channel rebalance.

We divide the existing rebalancing technologies into four categories: 1) Leader-Decision, 2) BackPressure, 3) Cost-Incentives, and 4) Self-Organizing.

3.4.1 Leader-Decision

The Liquidity Network proposed the Revive protocol that allows payment channels to be rebalanced off-chain. The network adopted by Revive needs to include a ring structure. In Revive, a leader will be chosen to guide the rebalancing process. During this period, all participating channels need to be frozen. A series of off-chain payment transactions are constructed based on each participant's preferences and current needs, then handed over to everyone for signature. After each participant has signed, the atomic transaction is forced to be executed.

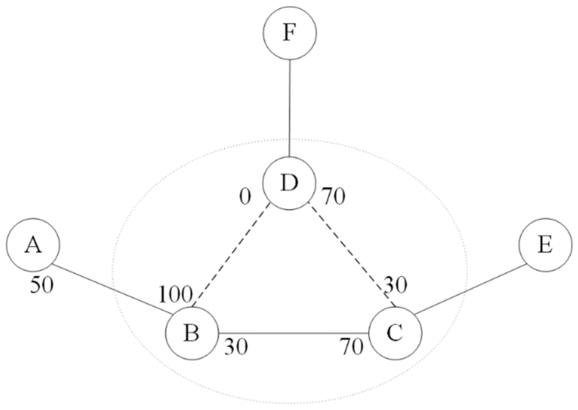


Figure 3.17: Imbalance Network with Ring Structure

In the figure, A wants to send 50 to E. The original payment channel from A to E is $A \to B \to C \to E$, but at this point, the deposit on the channel from B to C is insufficient to allow this payment to be made. Therefore, the network finds another path $A \to B \to D \to C \to E$ for the transaction. However, this path is longer than the original one and will cost more because it involves more intermediaries. Revive, in this case, could rebalance the deposits in each channel and make the shorter path available for participants.

Figure 3.17 shows the rebalancing process between participants in Revive. Before the rebalancing, a leader needs to be elected by all participants to receive the rebalancing requests from enough participants, and calculate the amount of deposit needed to be rebalanced in each channel. Revive uses a leadership rotation strategy. Each participant is identified by its public address in the global ledger, which is considered unique and numeric. For the first round, the participant with the smallest identifier will be elected as a leader. The participant with the next smallest identifier will be elected as the leader for

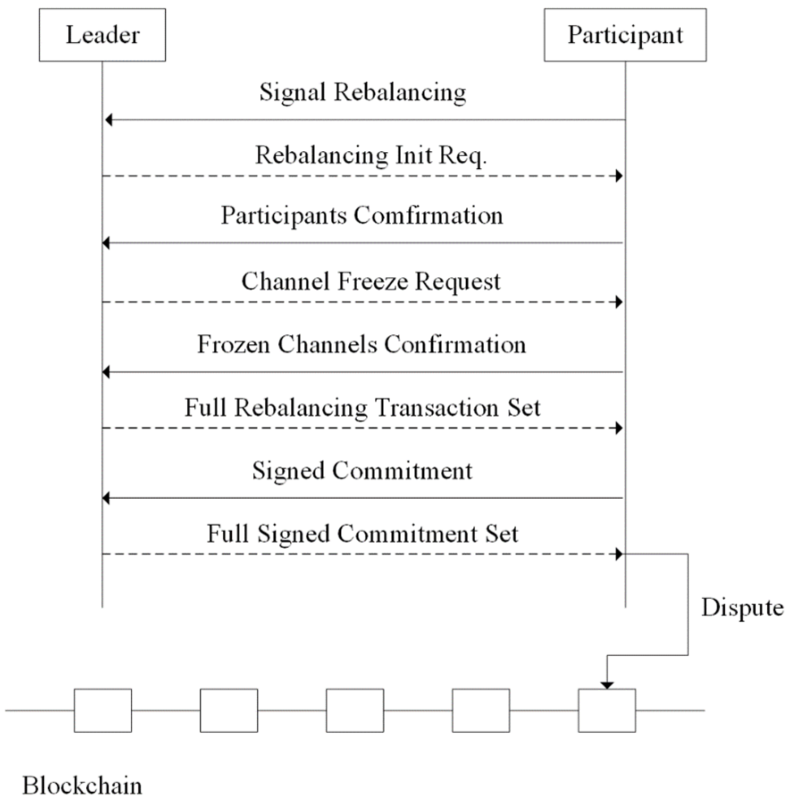


Figure 3.18: Rebalancing Process in Revive

the next round. The process is described in the following.

(1) Participants send rebalancing requests to the leader in this round. A threshold defines the number of rebalancing requests that a leader needs to receive from participants to initiate the rebalancing.

(2) The leader sends a rebalancing initiation request to all participants to ask for their confirmation after he receives enough rebalancing requests in (1).

(3) Participants respond to the leader with a Participant Confirmation. Then the leader prepares a list of participants who will join the rebalancing in this round.

(4) The leader sends Channel Freeze Requests to the participants involved to let them freeze their funds in the channel.

(5) Participants send a Frozen Channels Confirmation indicating which channels they have frozen and the balance they expect after the rebalancing.

(6) The leader calculates a set of transactions that indicates the balances of each participant and their balances for each channel after the rebalancing. The Merkle root hash of these transactions and their addresses will be sent to participants for verification and signing.

(7) Participants verify the hashes to detect whether they are correct. If so, they send the

hash with their signatures to the leader.
(8) The leader unfreezes channels by multicasting the states to participants after receiving all their signatures. The balance of channels has been updated.

If the leader does not collect enough signatures in (8), the Full Signed Commitment Set will not be generated. After a time out, the balances revert to the original amounts before this process begins.

3.4.2 BackPressure

The Celer Network proposed a Distributed Balanced Routing (DBR), which originated from the backpressure routing algorithm in wireless networks [42]. DBR calculates routing paths based on network congestion to avoid channel imbalance. It proposed a measurement standard, congestion-plus-imbalance (CPI) weight. The CPI calculates the difference between the inflow and outflow tokens of a node in a period of time. The greater the difference, the easier the node will be selected as the next hop. Because the more significant difference proves that channels of this node are more imbalanced, through transactions, the token flows in the reverse direction and channels are restored to balance.

DBR does not require additional scheduling nodes and can operate in a decentralized manner. It detects non-responding nodes and makes adjustments in each calculation. However, like the traditional backpressure algorithms, DBR needs to maintain the queue at each node. For large networks, this overhead is enormous. Moreover, DBR is a dynamic routing algorithm. It may choose a longer path according to the network congestion degree, which causes unnecessary delay.

3.4.3 Cost-Incentives

As we mentioned, the Spider routing algorithm uses a cost incentive rebalancing technology. In a channel, the node will increase the transaction fee on the side where the token often flows out. Through the incentive of lower transaction costs, tokens get transferred in the opposite direction, and the channel is rebalanced. There is no centralized scheduling node in the network. However, as we described before, each node needs to maintain information on the entire network, which increases the node's storage requirements and reduces the network's scalability.

3.4.4 Self-Organizing

Rene Pickhardt proposed Just-in-Time (JIT) routing [91] in 2019. In multi-hop routing, the pathfinding may fail due to insufficient funds on a certain channel. JIT attempts to suspend the routing process and transfer funds from other channels to this channel to rebalance channels that lack liquidity. If successful, payment can be sent; otherwise, temporary_channel_failure will be sent. Figure 3.19 gives a JIT rebalance example. In

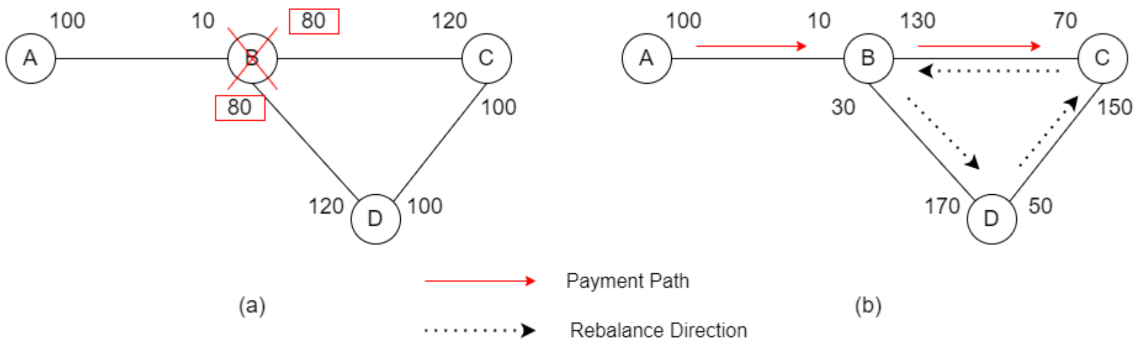


Figure 3.19: JIT Rebalance Process

Figure 3.19, A tries to transfer 90 to C. There are two paths $A \to B \to C$ and $A \to B \to D \to C$. However, the balance on B side is not enough to support this transaction. In the traditional path discovery process, A will receive a temporary_channel_failure feedback because there is no path found. JIT will move funds from B's neighbor to B in order to increase the balance on B side. In the figure, 50 moves from C to B, then B to D, and finally D to C. After this step, the total funds at each node will not change. But now B has sufficient balance to support the transaction via $A \to B \to C$.

JIT routing improves the routing success rate to a certain extent. At the same time, nodes only need to know their own channel information instead of that of the whole network, which protects privacy. However, if the node does not know the channel information of other nodes, it may encounter failure when transferring funds. The worst case is that it will fail anyway but will still take a long time to return the failure message to the initial node. This causes a higher latency.

3.4.5 Comparison of Rebalancing Methods

Table 3.10: Comparison of Routing Algorithm Under Different Evaluation Criteria

Evaluation Criteria	**Leader-Decision**	**BackPressure**	**Cost-Incentives**	**Self-Organizing**
Active or Passive	Active	Active	Passive	Active
Frozen Channels	✓	×	×	×
Ring Structure	✓	×	×	✓
Intermediary Transaction Fees	×	×	✓	×

Active or Passive: Active rebalancing is initiated by the node to rebalance the channel balance. Passive rebalancing is using incentives to encourage funds to move in the opposite direction along an imbalanced channel to achieve channel rebalance. For active rebalancing, there may be two reasons for failure: 1) channels in the network do not have enough balance, 2) there is no candidate channel for rebalancing in the network, for example, no ring structure is formed among channels. One consequence of passive rebalancing is that some payments have to pay higher fees than in the absence of passive rebalancing, resulting in insufficient capacity in the channel to transfer the payment amount plus increased fees.

Frozen Channels: Freezing channels means there is no liquidity during the rebalancing decision period. For leader-decision rebalancing, a leader is selected to manage the rebalancing process. For example, in Revive, the leader must confirm there is no state change in corresponding channels before the final decision. This method can ensure the accuracy of rebalancing. However, the partial unavailability of the payment channel network for a certain period will affect the transaction efficiency of the entire network.

Ring Structure: Leader-decision and Self-organization rebalancing need a ring structure to support the rebalancing movement. This structure can ensure that the channel is rebalanced without changing the balance of nodes. However, the status of the intermediate nodes in the path will affect the success rate of rebalancing, such as whether the balance is sufficient or whether the node is reachable.

Intermediary Transaction Fees: Cost-Incentives rebalancing uses different transaction fees to motivate the rebalancing process. This means the sender needs to transfer the payment amount plus transaction fees which increases the cost of transactions. Other rebalancing methods can set transaction fee to 0.

For the e-commerce industry, rebalancing goals are 1) less total fee costs, 2) higher rebalancing success rate, and 3) smaller impact on the whole network. So the best choice for our application is a low-transaction-fee active rebalancing without the need for ring structures and without the need to freeze channels.

In this chapter, we investigated the size of the current e-commerce network. We discuss the topology and routing algorithm requirements of Layer2 based e-commerce. After summarizing the problems and feasibility of applying various topology and routing algorithms in this scenario, we find that topologies like Fat-tree structure and Kademlia routing algorithms are more suitable for Layer2 based e-commerce. Topology like Fat-Tree can shorten the path length, reduce the number of required payment channels, and improve the extendibility and reliability of the network. Routing algorithms, like the Kademlia algorithm can improve pathfinding efficiency, reduce the size of routing tables, provide optimal paths, and reduce transaction fees.

In the next chapter, we will propose a blockchain Layer2 based e-commerce architecture, detailing the design of topology and routing algorithm.

Chapter 4

The Blockchain Layer2 based E-Commerce Architecture

In this chapter, we will introduce our proposed blockchain layer2 based design for e-commerce. In our design, the base layer is the blockchain network used to settle transactions. The second layer is the lightning network which can help to improve the number of transactions per second and reduce the transaction fees required. The participants in the lightning network are customers, merchants, and platform nodes maintained by an e-commerce platform like Amazon.

Each participant runs a lightning node. Each customer or merchant generates one or more payment channels to the platform nodes. Platform nodes establish payment channels with each other according to our proposed topology. These participants and payment channels form the e-commerce lightning network. Above the lightning network layer, there is a routing layer used to find a path from a customer to a merchant. We improved the Kademlia routing algorithm and applied it to this situation. After a participant joins the lightning network, our algorithm will send a request to a centralized process that we call the k_id allocator to be allocated a k_id. These k_ids organize the nodes into a Kademlia tree structure. The routing layer will use our improved Kademlia routing algorithm, k-routing, to find a path. This route will be returned to the lightning network layer, and a transaction will be sent along the returned path using source routing. Customers can generate many transactions on the lightning network layer if they have enough balance. When a customer or merchant wants to close the channel, a settlement transaction will take place on the blockchain, and the balance will be returned to their blockchain account. The design details will be described in the following sections.

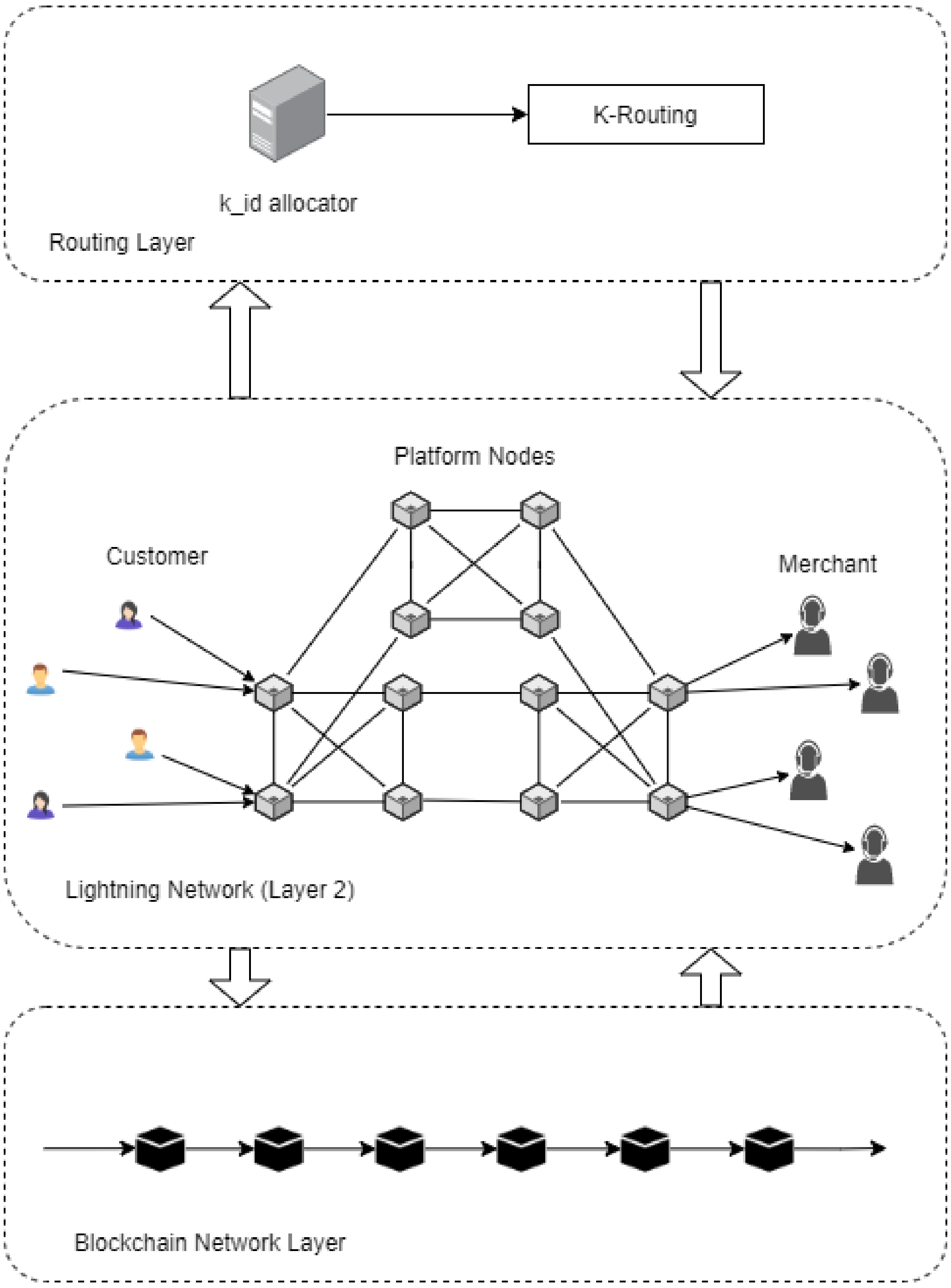


Figure 4.1: Proposed Layer2 based E-Commerce Design Architecture

4.1 Network Topology

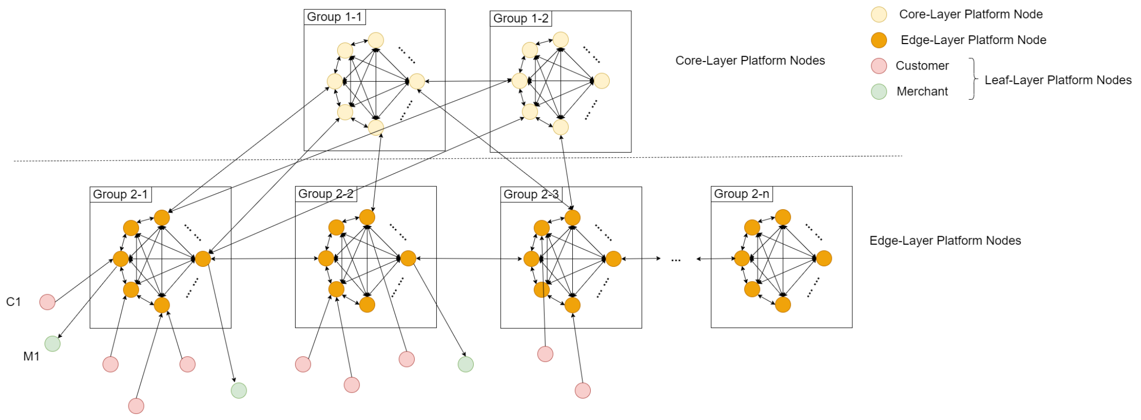


Figure 4.2: Layer2 based E-Commerce Topology

Section 3.2.5 shows how a Fat Tree topology can be horizontally scaled to support more servers while not increasing the path length. This is potentially suitable for a layer2 based e-commerce system. If the Fat-Tree topology is applied to the e-commerce system, switches are replaced by platform nodes and servers connected to the switches are replaced by the customer and merchant nodes which connect with platform nodes. Figure 4.2 shows our layer2 based e-commerce system topology.

There are two levels of platform nodes in this topology. Customers and merchants are in the Leaf-Layer, which directly connects with Edge-Layer platform nodes. All Core-Layer platform nodes are used to connect the Edge-Layer platform nodes. If there is no path via the Edge-Layer platform nodes or a path needs to pass too many intermediaries, a customer can find a route via Core-Layer platform nodes to quickly send transactions to a specific merchant.

Platform nodes are divided into groups. There are two groups in the Core-Layer, but they connect with the same Edge-Layer platform nodes. The second group of Core-Layer is used for an alternative path selection if the node in another group is unresponsive. Each Core-Layer group has several channels connected with its neighbour groups. These connections can provide more path choices. Moreover, it can relieve the traffic pressure when the transaction volume becomes enormous. Groups in the same level have the same number of platform nodes inside, and platform nodes are mesh connected inside a group. The advantage of mesh connection is that each platform node on the same level can directly communicate, which can minimize the number of intermediate nodes required to effect the transaction. We assume the number of platform nodes in each Core-Layer group is **G1**, and the number of platform nodes in each Edge-Layer group is **G2**. An e-commerce platform can decide the exact number of **G1** and **G2** according to the number of customers and merchants. As the number of users increases, including customers and merchants, more Edge-Layer groups can be added to help the system scale horizontally. Each Edge-Layer group has several channels connected with its neighbour groups. These connections provide more path selection and can avoid system unavailability if Core-Layer nodes become unavailable.

Even though there is no limitation on the number of channels that a node can connect with theoretically, this number may be limited in practice because of insufficient resources or bandwidth. **k** denotes the maximum number of channels that a single node can support. When the e-commerce platform, like Amazon, sets up **G1** platform nodes in each Core-Layer group and each Edge-Layer platform node has one payment channel with each Core-Layer group, the max number of Edge-Layer platform nodes that these Core-Layer nodes can support is

$$(k - (G1 - 1)) \times G1 \tag{4.1}$$

$(G1 - 1)$ is the number of payment channels that a Core-Layer platform node uses for interconnection with other Core-Layer platform nodes in the same group. Because each Edge-Layer node only connects one Core-Layer node in each group, $(k-(G1-1))$ presents the number of payment channels that one Core-Layer node can support except its inter-connection channels. In other words, it refers to the number of Edge-Layer nodes that a Core-Layer node can connect to.

Multiple Core-Layer groups are used to provide redundant channels between Core-Layer nodes and Edge-Layer nodes. Equation (4.2) shows the max number of Edge-Layer platform nodes under multiple Core-Layer groups.

$$(k - (G1 - 1)) \times G1 - b \tag{4.2}$$

b is the number of channels used to connect multiple core layer groups. The max number of Edge-Layer platform nodes decreases due to some channels of Core-Layer nodes being used to connect the other Core-Layer groups.

The number of customers and merchants that this topology can support is

$$((k - (G2 - 1)) \times G2) \times \frac{(k - (G1 - 1)) \times G1}{G2} \tag{4.3}$$

$(k - (G2 - 1)) \times G2$ is the number of customers that a group of Edge-Layer nodes can support, $((k-(G1-1)) \times G1)/G2$ is the number of Edge-Layer groups.

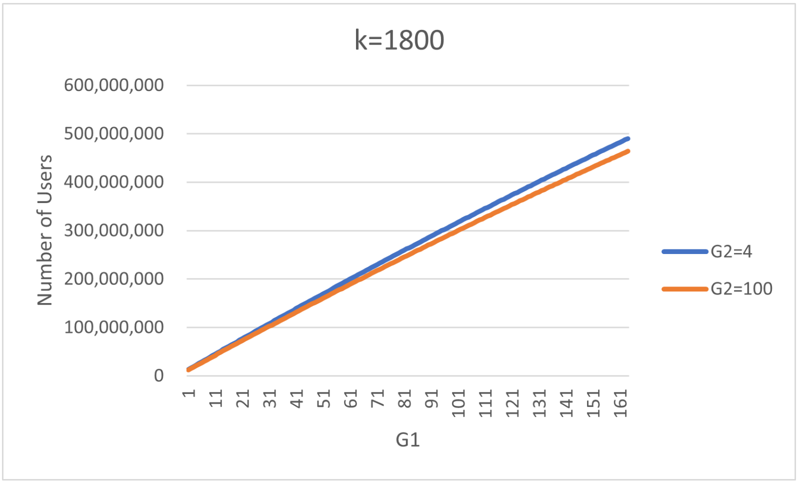


Figure 4.3: The number of Users that can be supported when k=1800

Using 1ml.com, a Lightning Network search and analysis engine we discover that in May 2021, the most connected node in the lightning network has 1,852 active payment channels. We therefore assume that the maximum number of channels that a single node can support is $k = 1800$. With the number of platform nodes in each Core-Layer group increasing, the number of customers and merchants that the platform can support is shown in Figure 4.3. We can find that the less platform nodes in each Edge-Layer group ($G2$), the more users the e-commerce platform can hold with the same Core-Layer structure.

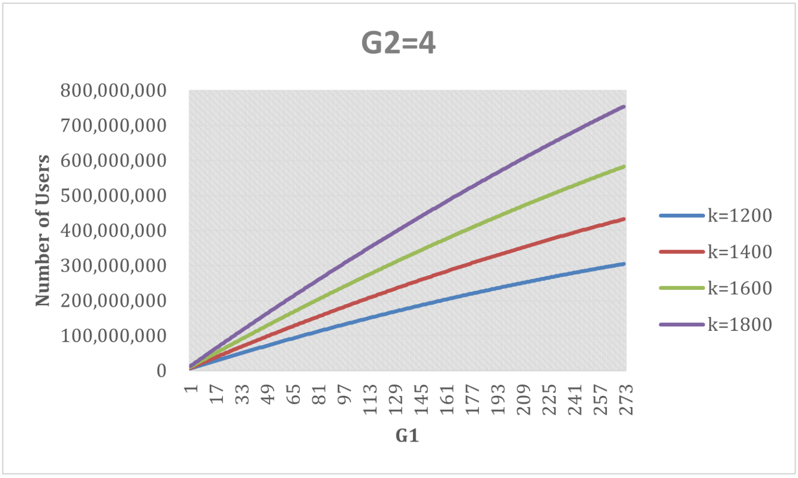


Figure 4.4: The number of Users when G2=4

If the number of platform nodes in each Edge-Layer group $G2 = 4$, Figure 4.4 shows the total number of users the system can support when the number of platform nodes in each Core-Layer group ($G1$) increases. In section 3.1, We presented information on the size of the amazon.com payment network. We will apply our network structure to that example. The number of Amazon customers is 300 million, and the number of merchants is 2.5

million, so the total number of leaf nodes should be more than $300,000,000+2,500,000 = 302,500,000$. If $k = 1200$, according to equation (4.3), when $G1 = 273$ or more, the number of leaf nodes that the network can support is more than 302,500,000, and the total number of Edge-Layer platform nodes from equation (4.1) is 253,344. If $k = 1400$, $G1$ should be no less than 177, and the number of Edge-Layer platform nodes is 216,648. If $k = 1600$, $G1$ should be 129 or more, and the number of Edge-Layer platform nodes is 189,888. If $k = 1800$, $G1$ should be no less than 99, and the number of Edge-Layer platform nodes is 168,498. Figure 4.5 shows the number of Edge-Layer platform nodes required to support the corresponding users when $G1$ increases. The e-commerce platform can add more Core-Layer groups to balance traffic, but the total number of Edge nodes will not change. The more payment channels a node can establish, the fewer platform nodes are required. This means that as node performance improves, more payment channels can be established to a single node, and the total number of nodes required decreases.

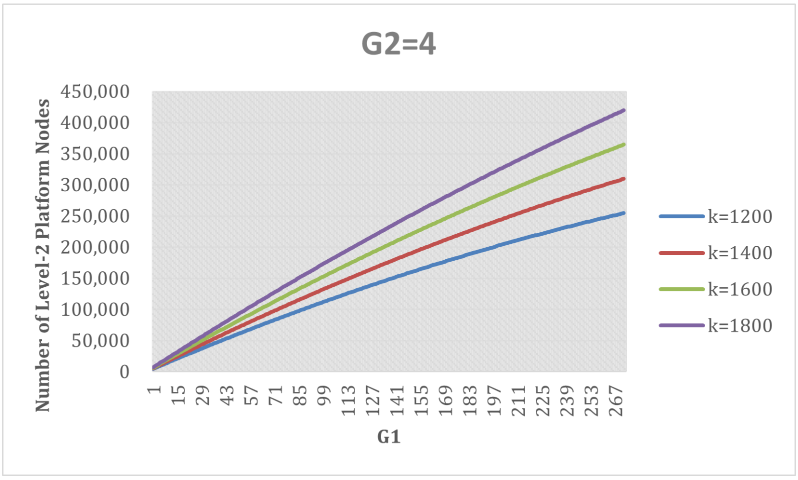


Figure 4.5: The number of Edge-Layer Platform Nodes when G2=4

Our proposed network topology is based on the Fat Tree topology. We improve the Fat Tree topology and make it more suited for Layer2 based mass e-commerce scenarios. Under the Fat Tree topology, traffic aggregates to the core layer and the core layer servers decide the path. However, in our proposed topology, Core-Layer nodes only provide another path choice if there is no path via Edge-Layer. Transactions can pass via the Edge-Layer nodes or via both Edge-Layer and Core-Layer nodes. In our topology, the e-commerce platform can scale to a larger size by adding more Core-Layer platform nodes or horizontally using more Edge-Layer platform node groups. It can support a user scale like Amazon.

4.2 K-Routing Algorithm

In section 3.3.3, we introduced the Kademlia routing algorithm developed for use in DHTs. It uses an XOR metric to measure the distance between two nodes which can significantly increase the route searching speed. However, the original goal of this algorithm is to find the $< key, value >$ rather than the nodes along the path. It updates the routing table when it finds the next closest nodes, adds these to the routing table, and then adds more nodes during the pathfinding process. We improve the existing Kademlia routing algorithm so that it can be better applied to State Channel based payment network. The Kademlia algorithm is suited for searching a network for content. Because all connections are based on IP addresses. When the new IP is added to the routing table, the source node can directly contact this IP address. However, our layer2 based e-commerce platform aims to find the next hop and ultimately find a path through the network of payment channels from the customer to a merchant.

We use the lightning network as a Layer2 overlay. Transactions can only be sent between nodes linked by payment channels. Participants must generate on-chain transactions and deposit funds to a payment channel to enable transactions. The update process of the Kademlia routing algorithm is not entirely suited for our network. Therefore, we propose a modified k-routing algorithm for e-commerce system. Figure 4.6 shows how the K-routing algorithm works. In the K-routing algorithm, only the next hops who have payment channels will be added to the routing table.

There are four types of users, 1) customers, 2) merchants, 3) Core-Layer platform nodes, and 4) Edge-Layer platform nodes. When a user joins the network, an entity called a k_id allocator assigns a unique k_id according to user's role and IP address. If the user is a customer or merchant, the allocator will also return an Edge-Layer platform node that the user node should connect with. If the user is an Edge-Layer platform node, the allocator will return a Core-Layer platform node in each Core-Layer group and other Edge-Layer platform nodes in the same group. If the user is a Core-Layer platform node, the allocator will return other Core-Layer platform nodes in a group. The k_id allocator maintains its k_id table. Once a new node joins the network, the new k_id will be updated in the table. We will describe the details of the k_id allocation mechanism in section 4.2.1.

Each participant maintains a local routing table. The routing table is a binary tree whose leaves are k-buckets. K-buckets are a list of routing addresses of other nodes in the network, which are maintained by each node and contain IP address, port, k_id, role, zone number, group number, lightning id, channel id, channel balance, and neighbor's

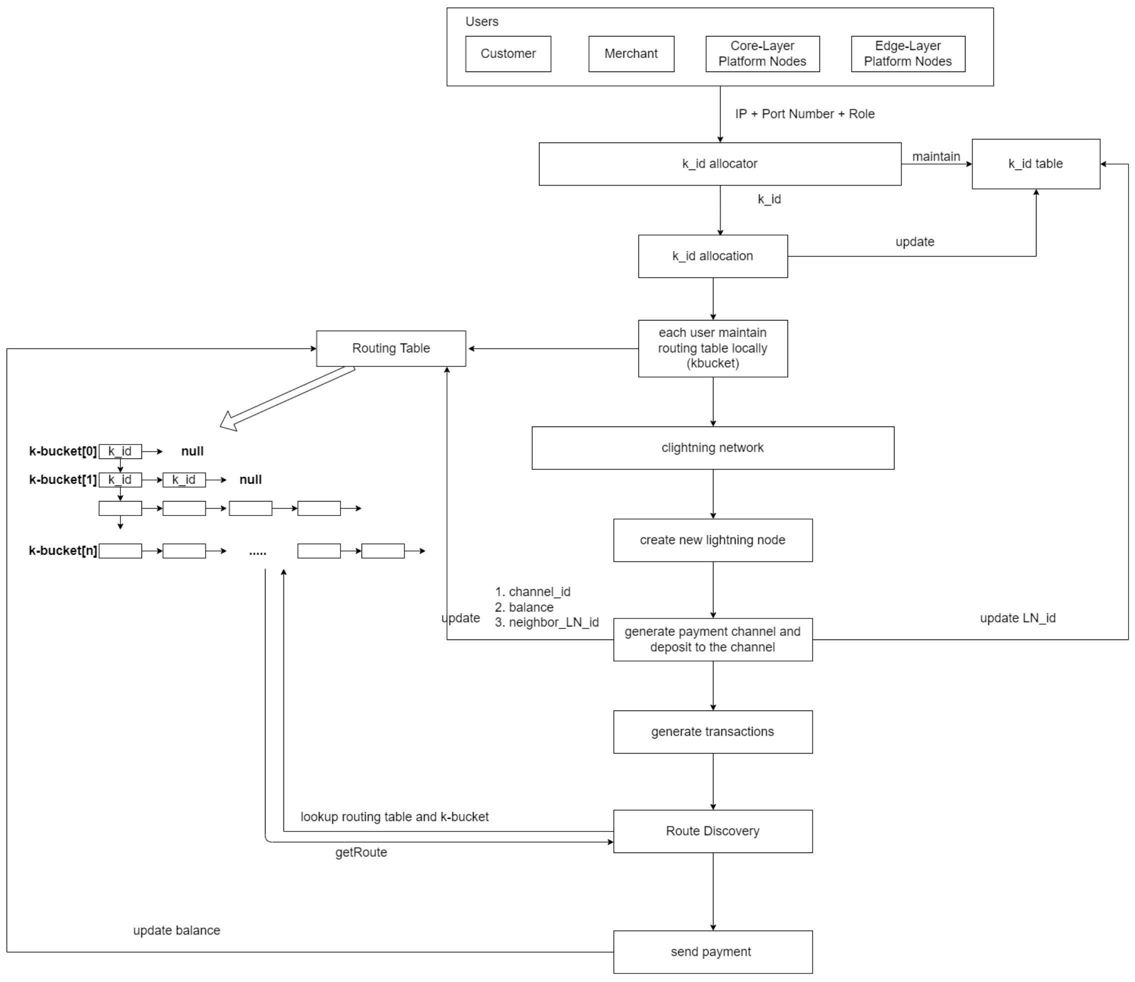


Figure 4.6: K-Routing Algorithm Process

lightning id. The details of the routing table will be described in section 4.2.2. The user will then spawn a new lightning node. The lightning node id of this user will be sent to the k_id allocator, and the allocator will update this data in a k_id table. The user establishes payment channels according to its routing table and deposits funds to these channels. The channel id, channel balance and neighbours' lightning node id will be updated in the local routing table. A route can be discovered using the local routing table and k-buckets when a customer attempts to send a transaction. With the XOR operation, the next hop will be found, and ultimately a path will be returned to the user. A lightning network transaction can now be sent via the discovered path to the destination. Finally, the corresponding nodes will update their local routing tables and wait for the next transaction.

4.2.1 k_id Allocator

Figure 4.7 shows the k_id allocator working process. In our design, the k_id allocators are servers that are used for allocating k_ids to different kinds of user. k_id allocators

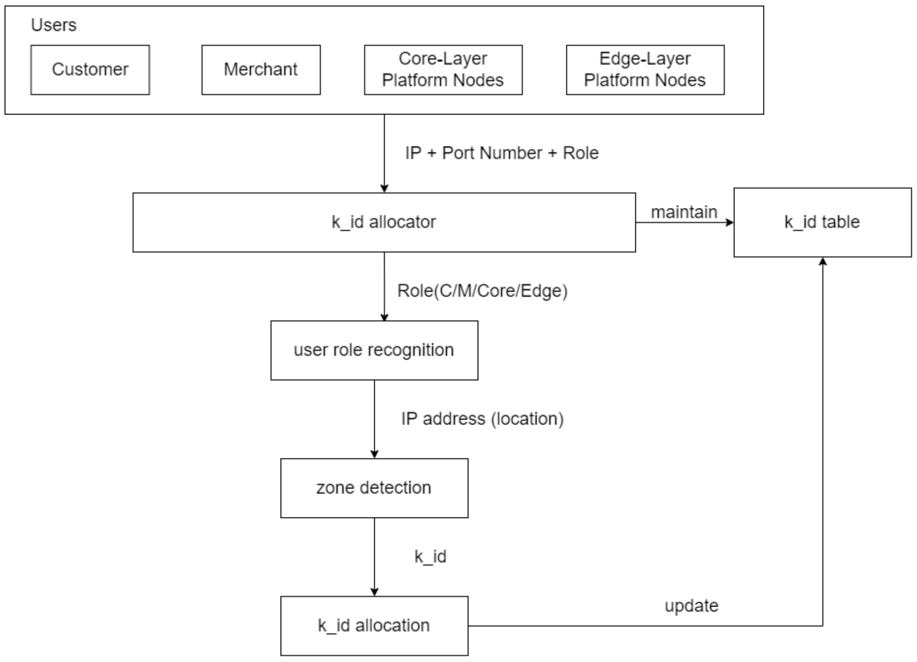


Figure 4.7: k_id Allocation Process

are organized as server clusters. Multiple servers share the same k_id table. When a new node joins the network, a broadcast request will be sent to the cluster for table updates. When a server fails, the system can isolate that server from the system and complete the new load distribution through the load transfer mechanism of each server which avoids single point failure. k_ids will be organized in a binary tree structure for the route discovery process. We assume that each k_id is allocated according to the customer's or merchant's IP address. E-commerce platforms create and maintain Core-Layer and Edge-Layer platform nodes. Each Edge-Layer platform node is responsible for an area containing multiple IP addresses and will establish connections with customers and merchants in that area. We define "zones" for these different ranges. Each zone is a continuous range of k_ids. For example, the Edge-Layer platform node, Edge-1 is responsible for a "zone", and the zone number of this Edge-1 is $zone1 \in [0, 2^{11})$. If the network location of a customer is within this zone, the k_id allocator checks his IP address and allocates an unused unique k_id in the range of zone1 to this user. This design can make sure that each Edge-Layer platform node and its customers/merchants are the closest under XOR distance. The e-commerce platform predicts the total number of customers and merchants and then knows the tree's height. Core-Layer platform nodes are placed in the middle of the binary tree. Figure 4.8 shows the tree structure of the K-routing algorithm.

E-commerce companies maintain an IP table corresponding to IP addresses. In Figure

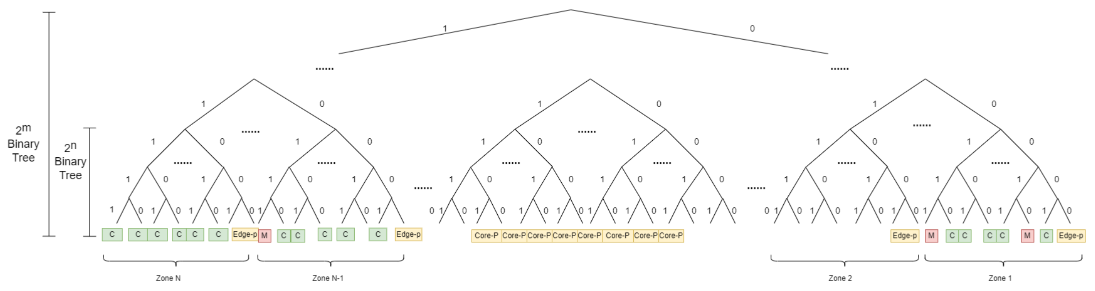


Figure 4.8: The k_id Binary Tree Structure for K-routing Algorithm (C is a customer node, M is a merchant node, Core-P is a Core-Layer platform node and Edge-p is a Edge-Layer platform node.)

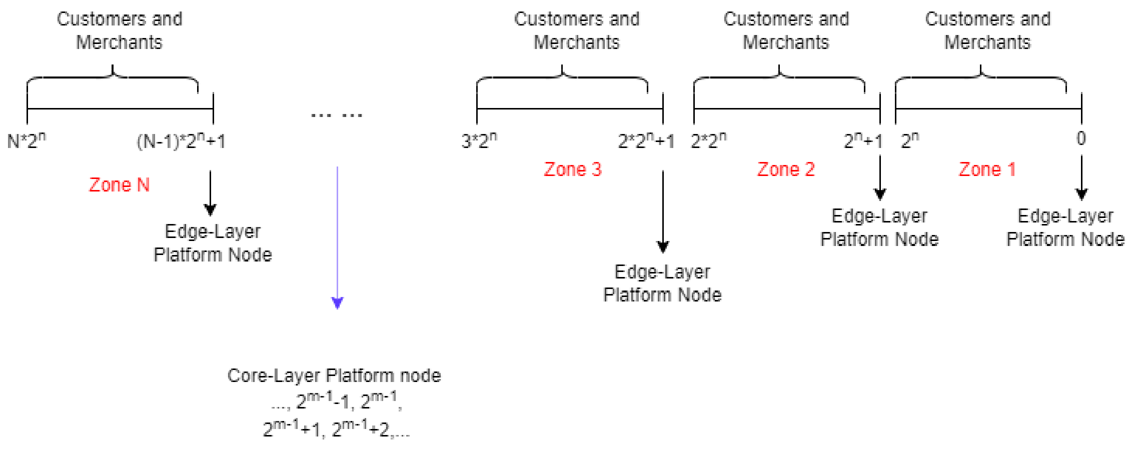


Figure 4.9: Zone Range of Binary Tree

4.9, each Edge-layer platform node is responsible for a certain range of IP addresses. This Edge-Layer node and the customers/merchants belonging to it consist of a Zone. When a customer or a merchant belongs to a certain IP address range, he will establish payment channels to the corresponding Edge-Layer node. We assume the max number of channels a node can have is k. In the binary tree, each zone is in a subtree with a height of n. One Edge-Layer platform node and customers and merchants that have channels with it are placed in such a subtree. All zones form a complete tree, and the total tree height is m. Take Dublin city as an example. From [92], there are 167,081 IP addresses in the Leinster State of Dublin. If a node can establish 1,800 payment channels, as described before, these IP addresses will be distributed to 93 zones. 93 Edge-Layer platform nodes are responsible for these customers and merchants according to their IP tables.

Table 4.1 is the format of k_id table entry.

Table 4.1: k_id Table Entry Format

netaddr	The IP address and port number of the newly joined user.
k_id	k_id allocated to this user.
Role	Role of the user, customer / merchant / Core-Layer platform node / Edge-Layer platform node
LN_id	Lightning node id of this user
zone_number	The zone number of this node.
group_number	The group number of Core-Layer or Edge-Layer platform nodes. If node is a customer or merchant, this is set to 0.

4.2.2 The Routing Table

The routing table is a binary tree whose leaves are k-buckets. In the Kademlia routing algorithm, k-buckets are a list of routing addresses of other nodes in the network, which are maintained by each node and contain the IP address, port, and NodeID for peer participants in the system. In our algorithm, IP address, port, k_id, role, zone number, group number, lightning id, channel id, channel balance, and neighbor's lightning id will be added to the routing table used to find the next hops in the payment network.

Each user will maintain its routing table locally. As we showed in the previous sections, when uses join the network, they will be issued with k_id, role, zone_number, group_number. After the user connects to the lightning network and generates payment channels with other nodes, more information will be acquired. Table 4.2 shows the table recorded by each user. Each record takes 242 bytes. The table size is independent of the network size in our design. Because each node can only establish a limited number of payment channels, and the routing table (k-buckets) records the connection information with other associated nodes. Searches do not change the size of the routing table. Only when other nodes disconnect or connect with this node will it have an impact on the size of the routing table. For platform nodes, if each node can establish 1800 payment channels, it takes around 400kb to record the routing table, including records of customers, merchants and other platform nodes it connects with. For a customer or merchant, it only takes 242 bytes.

Different k-buckets store node information with different distances from that node. For example, in Figure 4.10, a node's k-bucket[0] stores the node information whose distance is 1 from this node. K-bucket[1] stores the node information whose distance is between 2

Table 4.2: Routing Table Format

Parameters	**Meaning**	**size(bytes)**
netaddr	The IP address and port number of the user.	4
k_id	k_id allocated to this user.	4
Role	Role of the user, customer / merchant / Core-Layer platform node / Edge-Layer platform node	1
zone_number	The zone number of this node.	4
group_number	The group number of Core-Layer or Edge-Layer platform nodes. If node is a customer or merchant, this is set to 0.	4
LN_id	Lightning node id of this user	66
channel_id	The channel Identifier.	18
channel_balance	Funds available in the channel.	4
source LN_id	The node providing entry to the channel.	66
destination LN_id	The node providing the exit point for the channel.	66
active	Boolean value, is available for routing. This is linked to the channel flags data, where if the second bit is set, it signals a channel is temporarily unavailable (due to loss of connectivity) OR permanently unavailable where the channel has been closed but not settled on-chain.	1
base_fee_millisatoshi	The base fee (in millisatoshi) charged for the HTLC (BOLT 7).	4

and 3 from this node. Unlike the Kademlia algorithm, only nodes with a direct payment channel connection to this node will be recorded in the k-buckets of this node.

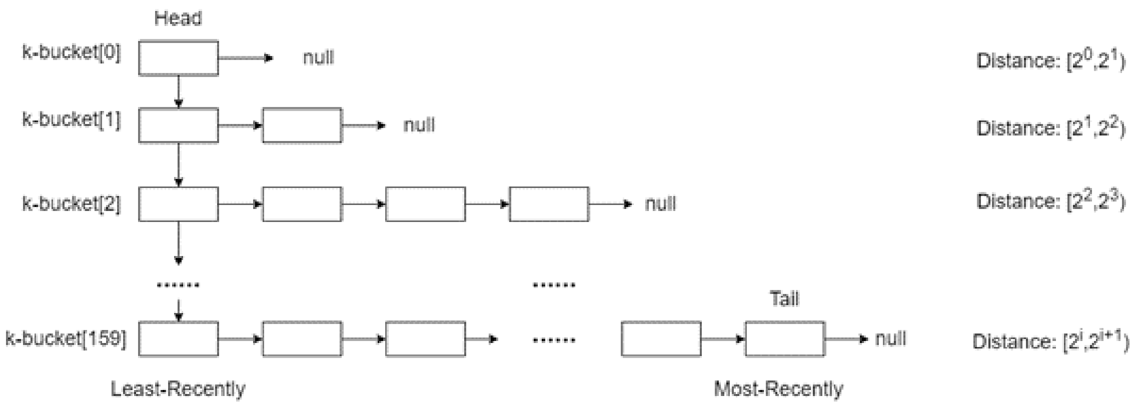


Figure 4.10: Kademlia's Routing Table - KBucket Structure

4.2.3 Maintaining the Routing Table

A node can take on any of four roles in our layer2 based e-commerce system. This is determined when a user joins the network. If it is:

1) a Core-Layer platform node, the allocator will allocate a k_id which is located near the middle of the binary tree and return other Core-Layer platform nodes' k_id in the same group from the k_id table. The allocator will also send this newly joined Core-Layer

platform node's k_id to other nodes in the same group to help them update their routing tables.

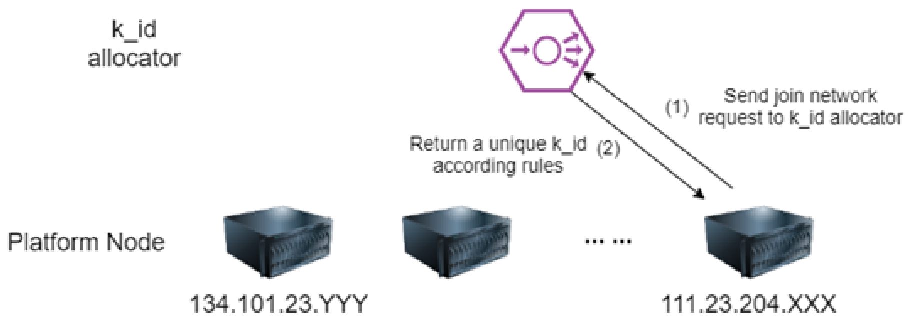

k_id
allocator
(1) Send join network request to k_id allocator
Return a unique k_id according rules (2)
Platform Node
134.101.23.YYY
111.23.204.XXX

Figure 4.11: A Core-Layer Platform Node Joins the Network

2) an Edge-Layer platform node, the allocator determine a new zone for it and return the k_id from the range of zones. The allocator also returns the k_id of a Core-Layer platform node selected from each existing Core-Layer group. The allocator also returns the k_ids of other Edge-Layer nodes in the same Edge-Layer group as the newly added Edge-Layer platform node. This new k_id will be sent to these corresponding nodes allowing them to update their routing tables.

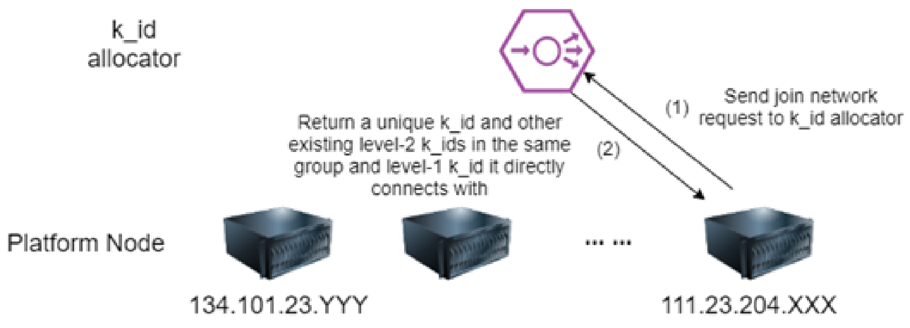

k_id
allocator
(1) Send join network request to k_id allocator
Return a unique k_id and other existing level-2 k_ids in the same group and level-1 k_id it directly connects with (2)
Platform Node
134.101.23.YYY
111.23.204.XXX

Figure 4.12: A Edge-Layer Platform Node Joins the Network

3) customer or merchant: the allocator allocates a unique k_id according to their IP address. The user's k_id and the k_id of the Edge-Layer platform node in this zone will be given to this user. This new k_id will be added to allocator's k_id table to avoid duplicate allocation. Also, this k_id will be sent to the corresponding Edge-Layer platform node by k_id allocator and the platform node adds it to its routing table.

In our design, the k-buckets only record node's k_ids with direct payment channels.

In a distributed network, nodes may frequently join and exit the network due to network fluctuations and other factors. However, the k-buckets save relatively static information. They need to be updated as some conditions change.

There are three ways to update k-buckets.

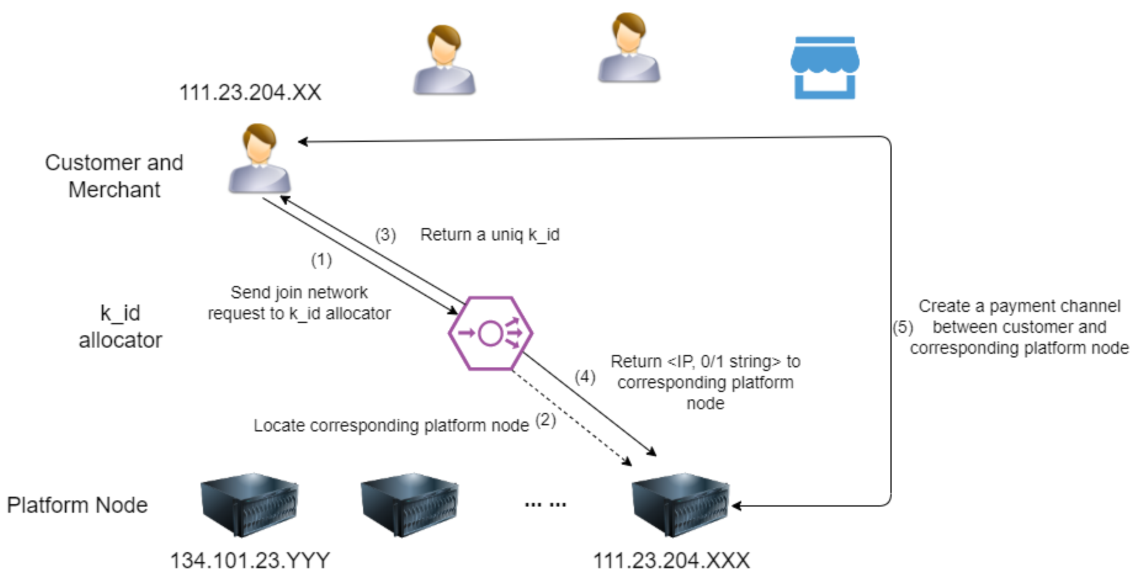

111.23.204.XX
Customer and Merchant
(3) Return a uniq k_id
(1)
Send join network request to k_id allocator
k_id allocator
(5) Create a payment channel between customer and corresponding platform node
(4) Return <IP, 0/1 string> to corresponding platform node
Locate corresponding platform node (2)
Platform Node
134.101.23.YYY
111.23.204.XXX

Figure 4.13: A Customer Joins the Network

1) Actively collect node status and update k-buckets. Any node can initiate a find-node request to the k_id allocator to refresh the node information in the k-bucket. This method is suitable for recovering a temporarily unresponsive node.
2) Passively collect node status and update k-buckets. When receiving requests from other nodes, such as a new corresponding node joining the network, the k_id of this new node will be added to the k-bucket.
3) Detect failed nodes. When the algorithm is determining whether a node in the k-bucket is online or not by initiating a PING request. A node will be deleted from the k-bucket if it is offline.

The newest node will be added to the end of the k-bucket. That is to say, nodes that have been online for a long time have a higher probability of being able to remain in the k-bucket list. This mechanism improves the stability of the K-routing network and reduces network maintenance costs. At the same time, this mechanism can prevent DDOS attacks to a certain extent. Because if the k-bucket is full, the k-bucket only updates after the old node fails.

4.2.4 Route Discovery

The XOR operation is used to calculate the logical distance between two nodes.

$$d(x, y) = x \oplus y$$

The result is 0 when the corresponding bits are the same, and 1 when they are not. The distance range of K-bucket[i] is $[2^i, 2^{(i+1)})$. If a node with ID A wants to find a node with an ID value of B, the K-routing performs a route discovery according to the following steps:

1) Calculate the distance to B, $d(A, B) = A \oplus B$. and see that the result belongs to a particular distance range. We then choose the K-bucket corresponding to this range.
2) Filter out α nodes closest to the target ID from the corresponding K-bucket, and check whether the channel balance on the side of this node is sufficient to support the transaction. If yes, initiate a $FIND\_NODE$ query request to each of these nodes in parallel. If no, the node with an insufficient balance is not considered a valid next hop. If the number of nodes in this K-bucket is less than α, just select this number.
3) For each node that receives the query operation, if it finds that it is B, it will answer that it is the destination; otherwise, it measures the distance between itself and B, and repeats step 2).
4) According to the topology design in the previous section, all Edge-Layer platform nodes know the IDs of the Core-Layer platform nodes connected to them. Therefore, if the α closest nodes to B found by an Edge-Layer platform node do not include a Core-Layer platform node, the corresponding Core-Layer platform nodes will be added as the next nodes. This design is because the IDs of Core-Layer platform nodes are distributed in the center of the k_id Binary Tree. If the IDs of customers and merchants are on the same side of the binary tree, the path between them may only go through the Edge-Layer platform nodes, resulting in a longer final path.
5) In this process, A can find multiple alternative paths that meet the balance requirement. The shortest path will be selected. If two or more paths are the shortest, the path with the highest available balance will be selected.
6) During the query process, nodes that do not respond in time are eliminated immediately. This ensures that the found nodes are online and reduces search time.

In our design, the k_id allocator regulates entry to the network and determines the topology, which introduces centralized control and dependence on a single node. In practice, e-commerce companies can use distributed k_id allocators to mitigate such affects.

4.3 Channel Rebalancing

Our layer2 based e-commerce topology is very different from the existing lightning network topology. The e-commerce company maintains platform nodes. Transactions can flow in both directions through channels of platform nodes. Rebalancing will be more flexible. The customer is the one that sends transactions, and the merchant is the party that receives transactions. In general, the transaction flow direction is from the customer to the platform and then from the platform to the merchant, rather than the reverse. This results in the balance of the customer side continually decreasing and that of the merchant side constantly increasing. If a customer does not have enough funds, he must redeposit more to the channel. Merchants can periodically withdraw money from the channel. Therefore, this section only discusses the channel rebalancing of platform nodes.

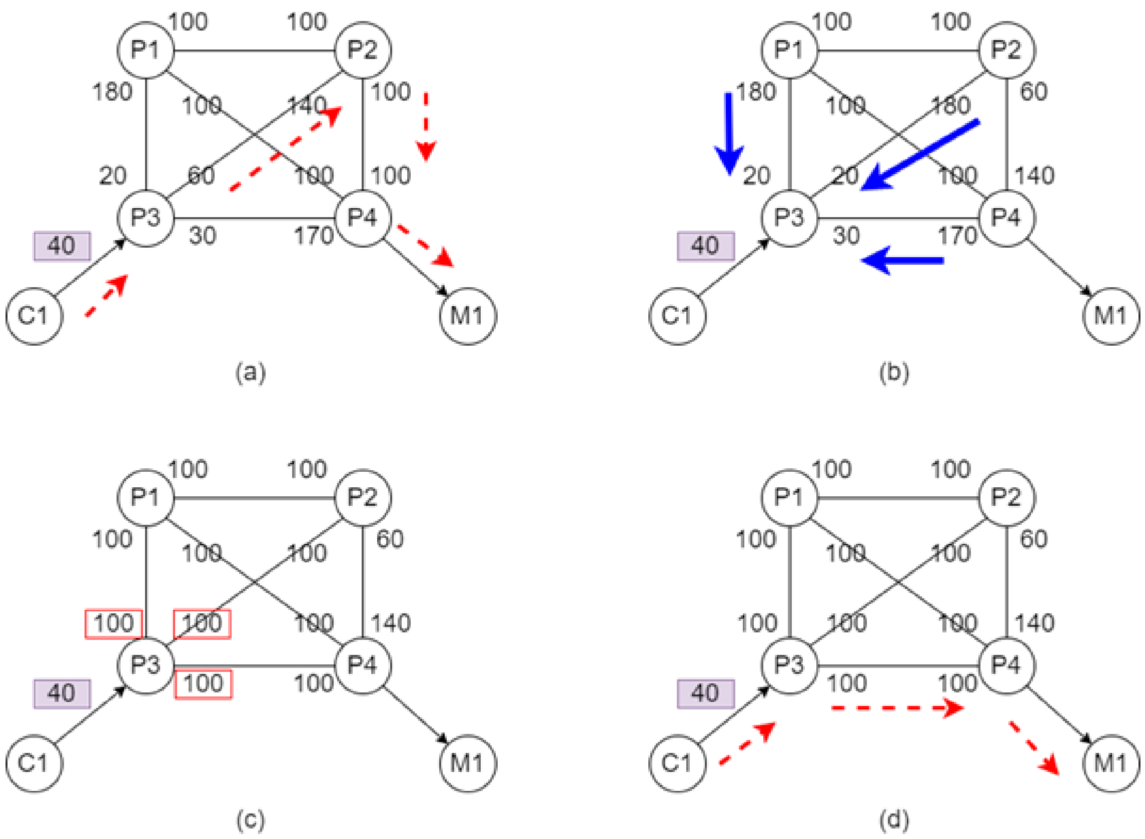


Figure 4.14: Rebalancing of Platform Channels with Small Payments(sub-figure (a) Pathfinding without Rebalancing, (b) Corresponding channels self-rebalancing, (c) After Rebalancing, (d) Pathfinding after Rebalancing)

4.3.1 Rebalancing of Platform Node Channels

The platform node channel rebalancing consists of (1) moving funds within a channel and (2) moving funds through a circle of nodes. For (1), if the balance on one side of the channel is lower than β after transactions (β can present the percentage of the total capacity of this channel), this node generates a payment invoice and sends it to the node on the other side. The node on the other side can then send the corresponding amount of funds and restore the channel to its original state. It also suits the scenario of only one e-commerce provider. The platform nodes are maintained by one provider. All funds on platform nodes belong to this provider. If the provider wants to control rebalancing in a simpler way, it may introduce a third party to manage the status of each channel which causes extra message exchange and centralization issues. In addition, not all channels need to be rebalanced at the same time. In this case, having the rebalancing happen inside part of channels would be a better solution.

For (2), if the value of the transaction is greater than the channel balance along the path, more funds will be moved in a circle to ensure that at least one channel can support the payment. It also works in the scenario with multiple e-commerce providers. Different providers maintain platform nodes. The total balance on each node needs to be the same after rebalancing.

Figure 4.14 shows the rebalancing process of platform nodes. The number on the graph is the balance of each side. Customer $C1$ wants to send 40 to merchant $M1$. The balances of channel $P3 \rightarrow P1$ is 20, and channel $P3 \rightarrow P4$ is 30, which is not enough. In sub-figure (a), path $C1 \rightarrow P3 \rightarrow P2 \rightarrow P4 \rightarrow M1$ is selected. Then if $C1$ wants to generate another transaction with the value of 40, no path is found because the balance of channel $P3 \rightarrow P2$ is insufficient. The payment channel is bidirectional and the platform nodes can invoice each other and will be paid. In (b), $P3$ generates invoices and sends them to relevant nodes $(P1, P2, P4)$, and these nodes send payments to $P3$ for rebalancing channels. Now in (c), the funds of each channel are equally distributed to both sides. In (d), a shorter path can be selected to transfer the new payment after rebalancing.

If platform nodes are not maintained by the same e-commerce company, a circular movement is needed. In Figure 4.15, if $C1$ wants to send 120 to merchant $M1$. The balance on $P3$ side cannot support the payment. A circular movement will be generated. $P3$ moves 25 via $P3 \rightarrow P2 \rightarrow P4 \rightarrow P3$. Now $P3$ has a channel that can support the transaction with a value of 120, but the total amount of funds of each node does not change.

In this chapter, we outlined our design for a blockchain layer2 based e-commerce architecture. With 2-Level of platform nodes, tree-based K-routing algorithm, and rebalancing mechanism, our design can support the network scale of Amazon theoretically. In the next chapter, we will do experiments and discuss whether it can achieve this goal.

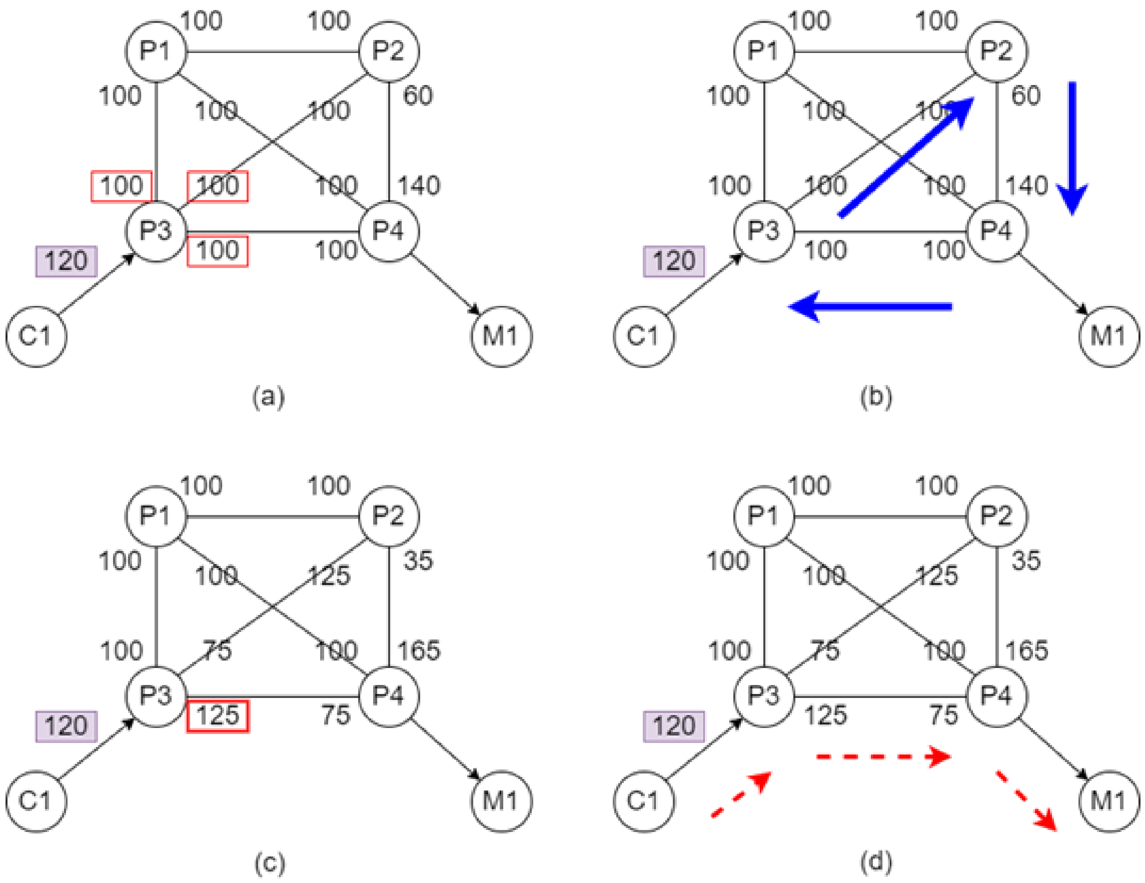


Figure 4.15: Rebalancing of Platform Channels with Large Payments

Chapter 5

Evaluation

In this chapter, we simulate the e-commerce payment scenario by building a series of small-scale structured payment channel networks to demonstrate the feasibility of our approach and evaluate network performance.

5.1 Experiments Setup

We validate the design of the previous chapter by building a small blockchain layer2-based e-commerce topology. The prototype was built using C-lightning nodes deployed in a custom-built Lightning Network. C-lightning nodes and bitcoin nodes run on Ubuntu 20.04 system with 2 CPU cores. The Bitcoin testnet is used as the Layer1 blockchain. The experiment explores two aspects: 1) routing algorithm and 2) topology. We assume transaction fees charged by all platform nodes are the same.

Section 4.1 described our proposed e-commerce network topology. There are three layers: 1) Core-Layer platform nodes, 2) Edge-Layer platform nodes, and 3) Leaf-Layer customers and merchants. We set up 3 different size topologies for experiments. The number of platform nodes for each level is shown in Table 5.1. For example, in $Small_{(8,8,16)}$, (8,8,16) means that there are in total 8 Core-Layer platform nodes, 8 Edge-Layer platform nodes and 16 Leaf-Layer nodes which include customer and merchant nodes. The 8 Core-layer nodes are divided into two groups of 4 nodes each. The 8 Edge-Layer nodes are divided into two groups of 4 nodes each. The 16 Leaf-Layer nodes are comprised of 8 customer nodes and 8 merchant nodes in this topology. Similarly, $LargeHighCore_{(40,84,168)}$ represents a topology that has 40 Core-Layer nodes (10 groups of 4 nodes each), 84 Edge-Layer nodes (21 groups of 4 nodes each), and 168 Leaf-Layer nodes (84 customer nodes and 84 merchant nodes).

Table 5.1: 2 Levels Platform Nodes Distribution in 3 Topologies

Topology Type	$Small_{(8,8,16)}$	$Medium_{(8,16,32)}$	$Large_{(8,84,168)}$	$LargeHighCore_{(40,84,168)}$
Core-Layer group Node Count	4	4	4	4
Number of Core-Layer groups	2	2	2	10
Total Core-Layer platform nodes	8	8	8	40
Edge-Layer group Node Count	4	4	4	4
Number of Edge-Layer groups	2	4	21	21
Total Edge-Layer platform nodes	8	16	84	84
Channel Count	60	108	516	1240

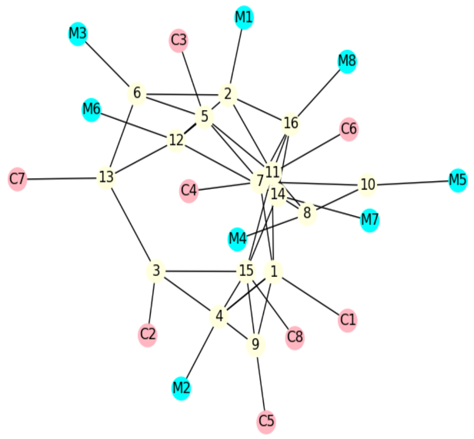
Small, Medium and Large topologies have the same number of Core-Layer platform nodes, but the number of Edge-Layer platform nodes expands horizontally which helps explore the performance of the routing algorithm when the number of Edge nodes increases. LargeHighCore topology maintains the same number of Edge-Layer platform nodes as the Large topology, but has more Core-Layer platform nodes. This is used to validate the performance of the routing algorithm when the number of Core-Layer nodes increases.

Table 5.2: Different Sizes of Topologies Under Unstructured Network

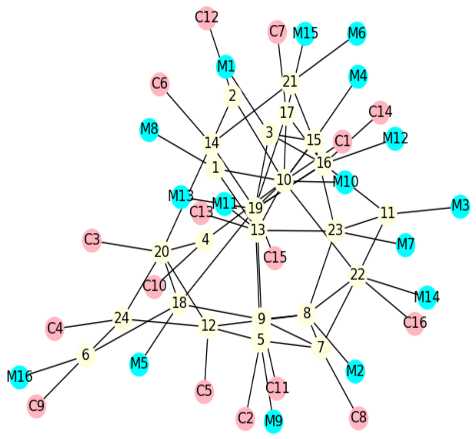
Topology Size	$Small$	$Medium$	$Large$
Number of Platform Nodes	16	24	92
Number of Customers	8	16	84
Number of Merchants	8	16	84
Channels of Most Connected Node	5	9	9

In order to compare the performance of the topology we proposed with the unstructured network, we also set up 3 unstructured topologies of different sizes. As shown in Table 5.2, the number of nodes in the unstructured topologies is the same as in Table 5.1. But the channels of each node are randomly generated. Each platform node randomly

establishes payment channels with the other two platform nodes. Each customer or merchant randomly selects a platform node. In these topologies, the most connected platform node has established a total of 9 payment channels.
Figure 5.1 shows the 3 simulation topologies.



(a) Small Topology



(b) Medium Topology

5.2 Experiment One – Routing Algorithm Performance

5.2.1 Evaluation Metrics

This experiment aims to compare our proposed K-routing algorithm with Dijkstra's algorithm, which is used by the C-lightning implementation and is the standard choice in the production lightning network.

The evaluation metrics in this experiment include:

1) Transactions per second. This refers to the number of transactions completed across the network in one second. Each customer node generates a certain number of transactions targeted at different merchants. For example, Customer 1 (C1) generates 100 transactions to Merchant 1,. . . ,n. We obtain the time cost after all transactions are completed in the network and calculate the transaction per second using

$$\frac{number\, of\, transactions}{total\, time\, cost} \tag{5.1}$$

2) Total transaction fees. In our experiment, the transaction fees charged by all platform nodes are the same: 1 msatoshi. There is a *listpays* command in c-lightning for querying payment status. We acquire the amount paid by the customer and the amount received by the merchant using this command. The result of $(the\, amount\, paid\, by\, the\, customer\, -$

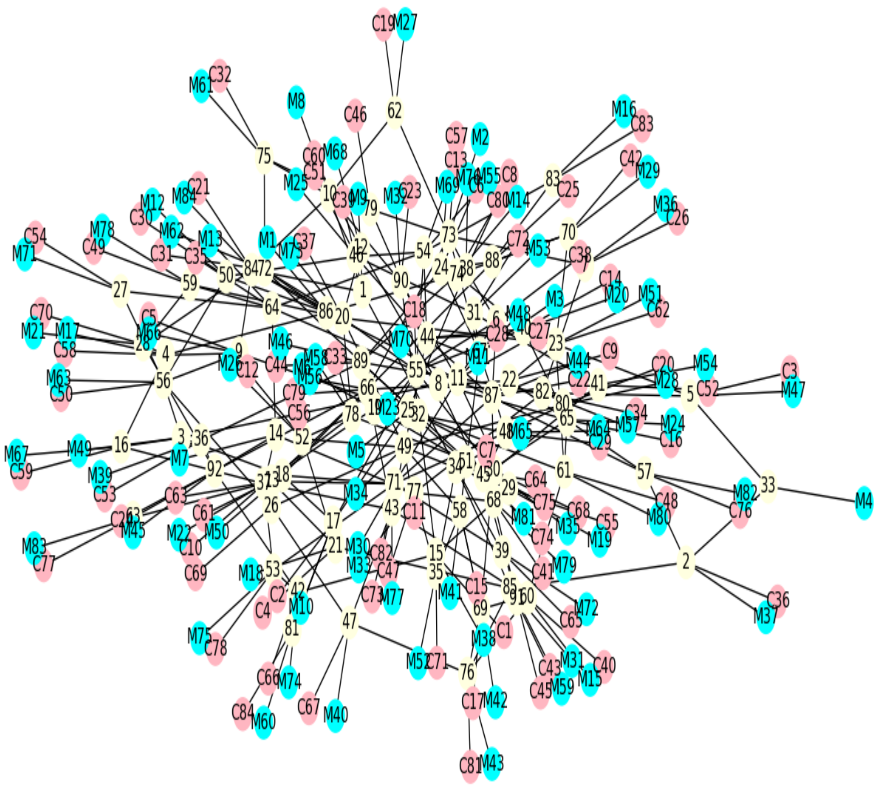

(c) Large Topology

Figure 5.1: Nodes Distribution on Unstructured Topologies

the amount received by the merchant) $\times$ *number of transactions* yields the total transaction fees.

3) Path length. This refers to the average path length from customers to merchants in the network. There is no direct returned value in the c-lightning network that will tell us the average path length. But 1 msatoshi presents one intermediate node in our assumption. Therefore, our calculation of the transaction fees indirectly indicates the path length.

4) Transaction concentration. This metric aims to analyse the number of transactions passing through each Core and Edge platform node from a given traffic set. The *listforwards* command in the c-lightning network displays all HTLCs that have been attempted to be forwarded by the lightning node. We can use this command to monitor the number of transactions passing through each platform node.

5.2.2 Experiment Results and Discussion

Transactions per second

We acquire the time cost of all transactions from customers to merchants and average this to calculate the transactions per second. Figure 5.2 show the throughput of Small, Medium, and Large topology when the number of transactions from customers to merchants increases from 100 to 500.

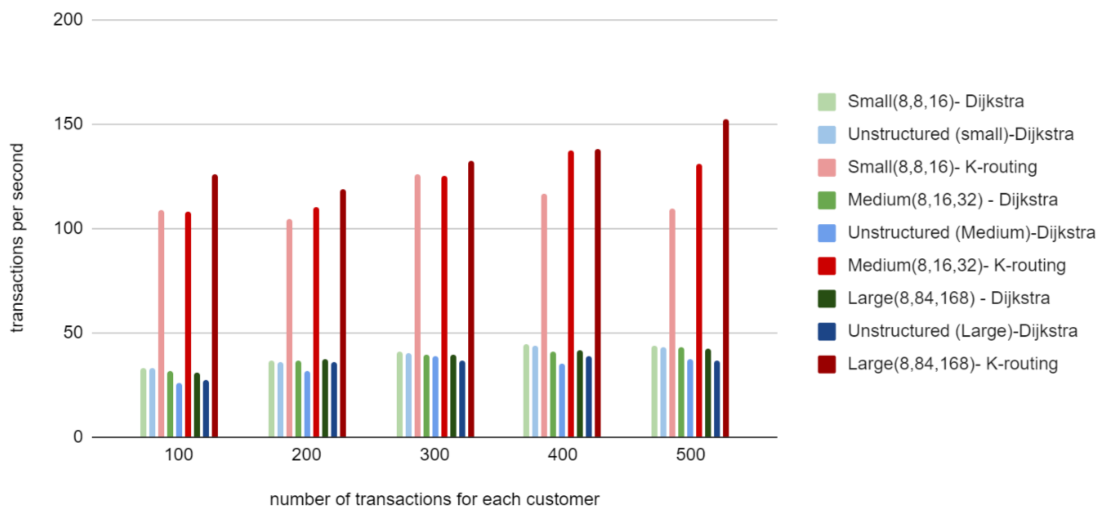


Figure 5.2: Throughput of Dijkstra Algorithm and K-routing Algorithm

In Figure 5.2, we analyze the result of Dijkstra and K-routing in our proposed topology, also compare with the result of an unstructured Lightning topology with the same number of customers, merchants and platform nodes. We can see that the transactions per second using the K-routing algorithm are more than 3 times that of the Dijkstra algorithm. In

Figure 5.1, we can also see that as the number of nodes increases, the transactions per second across the network also increases with the K-routing algorithm. Using Dijkstra, the transactions per second decreases slightly as the number of nodes increases. For Dijkstra's algorithm, as the number of nodes increases, Dijkstra's search space becomes larger. At the same time, in the process of finding the shortest path, a large number of temporary paths are generated and these occupy a large amount of memory space, resulting in the low efficiency of the algorithm. Although the search space becomes larger for the K-routing algorithm, only a limited number of nodes need to be searched. This significantly reduces memory requirements and improves search speed. We find that K-routing has better scalability than Dijkstra's algorithm and is more suited to large-scale networks.
Comparing the throughput of our proposed topology and unstructured lightning topology, we find the fat tree based topology can deal with more transactions per second. We conducted further experiments to find the reason.

Each customer generates a transaction to each merchant which can help us find the path length of each transaction in the network. Figure 5.3 show the count of occurrences of the path length in the unstructured and fat tree based topologies. We find the longest path in the small topology is 6 hops. For fat tree based topology, the path length is concentrated in 3 or 4 hops. The path length of unstructured topology is concentrated in 4 hops.

In the Medium topology, the longest path is still 6 hops. The path length under fat tree based topology is concentrated in 4 hops and that of unstructured topology is concentrated in 4 or 5 hops.

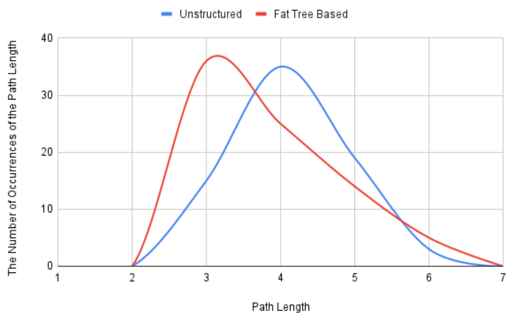
In the Large topology, the longest path of unstructured topology is higher than the fat tree based topology. In the fat tree topology, the path length is concentrated at 5 hops, which is smaller than the path length of the unstructured topology.

From the result above, we can find the majority of path length has been shortened in the fat tree based topology. More transactions can be passed from customers to merchants in the fat tree based topology than an unstructured lightning topology. So, our proposed topology can provide better performance than the unstructured lightning network.

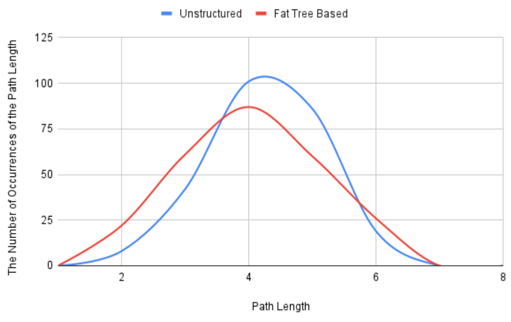
Transaction concentration

We acquire the number of transactions forwarded by each platform node in the Core-Layer and Edge-Layer. The following figures show the results.

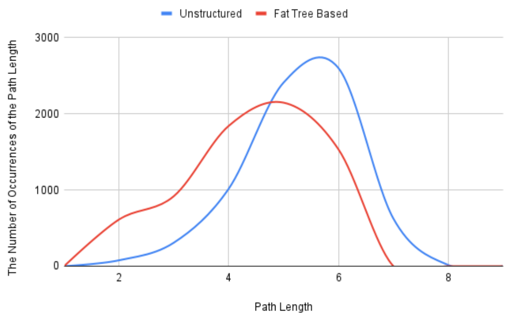
We find that more transactions are distributed on Edge-Layer platform nodes in Figure



(a) Small Topology



(b) Medium Topology



(c) Large Topology

Figure 5.3: Count of Occurrences of the Path Length in the Unstructured and Fat Tree based Topologies

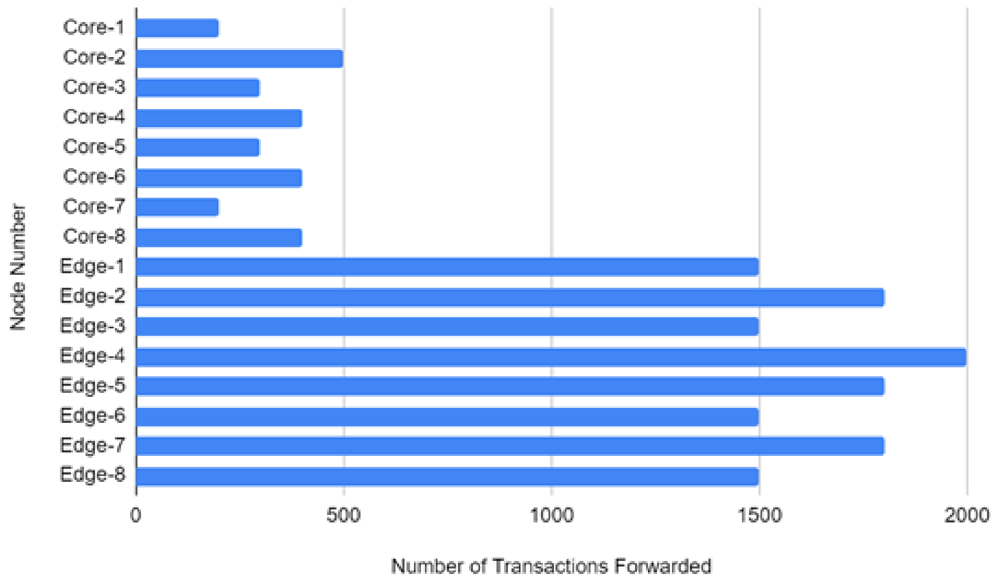


Figure 5.4: The number of transactions forwarded by each platform node in $Small_{(8,8,16)}$

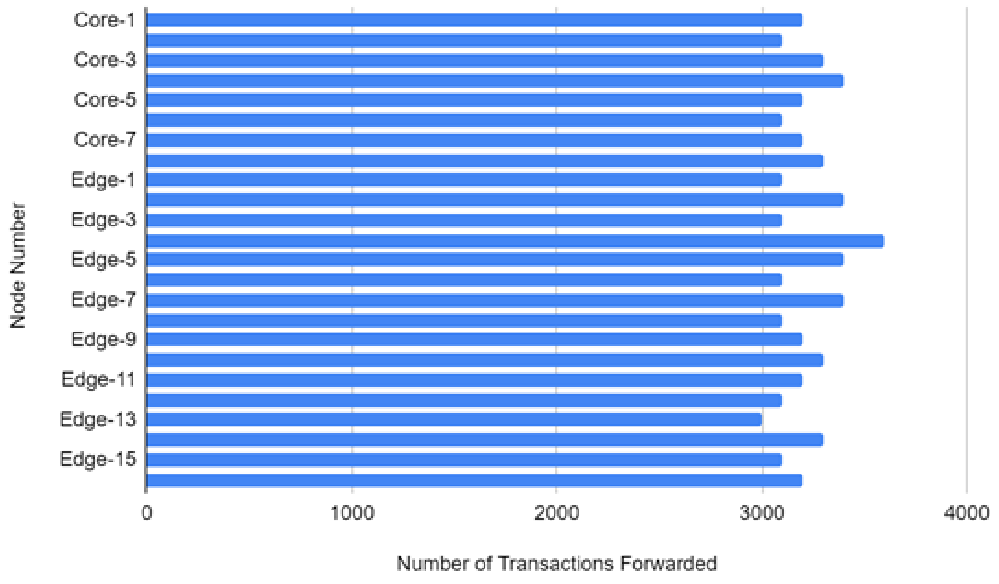


Figure 5.5: The number of transactions forwarded by each platform node in $Medium_{(8,16,32)}$

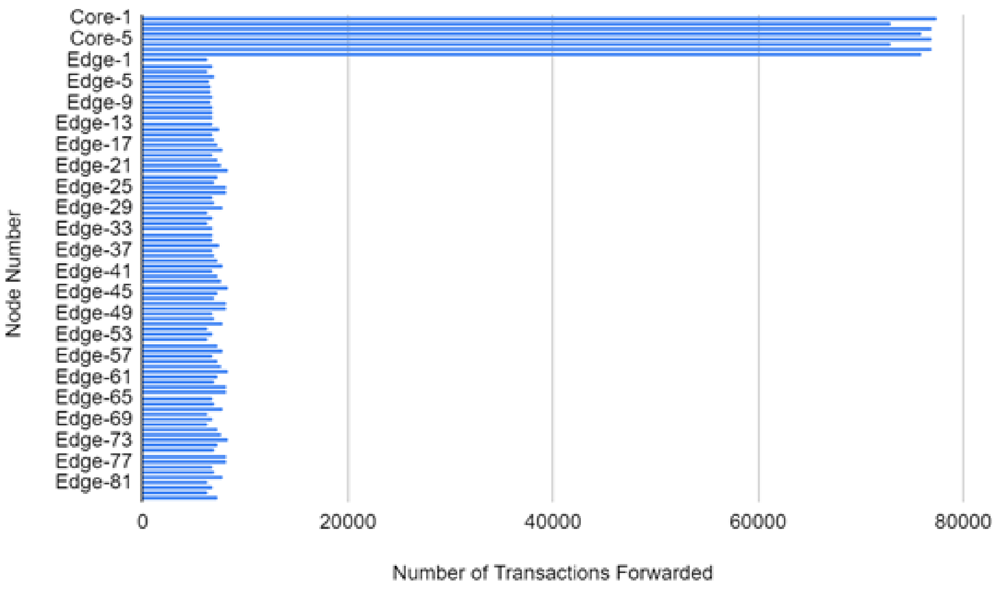


Figure 5.6: The number of transactions forwarded of each platform node in $Large_{(8,84,168)}$

5.4. There are only 2 groups of Edge nodes in $Small_{(8,8,16)}$. Two groups of adjacent nodes are directly connected. Most paths that go through Edge-Layer platform nodes are shorter than these that would go through Core-Layer platform nodes. Therefore Edge-Layer platform nodes are more likely to be selected. In Figure 5.5, we can see that transactions are more evenly distributed across the network. In $Medium_{(8,16,32)}$, there are 4 groups of horizontally connected Edge nodes. In the pathfinding stage, in order to shorten the path length, Core nodes are more likely selected. Therefore, more transactions are forwarded by Core nodes than before. However, as the Edge-Layer groups grow in number, the situation changes.

In $Large_{(8,84,168)}$, the number of Edge-Layer groups continues to grow and Figure 5.6 shows the result. We find that most of the transactions are forwarded by Core nodes. Since Edge-Layer groups expand horizontally, going through Edge-Layer nodes will cause the path length to increase significantly. Therefore, more and more transactions will choose to go through Core-Layer nodes to shorten the path length. This will cause congestion in the Core-Layer nodes and exhaustion of the corresponding channel balances. Increasing the number of groups in the Core-Layer is a possible solution to balance the load. Therefore, we do a fourth experiment and setup a $LargeHighCore_{(40,84,168)}$ topology. Figure 5.7 shows the result of $LargeHighCore_{(40,84,168)}$ with 10 groups of Core-Layer platform nodes.

The traffic patterns from customers to merchants are the same as those we used in $Large_{(8,84,168)}$. Transactions are more evenly distributed to Core and Edge platform nodes in Figure 5.7. From 4.2.4, if paths have the same length, the one with the larger channel

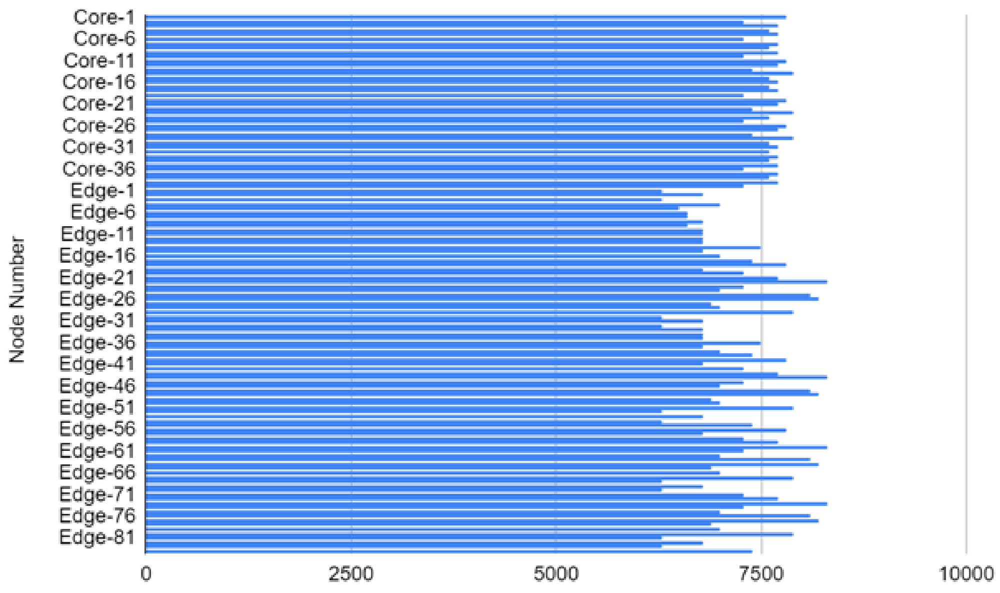


Figure 5.7: The number of transactions forwarded of each platform node in $LargeHighCore_{(40,84,168)}$

balance is selected. Therefore, traffic is distributed more evenly to each Core node.

We set up four topologies and analyse the traffic passing through Core-Layer and Edge-Layer platform nodes from a given traffic set this part. We find that when Edge-Layer has slightly more groups than Core-Layer, choosing Edge-Layer platform nodes or Core-Layer platform nodes as intermediate nodes does not cause a significant change in path length. The traffic passing through these nodes are also nearly evenly distributed. However, when the number of groups of Edge-Layer nodes is much more than that of Core-Layer, the choice of Edge-Layer platform nodes for most nodes will increase the path length. Therefore, the customer may prefer to choose the shortest path which passes through Core-Layer platform nodes. As the number of transactions increases, this causes most of the transactions to be forwarded through the Core-Layer platform nodes. At this time, appropriately increasing the number of groups of Core-Layer platform nodes will help alleviate the issues of transaction concentration.

Transaction fees to Path Length

Figure 5.8 shows that the path lengths of K-routing and Dijkstra are basically the same, although they mostly use different paths to send a transaction. We find that the K-routing algorithm can find paths as short as Dijkstra but does this in less time.

We conclude from experiment 1 that K-routing has the potential to be applied to large-scale networks. As the search space expands, the path length does not change significantly. The time cost to find this path is significantly lower than with the Dijkstra algorithm.

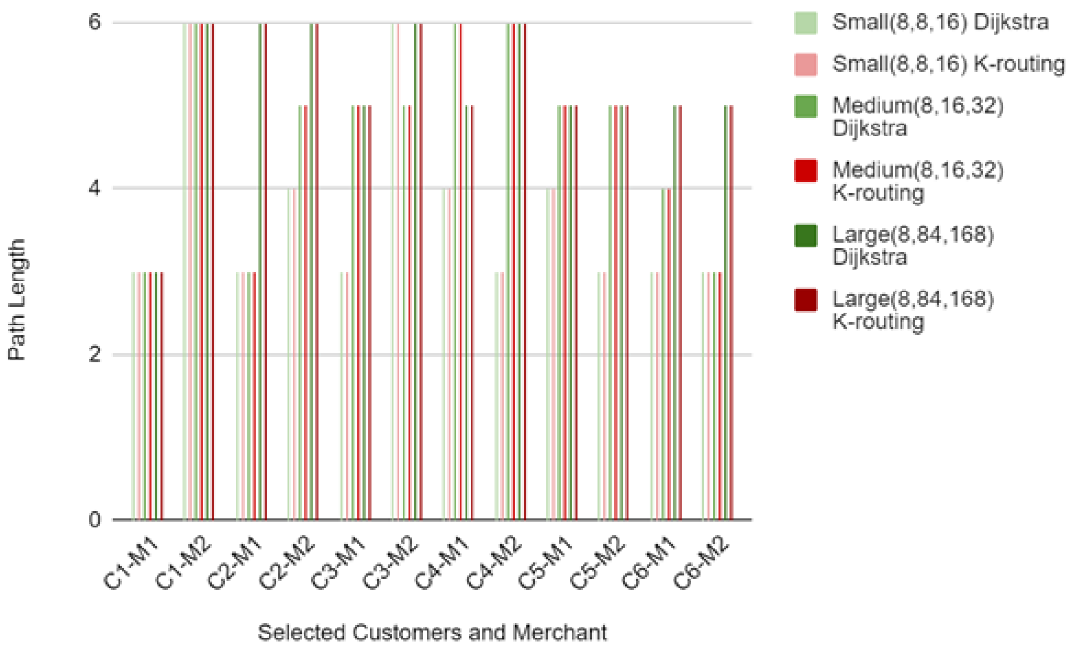


Figure 5.8: Path Length from selected customers to selected merchant

5.3 Experiment Two - Topology

5.3.1 Evaluation Metrics

We find that the K-routing algorithm can find paths as short as Dijkstra, the shortest path between any two nodes. As the network scale expands, the number of hops to reach the most distant merchant under the shortest path is a problem that needs to be discussed. If the number of hops to the merchant increases significantly with the expansion of the network scale, it will lead to time delays across the entire system and an increase in transaction fees. This experiment also analyses the topology availability in the face of partial network failure.

The evaluation metrics we use are as follows.

1) Path length limitation. This refers to the number of hops to reach the most distant merchant under the shortest path. It compares the production unstructured Lightning topology with our structured topology.

2) Average path length. This refers to the average path length from all customers to merchants in the network. The formula is

$$\frac{PL(C_1M_1) + PL(C_1M_2) + \cdots + PL(C_nM_k)}{n \times k} \tag{5.2}$$

$PL(C_1M_1)$ is the path length from Customer1 to Merchant 1. n is the number of customers. k is the number of merchants. This metric increases if a longer path is chosen due to a failed node. We examine how this metric changes in the face of partial failure of network.

5.3.2 Experiment Results and Discussion

Path length limitation

We can find from Figure 5.8 in section 5.2.3, when all nodes are responsive and have enough channel balance, the path length from a customer to the most distant merchant is 6 hops. We also look at as how far a node is from the node most distant from it in the graph with the shortest paths in the production Lightning Network in the time period of February 2018 to May 2022. Figure 5.9 [93] shows this value over time.

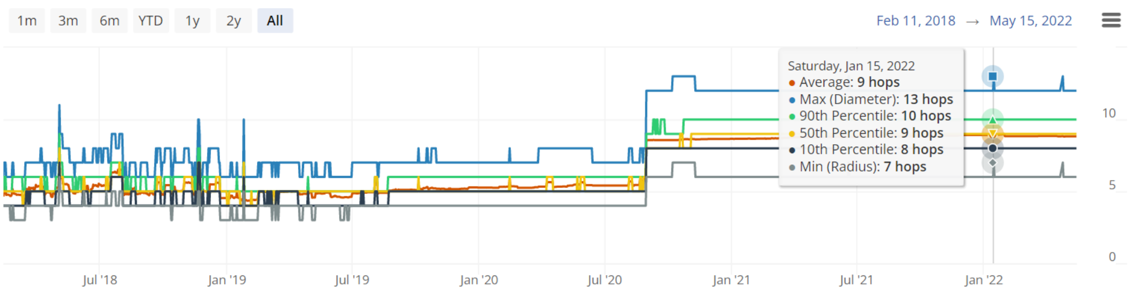


Figure 5.9: The Maximum Number of Hops to Reach Another Node in the Unstructured Lightning Network

There are several metrics in Figure 5.9, including average path length(red line), maximum path length(blue line), minimum path length(grey line). From February 2018 [93], as the lightning network size and the number of participants expand, the path length of a node to its most distant node grows from 6 to 13 hops. For an e-commerce network, such a long path will result in much overhead, including transaction fees that users need to pay and transaction delays.

Our proposed topology divides platform nodes into two levels. The levels each expand horizontally. Unlike the unstructured Lightning Network, whose path length increases as the network expands, our topology can increase the network scale without increasing the path length. This decreases transaction delay and transaction fees paid to intermediate nodes.

Average path length

In this experiment, we take Core-Layer platform nodes in our test network offline one by one. There is no simple command in the c-lightning network that will tell us the average path length, but 1 msatoshi represents one intermediate node in our assumption.

Therefore, the transaction fees we calculate indirectly indicate the path length. Figure 5.10 to Figure 5.13 shows how the average path length from customers to merchants changes as we progressively take more Core-Layer platform nodes offline.

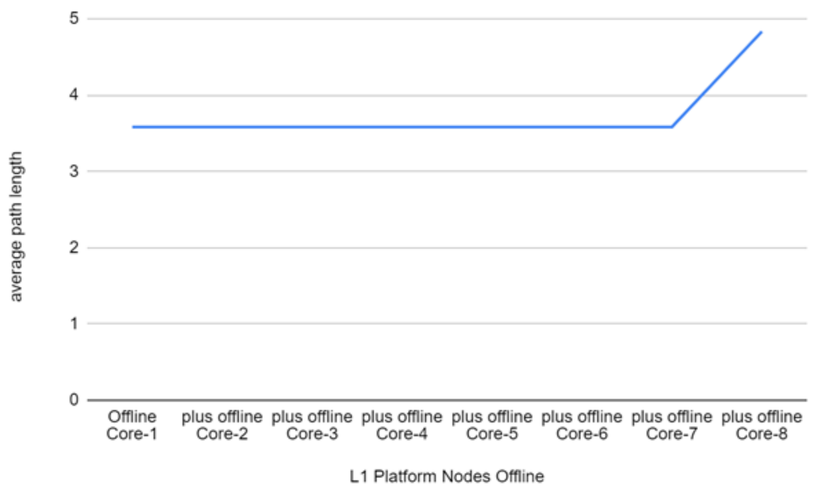


Figure 5.10: $Small_{(8,8,16)}$ - Average Path Length after Offline Core-1, Core-2, …., Core-8 One by One

In Figure 5.10, we see that the unavailability of the initial Core-Layer platform nodes does not significantly affect the average path length of the entire network, although it will increase the load on the remaining nodes. The average path length increases abruptly only when all Core nodes are unavailable.

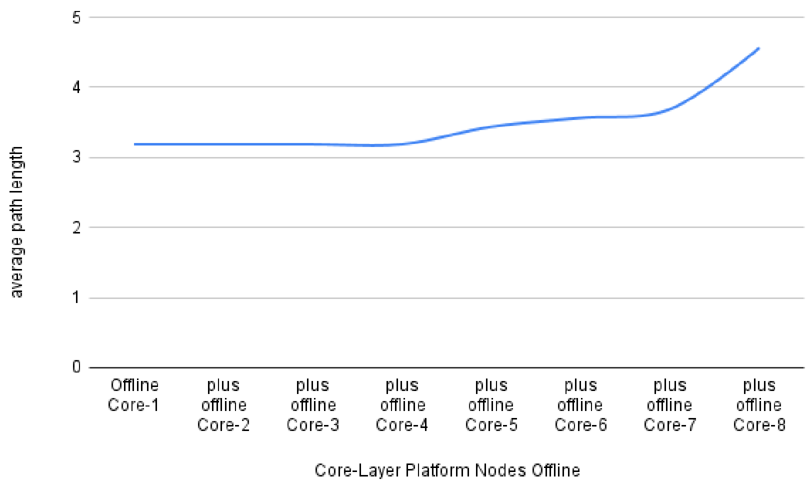


Figure 5.11: $Medium_{(8,16,32)}$ - Average Path Length after Offline Core-1, Core-2, …., Core-8 One by One

In Figure 5.11, as the number of Edge-Layer nodes increases, when offline Core-5 nodes, the average path length slightly increases. It does not increase significantly until the last node goes offline.

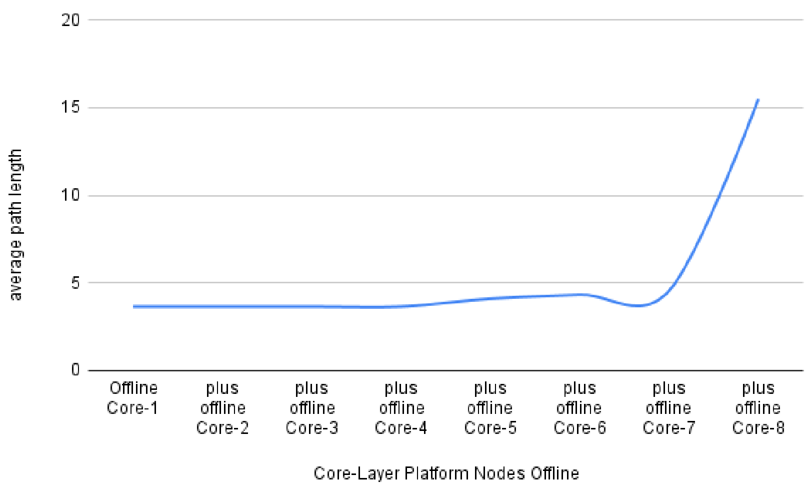


Figure 5.12: $Large_{(8,84,168)}$ - Average Path Length after Offline Core-1, Core-2, ..., Core-8 One by One

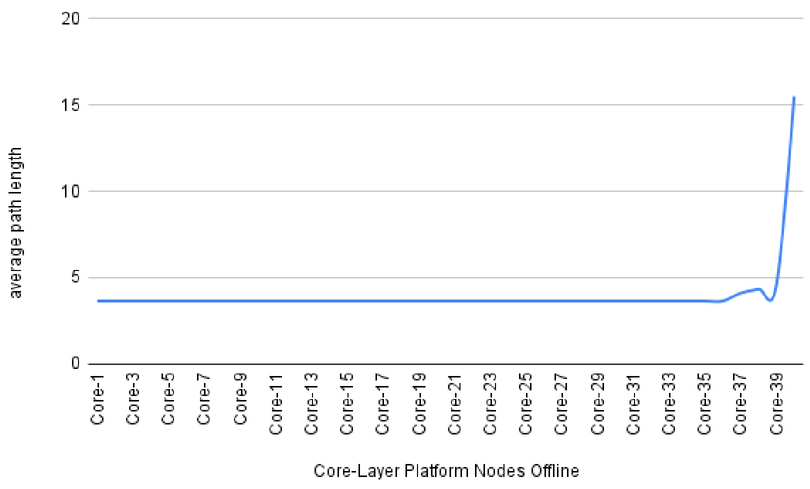


Figure 5.13: $LargeHighCore_{(40,84,168)}$ - Average Path Length after Offline Core-1, Core-2, ..., Core-40 One by One

In Figure 5.12 and 5.13, only when the last group of Core-Layer nodes offline, the average path increases. Compared with Figure 5.12, there are more groups of Core-Layer nodes in Figure 5.13. We find that our topology can handle node failure well. At the same time, multiple Core groups can improve the fault tolerance of the network.

In this experiment, we prove that our structured Lightning Network topology can shorten the path length compared with the unstructured Lightning Network topology. In addition, the failure of a small number of Core-Layer platform nodes will not affect the availability of the whole system.

5.4 Implementing an Amazon-Scale Network

Amazon's online shopping system can support hundreds of millions of users' purchase operations. Whether our proposed system can achieve this level of performance will be an essential criterion to measure whether the system can be applied in practical scenarios. In this section, we will discuss what an Amazon-scale network based on payment channels would look like.

5.4.1 Network Size Topology

Amazon has about 300 million customers and 2.5 million merchants. In section 4.1, we calculated that if the number of platform nodes in each Edge-Layer group is 4 and when the number of channels of a node can support is 1,800, about 168,498 Edge-Layer platform nodes are required to support an Amazon-scale e-commerce network and the optimal number of platform nodes in each Core-Layer group is 99 .

5.4.2 Routing Table Size

In section 4.2.2, we described the format of the routing table. Each node maintains its routing table, which we call k-buckets. Table 5.3 shows the routing table size of 4 different topologies and the time cost of 100 transactions simulation.

Table 5.3: Routing Table Size and Simulation Time cost

Topology Size	**Role of Nodes**	**Routing Table Size (bytes)**	**Time Cost α=3 (milliseconds)**	**Time Cost α=4 (milliseconds)**
$Small_{(8,8,16)}$	Core-Layer Platform Nodes	1,210	915.857	917.165
	Edge-Layer Platform Nodes	1,694		
	Customers/Merchants	242		
$Medium_{(8,16,32)}$	Core-Layer Platform Nodes	1,694	927.028	916.988
	Edge-Layer Platform Nodes	1,694		
	Customers/Merchants	242		
$Large_{(8,84,168)}$	Core-Layer Platform Nodes	5,808	842.493	851.635
	Edge-Layer Platform Nodes	1,694		
	Customers/Merchants	242		
$LargeHighCore_{(40,84,168)}$	Core-Layer Platform Nodes	5,808	895.105	956.278
	Edge-Layer Platform Nodes	10,890		
	Customers/Merchants	242		

Core-Layer platform nodes keep a record of their Edge-Layer platform nodes in their k-buckets. Edge-Layer platform nodes keep a record of their Core-Layer nodes and customer/merchant nodes in their k-buckets. Customers or merchants only record the data of the Edge-Layer nodes they directly have payment channels with. So, the routing table size is the same during the route discovery process if there is no node failure. The client only needs to confirm there is no node closer to the target than the previous one in a time period. It does not need to wait for all α queries to return responses which avoids latency caused by single node failure. Therefore, the time cost does not change significantly in these four topologies when α increases.

5.4.3 Time cost to setup the network

The time cost to set up the network is another issue we need to consider. Establishing a large scale layer2-based e-commerce network means a great amount of on-chain transactions will be generated on the blockchain network to open payment channels. Take the Bitcoin blockchain as an example. A bitcoin transaction is usually 250-500 bytes. Only about 2,000 transactions can fit within a block. The confirmation time for a block is 10 minutes. A block can only process an average of 3-4 transactions per second. Whether an Amazon-scale layer-2 based network can be built in an acceptable time with such processing capability needs to be proven.

Table 5.4: Proportion of e-commerce transactions in total bitcoin transactions

Topology	**Fundchannel Time Cost(minutes)**	**channel count**	**Total Bitcoin Transactions**	**Proportion**
$Small_{(8,8,16)}$	28.23	60	5,865	0.01023340024
$Medium_{(8,16,32)}$	50.47	108	10,672	0.01011994002
$Large_{(8,84,168)}$	243.61	516	50,947	0.01012817241
$LargeHighCore_{(40,84,168)}$	640.66	1240	117,060	0.01059285836

We calculate the time cost to set up $Small_{(8,8,16)}$ topology. The time includes execution and confirmation time to transfer funds from a bitcoin wallet to a lightning wallet, as well as execution and confirmation time of fund channels. This comes to a total time of 28.23 minutes. In Table 5.4, the channel count is the number of channels connecting platform nodes. The total bitcoin transactions represents the number of transactions related to the setup that have been confirmed on the blockchain in 28.23 minutes. Proportion is the ratio of the number of e-commerce transactions to total bitcoin transactions. The network setup takes around 1% of the total on-chain transactions over the time period. The time our experiment took to setup the largest network confirms this.

Table 5.5: The number of channels required and Time Cost under an Amazon scale network

Metrics	**Topology with only one Core-Layer group**	**Add one more Core-Layer group**
Number of Channels	3,505,490	505,490
Time cost with full lightning network resources	10 days	1.5 days
Time cost with 1% lightning network resources	1000 days	150 days

Turning to the network scale we described in section 4.1, 3,505,490 payment channels are required to build an Amazon scale payment channel based network, including channels between Core and Edge nodes, channels between Core nodes, channels between Edge nodes, and channels between customers or merchants and Edge nodes. Each additional Core-Layer group will add 505,490 on-chain transactions to generate payment channels.

Table 5.5 shows the number of channels required and the time cost to build an Amazon size network. As we find in Table 5.4, if we limit ourselves to taking 1% of the bitcoin processing capability, it will take about 1000 days (2.7 years). This seems a long time. However, e-commerce networks generally introduce new capabilities from a small region to worldwide step by step. Therefore, the time cost may be acceptable. The Layer2 network can be maintained for a long time once it is established. Moreover, a platform node can fund multiple payment channels using a single bitcoin transaction [94]. A node can fund over 100 payment channels in a single transaction using a technique described in a Blockstream report from 2020 [95]. This can reduce the total number of on-chain transactions in our network and significantly reduce time costs.

5.4.4 Funds Locked in the Network

The payment channel network requires participants to deposit initial funds in channels to support transactions when they join the network. Figure 5.14 shows the change in the total amount of bitcoins locked in the payment channel network when each platform node locks between 0.01BTC and 0.19BTC on each side of its payment channels. We assume the amount deposited on each side of the channel is the same in Figure 5.9. In practice, busy nodes would need to trade-off a high balance against the need to invoke a re-balancing algorithm frequently.

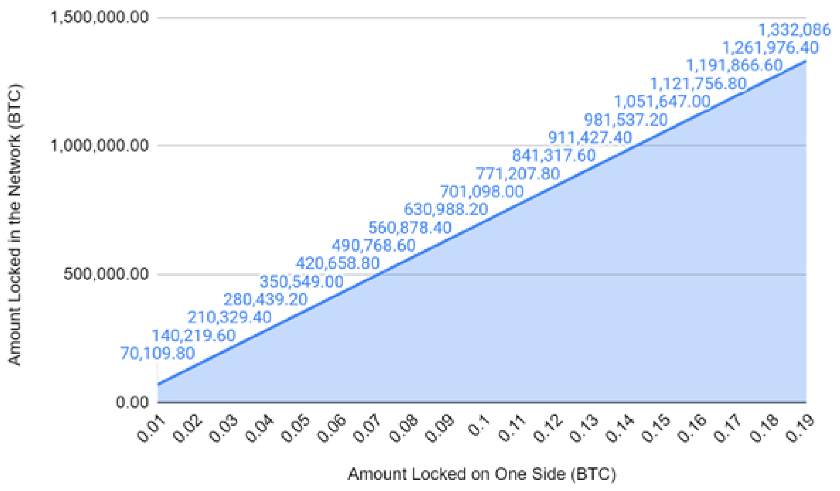


Figure 5.14: Total Amount of BTC Platforms Locked in the Payment Channel Network When Initial Deposits Locked on Each Side of a Channel Increases

In July 2022, one bitcoin has a price of approximately $24,000. This means if each platform node holds a deposits 0.1BTC on its channel, the total amount of bitcoins locked in the network is 701,098BTC which equals to $16.8 billion. This is a huge figure, but when compared with Amazon's 2021 revenue of $469 billion and their cash-on-hand of $33.36 billion [96], it is not infeasibly large. The network is likely to be rolled-out gradually and this may make the job easier.

5.4.5 Limitations

Above this section, we discussed the simulation results of our proposed architecture. It shows that the blockchain state-channel-based approach can be applied to mass e-commerce. However, there are also some limitations of our design and experiments. Our topology seeks to make a large blockchain Layer2 based network possible. But we only prove this in very limited simulation scenarios. So we still need more realistic considerations and environment deployment.

In the design, centralized elements like the k _id allocator have been introduced. Users may choose to tolerate centralized control of the network topology in return for high performance. However, the failure and attack of the centralized controller are problems that need to be solved as this could render the entire network unavailable. A distributed k _id allocation mechanism would be a direction for further research.

Rebalancing is a big issue that needs to be solved in the future. As we discussed in section 5.2.2, with the expansion of the network scale, the path discovery process will be more inclined to choose a path through Core-Layer platform nodes. This may cause the channel balance of the Core-Layer nodes to be quickly exhausted. This requires fast channel rebalancing. The channel balancing mechanism we proposed in Section 4.3 currently only exists at the theoretical level, and there is no corresponding practical analysis. In addition, in practice, there is not only one e-commerce company or platform in the network. In our vision, each platform can build a topology as in Section 4.1. At this point, a topology is a subnet. Different subnets can be connected through their Core-Layer platform nodes. This can help different platforms quickly create their own sub-topology and integrate it into the network. However, this complicates the problem of rebalancing. We still need to take more time on the rebalancing issues.

Reviewing this chapter, we design and implement experiments involving the routing algorithm and topology. Through the analysis of the experimental results, we conclude that the K-routing algorithm can significantly shorten the time while finding the shortest path. When the network scale increases, the maximum of hops does not increase. At the same time, transactions can be distributed evenly across each platform node which helps to avoid the trend of centralization. When nodes fail, our network maintains good availability. The number of on-chain transactions required for building a large-scale Layer2 network is also acceptable. Our design enables network construction in a relatively short period. Therefore, our proposed blockchain Layer2 based mass e-commerce network has application prospects.

Chapter 6

Conclusion

Our primary research question is whether blockchain layer2 technology can be used for payment in mass e-commerce. In this thesis, we describe the characteristics of current e-commerce and the problems faced in payment. The advantages of blockchain layer2, such as instant payment, trust without a third party, high privacy and low transaction fees, can solve the problem of e-commerce payment very well. We then discussed the currently proposed layer2 technologies and classified them. We find that State Channel based approaches, with almost no transaction delay, a high level of privacy, and high throughput, can better meet the requirements of e-commerce, which needs high transaction frequency, small transaction amounts and involves a large number of participants. In order to achieve the objectives in Chapter 1, we also review different kinds of topologies and routing algorithms. Topologies based on Tree structures and routing algorithms like Kademlia can effectively improve the reliability of the network, shorten the path length, scale the network and reduce latency.

For a suitable topology for a mass e-commerce scenario, we propose a 3-Level topology. Participants take on roles as the customer, platform node, or merchant. Customers send transactions in most cases, and merchant receive payments in the Leaf-Layer. The platform nodes are divided into Core-Layer and Edge-Layer nodes and act as intermediaries between customers and merchants. After experimental analysis, we found that the tree-based 2-Level-Platform topology can effectively reduce the number of payment channels, further reducing the number of on-chain transactions. At the same time, our topology can shorten the path length to 6 nodes to reach the most distant merchant with the shortest path. Compared to the current production Lightning Network's path length of 13 nodes, our topology significantly reduces transaction latency. Moreover, even if some nodes in the network are unavailable, the availability of the rest of the network is not affected.

Platform nodes are maintained by e-commerce companies. We also give formulas to calculate the number of platform nodes required for different sizes of e-commerce topology which helps e-commerce companies to setup and maintain networks.

We propose a K-routing algorithm which is a better choice to find paths from the source to destination nodes for mass e-commerce topology. We acquire metrics, including transactions per second, transaction fees, path length and transaction concentration to analyze the routing performance. We find that as the search space expands, the path length does not change significantly. The time cost to find this path is significantly lower than with the Dijkstra algorithm which widely implemented to production Lightning network. The K-routing algorithm can effectively narrow the path search range and improve search efficiency.

Whether our proposed Layer-2 based mass e-commerce network can support an Amazon scale network is another issue needs to be considered. Therefore, we calculate the platform nodes required in Core-Layer and Edge-Layer and the total number of payment channels. It only takes about 10 days for an Amazon-scale e-commerce network to complete the construction of the payment channel network if all network resources are used to do it. Once the network is established, customers can generate hundreds of millions of instant payments with tiny transaction fees. This addresses the problem of high transaction fees in traditional e-commerce.

Our research has important implications for the e-commerce industry. E-commerce platforms, customers and merchants can all benefit from it. E-commerce platforms can plan network scale and allocate hardware facilities. The reduced number of payment channels can significantly reduce the capital investment required of e-commerce companies when building their networks. The platform node group management enables the e-commerce platform to deal with node failures and balance the network traffic efficiently. Meanwhile, we reduce the path length and help customers save transaction fees. Merchants can collect payments in real-time, improving liquidity. Our proposed blockchain layer2 based mass e-commerce network has great application potential.

Bibliography

[1] Josh Gage. Amazon.com buyer fraud service gains scalability, cuts costs in half using aws. 2017. [online] Available: https://aws.amazon.com/solutions/case-studies/AmazonBuyerFraud/.

[2] Claire Greene, Joanna Stavins, et al. The 2016 and 2017 surveys of consumer payment choice: summary results. *Federal Reserve Bank of Atlanta Discussion Paper*, (18-3), 2018.

[3] Visa UK. Visa fact sheet. 2018. [online] Available: https://www.visa.co.uk/dam/VCOM/download/corporate/media/visanet-technology/aboutvisafactsheet.pdf.

[4] J. Göbel and A.E. Krzesinski. Increased block size and bitcoin blockchain dynamics. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6, 2017.

[5] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert Van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, Santa Clara, CA, March 2016. USENIX Association.

[6] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 123–140, New York, NY, USA, 2019. Association for Computing Machinery.

[7] A.K.M. Najmul Islam, Matti Mntymki, and Marja Turunen. Why do blockchains split? an actor-network perspective on bitcoin splits. *Technological Forecasting and Social Change*, 148:119743, 2019.

[8] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and

Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 123–140, New York, NY, USA, 2019. Association for Computing Machinery.

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

[10] Steven Millward. China's alipay just saw a record 1 billion transactions in a day. 2016. [online] Available: https://www.techinasia.com/alibaba-alipay-1-billion-transactions.

[11] Daniel Tolstoy, Emilia Rovira Nordman, Sara Meln Hnell, and Nurgl zbek. The development of international e-commerce in retail smes: An effectuation perspective. *Journal of World Business*, 56(3):101165, 2021.

[12] Amit Sharma, Hongrui Liu, and Hongwei Liu. Best seller rank (bsr) to sales: An empirical look at amazon.com. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 609–615, 2020.

[13] Ahmed Afif Monrat, Olov Schelén, and Karl Andersson. A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7:117134–117151, 2019.

[14] Xiaojun Liu, Wenbo Wang, Dusit Niyato, Narisa Zhao, and Ping Wang. Evolutionary game for mining pool selection in blockchain networks. *IEEE Wireless Communications Letters*, 7(5):760–763, 2018.

[15] Daniel Fullmer and A. Stephen Morse. Analysis of difficulty control in bitcoin and proof-of-work blockchains. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5988–5992, 2018.

[16] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[17] Chunchi Liu, Yinhao Xiao, Vishesh Javangula, Qin Hu, Shengling Wang, and Xiuzhen Cheng. Normachain: A blockchain-based normalized autonomous transaction settlement system for iot-based e-commerce. *IEEE Internet of Things Journal*, 6(3):4680–4693, 2019.

[18] Wenlin Xie, Wei Zhou, Lanju Kong, Xiangdong Zhang, Xinping Min, Zongshui Xiao, and Qingzhong Li. Ettf: A trusted trading framework using blockchain in e-commerce.

In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 612–617, 2018.

[19] Yinsheng Li, Shuai Xue, Xu Liang, and Xiao Zhu. I2i: A balanced ecommerce model with creditworthiness cloud. In *2017 IEEE 14th International Conference on e-Business Engineering (ICEBE)*, pages 150–158, 2017.

[20] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 201–226, Cham, 2020. Springer International Publishing.

[21] Bin Hu, Zongyang Zhang, Jianwei Liu, Yizhong Liu, Jiayuan Yin, Rongxing Lu, and Xiaodong Lin. A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems. *Patterns*, 2(2):100179, 2021.

[22] Keerthi Nelaturu, Han Du, and Duc-Phong Le. A review of blockchain in fintech: Taxonomy, challenges, and future directions. *Cryptography*, 6(2), 2022.

[23] Thaddeus Dryja Joseph Poon. The bitcoin lightning network: Scalable off-chain instant payments. 2016. [online] Available: https://lightning.network/lightning-network-paper.pdf.

[24] AliPay. Accessible digital payments for everyone. 2022. [online] Available: https://global.alipay.com/platform/site/ihome.

[25] Go Global. Alibaba in numbers – statistics and trends. 2015. [online] Available: https://www.go-globe.com/alibaba-statistics-trends/.

[26] Ipsos. Third-quarter mobile payment user research report. 2019. [online] Available: https://www.ipsos.com/zh-cn/yipusuoipsoszhongbang-disanjidudisanfangyidongzhifuyonghuyanjiubaogao.

[27] Massimo Bartoletti, Stefano Lande, Livio Pompianu, and Andrea Bracciali. A general framework for blockchain analytics. In *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, pages 1–6, 2017.

[28] Majid Khabbazian, Tejaswi Nadahalli, and Roger Wattenhofer. Outpost: A responsive lightweight watchtower. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, page 31–40, New York, NY, USA, 2019. Association for Computing Machinery.

[29] Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. eltoo: A simple layer2 protocol for bitcoin. *White paper: https://blockstream. com/eltoo. pdf*, 2018.

[30] The Raiden Network [Internet]. High speed asset transfers for ethereum. 2016 [cited 2022 July]. Available from: http://raiden.network/.

[31] Andrej Kos Matevž Pustišek, Anton Umek. Approaching the communication constraints of ethereum-based decentralized applications. *Sensors*, 19(11), 2019.

[32] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. Sprites: Payment channels that go faster than lightning. *CoRR*, abs/1702.05812, 2017.

[33] Rami Khalil and Arthur Gervais. Revive: Rebalancing off-blockchain payment networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 439–453, New York, NY, USA, 2017. ACM.

[34] C. Lin, M. Ma, X. Wang, Z. Liu, J. Chen, and S. Ji. Rapido: A Layer2 Payment System for Decentralized Currencies. *ArXiv e-prints*, August 2018.

[35] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18, Cham, 2015. Springer International Publishing.

[36] Wattenhofer R. Burchert C., Decker C. Scalable funding of bitcoin micropayment channel networks. *19th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2017.

[37] Liew Voon Kiong. *Investing in DeFi and NFT Projects on Alternative Blockchains: A Beginner's Guide to Investing in DeFi and NFT Projects on Alternative Blockchains other than Ethereum.* Liew Voon Kiong, 2021.

[38] Lydia D. Negka and Georgios P. Spathoulas. Blockchain state channels: A state of the art. *IEEE Access*, 9:160277–160298, 2021.

[39] Shermin Voshmgir. *Token Economy: How the Web3 reinvents the Internet*, volume 2. Token Kitchen, 2020.

[40] Jeremy Longley and Oliver Hopton. Funfair technology roadmap and discussion. *Funfair Technol., New York, NY, USA, White Paper*, page 6, 2017.

[41] Tom Close and Andrew Stewart. Forcemove: an n-party state channel protocol. *Magmo, White Paper*, 2018.

[42] Mo Dong, Qingkai Liang, Xiaozhou Li, and Junda Liu. Celer network: Bring internet scale to every blockchain. *arXiv preprint arXiv:1810.00037*, 2018.

[43] Jianli Luo, Yidan Hu, and Yanhu Bai. Bibliometric analysis of the blockchain scientific evolution: 2014–2020. *IEEE Access*, 9:120227–120246, 2021.

[44] Kyung-Hyune Rhee Siwan Noh, Sang Uk Shin. Pyros: A state channel-based access control system for a public blockchain network. *Security and Communication Networks*, 2020.

[45] M. Guo, J. Peng, J. Zhao, J. Quan, L. Wu, and T. Ye. Game analysis on sales return involving b2c e-commerce seller, buyer and platform. In *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*, pages 488–494, Sep. 2018.

[46] Amritraj Singh, Kelly Click, Reza M. Parizi, Qi Zhang, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications*, 149:102471, 2020.

[47] Carl Worley and Anthony Skjellum. Blockchain tradeoffs and challenges for current and emerging applications: generalization, fragmentation, sidechains, and scalability. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1582–1587. IEEE, 2018.

[48] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, pages 1–47, 2017. https://www.plasma.io/plasma.pdf.

[49] Michael Herbert Ziegler, Marcel Großmann, and Udo R Krieger. Integration of fog computing and blockchain technology using the plasma framework. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 120–123. IEEE, 2019.

[50] Vanita Jain, Arun Kumar Dubey, and Anurag Choubey. Plasma chain and blockchain

security model. In *Implementing and Leveraging Blockchain Programming*, pages 79–95. Springer, 2022.

[51] Jonas Nick, Andrew Poelstra, and Gregory Sanders. Liquid: A bitcoin sidechain. *Liquid white paper*, 2020. https://blockstream. com/assets/downloads/pdf/liquid-whitepaper. pdf.

[52] Roger Coll Aumatell. Analysis and development of blockchain rollups. Master's thesis, Universitat Politècnica de Catalunya, 2021.

[53] Craig Calcaterra and Wulf A Kaal. Decentralized finance (defi). *Decentralization-Technology's impact on organization and societal structure, degruyter publishers (2021)*, 2021. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3782216.

[54] U. Fiege, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 210–217, New York, NY, USA, 1987. Association for Computing Machinery.

[55] Frazer Chard and Cayo Fletcher-Smith. Blockchain scalability for smart contract systems using eutxo model. 2022. https://arxiv.org/abs/2202.00561.

[56] Liew Voon Kiong. *Investing in DeFi and NFT Projects on Alternative Blockchains: A Beginner's Guide to Investing in DeFi and NFT Projects on Alternative Blockchains other than Ethereum.* Liew Voon Kiong, 2021.

[57] Alex Wang Daniel Wang, Jay Zhou. Loopring: A decentralized token exchange protocol. *White Paper*, 2018. https://loopring.org/resources/en_whitepaper.pdf.

[58] Zhipeng Wang, Stefanos Chaliasos, Kaihua Qin, Liyi Zhou, Lifeng Gao, Pascal Berrang, Ben Livshits, and Arthur Gervais. On how zero-knowledge proof blockchain mixers improve, and worsen user privacy. *arXiv preprint arXiv:2201.09035*, 2022.

[59] Ceren Kocaoğullar, Arthur Gervais, and Benjamin Livshits. Towards private on-chain algorithmic trading. *arXiv preprint arXiv:2109.11270*, 2021.

[60] Zachary J Williamson. The aztec protocol introduction. 2018. [online] Available: https://github.com/AztecProtocol/AZTEC.

[61] David Schwartz Mikhail Wall. Ethereum scalability and zk-rollups. 2020. [online] Available: https://docs.hermez.io/.

[62] Francesco Bruschi, Manuel Tumiati, Vincenzo Rana, Mattia Bianchi, and Donatella Sciuto. A scalable decentralized system for fair token distribution and seamless users onboarding. *Blockchain: Research and Applications*, 3(1):100031, 2022.

[63] Fuel Labs. Introducing: Yul+ — a new low-level language for ethereum. 2020. [online] Available: https://fuel-labs.ghost.io/introducing-yul-a-new-low-level-language-for-ethereum/.

[64] Augusto Teixeira and Diego Nehab. The core of cartesi. *Cartesi Whitepaper*, 2018. [online] Available: https://coinpare.io/whitepaper/cartesi.pdf.

[65] Blaine Johnson Christina Davies. Enya launches public testnet of omgx - omg network's next-generation ethereum scaling solution. *Business Wire*, 2021. [online] Available: https://www.businesswire.com/news/home/20210511005664/en/Enya-Launches-Public-Testnet-of-OMGX—OMG-Network%E2%80%99s-Next-Generation-Ethereum-Scaling-Solution.

[66] William Hughes, Alejandro Russo, and Gerardo Schneider. Multicall: A transaction-batching interpreter for ethereum. In *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, BSCI '21, page 25–35, New York, NY, USA, 2021. Association for Computing Machinery.

[67] Tobias Schaffner. Scaling public blockchains. *A comprehensive analysis of optimistic and zero-knowledge rollups. University of Basel*, 2021. [online] Available: https://wwz.unibas.ch/fileadmin/user_upload/wwz/00_Professuren/Schaer_DLTFintech/Lehre/Tobias_Schaffner_Masterthesis.pdf.

[68] Maryam Mohsin. 10 amazon statistics you need to know in 2021. *OBERLO*, 2021. [online] Available: https://www.oberlo.com/blog/amazon-statistics.

[69] CIW Team. Alibaba profit down 75% in q4 2021. *China Internet Watch*, 2022. [online] Available: https://www.chinainternetwatch.com/31097/alibaba-quarterly/.

[70] DMR Homepage. Taobao statistics, user counts, facts news(2022). *Technology Statistics and Fun Facts*, 2022. [online] Available: https://expandedramblings.com/index.php/taobao-statistics/.

[71] Bo Song and Zhong-hua Zhao. Online holiday marketing's impact on purchase intention: China's double-11 shopping carnival. *Journal of Management and Humanity research*, 1:11–24, 2019.

[72] Brad Everman, Narmadha Rajendran, Xiaomin Li, and Ziliang Zong. Improving the cost efficiency of large-scale cloud systems running hybrid workloads-a case study of Alibaba cluster traces. *Sustainable Computing: Informatics and Systems*, 30:100528, 2021.

[73] Ashar Ahmad, Muhammad Saad, Joongheon Kim, DaeHun Nyang, and David Mohaisen. Performance evaluation of consensus protocols in blockchain-based audit systems. In *2021 International Conference on Information Networking (ICOIN)*, pages 654–656. IEEE, 2021.

[74] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, page 63–74, New York, NY, USA, 2008. Association for Computing Machinery.

[75] Gijs van Dam, Rabiah Abdul Kadir, Puteri NE Nohuddin, and Halimah Badioze Zaman. Improvements of the balance discovery attack on lightning network payment channels. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 313–323. Springer, 2020.

[76] A scala implementation of the lightning network. *GitHub Repository*, 2016. [online] Available: https://github.com/ACINQ/eclair.

[77] Lightning network daemon-lnd implementation. *GitHub Repository*, 2017. [online] Available: https://github.com/lightningnetwork/lnd.

[78] A highly modular bitcoin lightning library written in rust. *GitHub Repository*, 2019. [online] Available: https://github.com/rust-bitcoin/rust-lightning.

[79] Luuc Van Der Horst, Kim-Kwang Raymond Choo, and Nhien-An Le-Khac. Process memory investigation of the bitcoin clients electrum and bitcoin core. *IEEE Access*, 5:22385–22398, 2017.

[80] Cyril Grunspan, Gabriel Lehéricy, and Ricardo Pérez-Marco. Ant routing scalability for the lightning network. *arXiv preprint arXiv:2002.01374*, 2020.

[81] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 53–65, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[82] Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. Flare: An approach to routing in lightning network. *White Paper*, page 144, 2016.

[83] Philipp Hoenisch and Ingo Weber. AODV–Based routing for payment channel networks. In *Proceedings of the First International Conference on Blockchain, ICBC 2018*, pages 107–124, Cham, 2018. Springer International Publishing.

[84] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Silentwhispers: Enforcing security and privacy in decentralized credit networks. *Cryptology ePrint Archive*, 2016.

[85] 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.

[86] Peng Wang, Hong Xu, Xin Jin, and Tao Wang. Flash: efficient dynamic routing for offchain networks. pages 370–381, 2019.

[87] Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari, Dejun Yang, and Jian Tang. Coinexpress: A fast payment routing mechanism in blockchain-based payment channel networks. In *27th International Conference on Computer Communication and Networks, ICCCN 2018, Hangzhou, China, July 30 - August 2, 2018*, pages 1–9. IEEE, 2018.

[88] Elias Rohrer, Jann-Frederik Laß, and Florian Tschorsch. Towards a concurrent and distributed route selection for payment channel networks. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Hannes Hartenstein, and Jordi Herrera-Joancomartí, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 411–419, Cham, 2017. Springer International Publishing.

[89] Peng Wang Tao Wang. Traces and code for offchain routing-flash simulation result. *Github Repository*, 2019. https://github.com/NetX-lab/Offchain-routing-traces-and-code.

[90] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia Fanti, and Pramod Viswanath. Routing cryptocurrency with the spider network. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, HotNets '18, page 29–35, New York, NY, USA, 2018. Association for Computing Machinery.

[91] Rene Pickhardt and Mariusz Nowostawski. Imbalance measure and proactive channel rebalancing algorithm for the lightning network. *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5, 2020.

[92] MyIP.MS. Hosting info, websites ip databases. 2022. [online] Available: https://myip.ms/browse/states/1/countryID/IRL/txt/Leinster.

[93] Lightning network eccentricity. *BitcoinVisuals Homepage.* https://bitcoinvisuals.com/ln-eccentricity [Last Access: July 2022].

[94] Blockstream Homepage. Lightning network multifundchannel - command for establishing many lightning channels. https://lightning.readthedocs.io/lightning-multifundchannel.7.html [Last Access: July 2022].

[95] Blockstream Team. New release: c-lightning 0.9.1. *Medium*, 2020. https://medium.com/blockstream/new-release-c-lightning-0-9-1-fbe4040980d9.

[96] Daniela Coppola. Annual net income of amazon.com from 2004 to 2021. *Statista*, 2021. https://www.statista.com/statistics/266288/annual-et-income-of-amazoncom/.