Summer 6-15-2021

Off-chain Transaction Routing in Payment Channel Networks: A Machine Learning Approach

Heba Kadry
The American University in Cairo AUC, hebakadry@aucegypt.edu

The American University in Cairo
School of Sciences and Engineering
Robotics, Control and Smart Systems

Off-chain Transaction Routing in Payment Channel Networks: A Machine Learning Approach

By
Heba Ahmed Kadry El-Riedy

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Robotics, Control and Smart Systems (RCSS)

Under supervision of:
Dr. Yasser Gadallah
Professor and Chair, Department of Electronics and Communications Engineering

May, 2021

DEDICATION

This thesis is dedicated to my parents for their kindness, patience, sacrifice, and devotion and for their endless and unparalleled support towards fulfilling my academic goals.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to the thesis advisor, Dr. Yasser Gadallah, for his tremendous support in completing this research work. Every comment he made in the thesis built a solid and robust foundation in the thesis as well as my academic skills.

My utmost gratitude also goes to my academic advisor and the RCSS program director, Dr. Maki Habib, for his exerted efforts to facilitate and grant the financial assistance that was a broad base of support to complete my study in the program. In addition, I am so grateful for the time spent as a research assistant in the robotics labs that added to my academic experience.

My special and sincere thank you also goes to my dear colleague, Ahmed Fahim, for his continued moral support and encouragement throughout the years of working on my thesis. I remember when we spent hours discussing the challenges of the thesis or even the complicated coursework and how these discussions boost the confidence to stand and move forward towards achieving my academic potential.

I also would like to appreciate the valuable and fruitful discussion in the thesis defense with Dr. Karim Seddik and Dr. Amr El-Sherif, the internal and external examiners. Their constructive comments helped me add further clarifications to enhance the final thesis quality.

Finally and foremost, many thanks for a great deal of support and assistance that grant me the completion of this thesis work and my master's degree.

ABSTRACT

Blockchain is a foundational technology that has the potential to create new prospects for our economic and social systems. However, the scalability problem limits the capability to deliver a target throughput and latency, compared to the traditional financial systems, with increasing workload. Layer-two is a collective term for solutions designed to help solve the scalability by handling transactions off the main chain, also known as layer one. These solutions have the capability to achieve high throughput, fast settlement, and cost efficiency without sacrificing network security. For example, bidirectional payment channels are utilized to allow the execution of fast transactions between two parties, thus forming the so-called payment channel networks (PCNs). Consequently, an efficient routing protocol is needed to find the payment path from the sender to the receiver, with the lowest transaction fees. This routing protocol needs to consider, among other factors, the unexpected online/offline behavior of the constituent payment nodes as well as payment channel imbalance. This study proposes a novel machine learning-based routing technique for fully distributed and efficient off-chain transactions to be used within the PCNs. For this purpose, the effect of the offline nodes and channel imbalance on the payment channels network are modeled. The simulation results demonstrate a good tradeoff among success ratio, transaction fees, routing efficiency, transaction overhead, and transaction maintenance overhead as compared to other techniques that have been previously proposed for the same purpose.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

a	Action
dst	Transaction destination
H	Hash function
Q	Action-value
r	rewards
s	state
src	Transaction source
π	Policy

Subscripts

$*$	Optimal
a	Action
arg	Argument
d	Destination node
max	Maximum
min	Minimum
x	Source node
z	Neighbor node

Acronyms

AMP	Atomic multipath payments
AODV	Ad-hoc on-demand distance vector
BFS	Breadth-first search
BTC	Bitcoin network cryptocurrency
HTLC	Hashed time-lock contract
LN	Lightning Network
MANET	Mobile ad-hoc networks
MARL	Multi-agent reinforcement learning
ML	Machine learning
P2P	Peer-to-peer
PCN	Payment channel network
QoS	Quality of service
RL	Reinforcement learning
TXN	Transaction
XRP	Ripple network cryptocurrency

Chapter 1
Introduction

An economic dimension of peer-to-peer networks has been established by a previously unknown author, Satoshi Nakamoto, who introduced the Bitcoin concept in 2008 [1]. Blockchain, the underlying technology of Bitcoin, has widely spread in business applications. It is viewed as one of the most important technologies that will have massive impacts for years to come [2]. A blockchain consists of a distributed electronic database called a ledger, which is able to store any sort of data such as records, events, or transactions. This ledger continually grows as blocks linked to each other as chains are added, where each block has a limited storage size [3]. The verified block is added to the end of the growing chain of blocks, in which the data is continuously verified based on a consensus mechanism or a global agreement. In other words, the blockchain's primary value is the ability to deploy cryptographic mechanisms to reach consensus across unrelated parties in the ledger [4]. No centralized node or authority is required. Also, not necessarily all the nodes keep a complete copy of the entire database. This architecture allows people to trust the system's output without the need to trust any part of it.

1.1 The Blockchain Technology

Blockchain introduces database functionalities similar to distributed networks. It is often referred to as a "radical innovation" or general-purpose technology. It is a technology that can create subsequent innovation and productivity gains across multiple industries, similar to what the Internet achieved [4]. A blockchain is a type of database, called a ledger, where records are added in the form of blocks. Each block is chained to the previous one using a cryptographic signature. The real novelty of this technology is that it is more than just a database. It can set rules about a transaction (business logic) that are tied to the transaction itself. This contrasts with conventional databases, in which rules are often set at the whole database level or in the application, but not in the transaction [5].There are different ways to maintain the accuracy and integrity of a ledger, but they are broadly known as consensus. They represent several blockchain architecture options, as shown in Figure 1-1. Permissionless ledgers such as Bitcoin and Ethereum [6] have no single owner, and they cannot be owned. Anyone can contribute data to the ledger and possess an identical copy of the ledger. So, no actor can prevent adding a transaction to the ledger, which creates censorship resistance.

On the other hand, permissioned ledgers have one or many owners. The integrity of the ledger is maintained by trusted actors, such as government departments or banks. This process is much simpler than the consensus process used by permissionless ledgers. An example of such blockchains is the Ripple [7], which is a global financial transaction system.

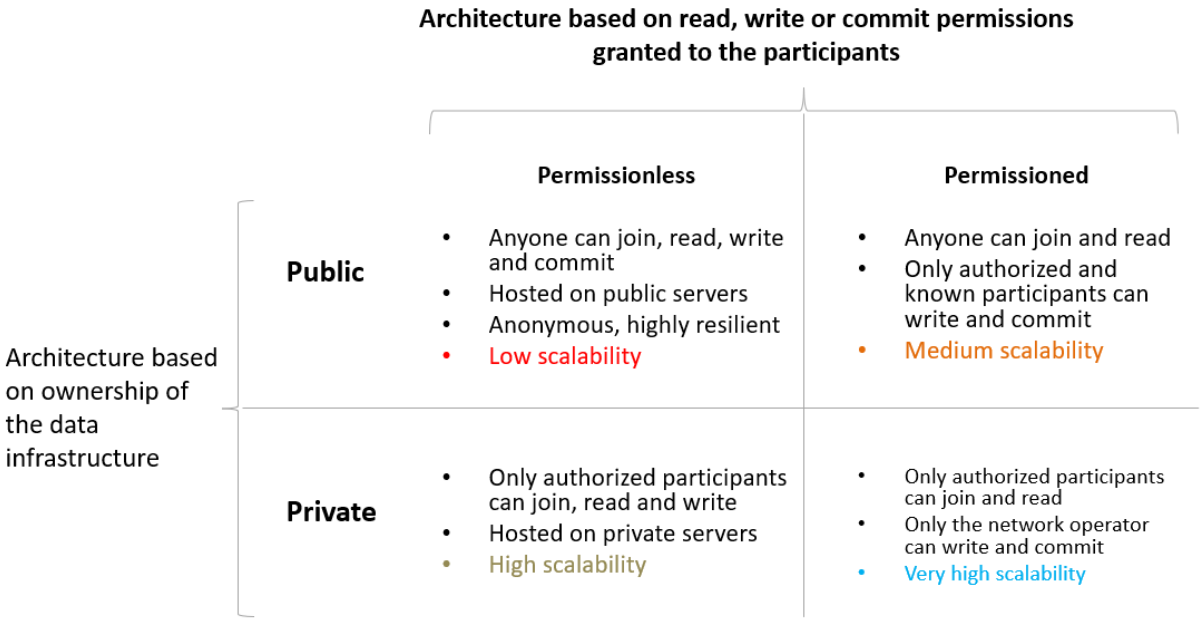


Figure 1-1. Blockchain-architecture options [8]

1.2 Blockchain Applications

Most of the blockchain applications are currently related to cryptocurrency, allowing the transfer of monetary values between parties. The blockchain was a priority topic at Davos, as a World Economic Forum. A survey suggested that 10 percent of global gross domestic product (GDP) will be stored in blockchains by 2027 [8]. Designed to be much more than a payment system, Ethereum was launched in 2014 as an open-source, public, blockchain-based distributed computing platform that provides a 'crypto-economically-secured' platform for the development of any kind of decentralized application. Therefore, blockchain applications go far beyond cryptocurrency. They are able to create transparency and fairness while saving businesses time and money. This technology is impacting various sectors in different ways that range from how contracts are enforced to making the government work more efficiently. The blockchain opportunities are enormous within different sectors, as summarized in the following [9].

1.2.1 Financial applications

Currently, blockchain technology is applied to various financial fields, including cross-border payments and financial assets settlement. An example of bank cooperation is the Global Payments Steering Group (GPSG), whose members involve Santander, Bank of America, and UniCredit. The cryptocurrency behind GPSG is XRP, created by Ripple [7]. The Ripple network has 300+ financial institutions, including banks and payment providers worldwide. Although Ripple is built upon an open-source, decentralized consensus protocol, the current deployment of Ripple is solely managed by Ripple Labs.

1.2.2 Internet of things

The growing interest and investments for implementing decentralized IoT platforms are mainly motivated by blockchain technology and its inherent characteristics [10]. The primary purpose is to deliver secure and auditable data exchange in environments with plenty of interconnected smart devices, such as wireless sensor networks.

1.2.3 Business and industrial applications

Blockchain has the capability to become a foundation of disruptive innovations in management through automating, improving, and optimizing business and industrial processes. For example, blockchain can increase transparency and accountability in supply chain networks, thus enabling more flexible value chains. Also, blockchain will significantly impact different industries, with the energy sector being at the top of the list. Nevertheless, the energy industry trading is different from the financial sector. The former requires online measurements and records for the physical product itself (e.g., oil, gas, electricity, nuclear power, solar power, wind power) for metering and billing of consumption. This cyber-physical collaboration can benefit from the Fourth Industrial Revolution applying the concepts of Industrial Internet of Things (IIoT) and machine-to-machine (M2M) communication. Hence, blockchain offers an opportunity to make use of the unpredictable nature of renewable energy. It also paves the way towards the transition to a decarbonized, decentralized, democratized, and resilient energy system [11] [12].

1.2.4 Privacy and security

Blockchain offers an opportunity to enhance the security aspects of big data. When combined with other efficient storage systems, blockchain is able to implement data mining methods. Another example is Namecoin [13], an open-source blockchain technology that

implements a decentralized version of the domain name system (DNS). The main benefits of a decentralized DNS approach are security, censorship resistance, efficiency, and privacy.

1.2.5 Integrity Verifications

The blockchain integrity verification applications store data related to the creation and lifetime of products or services. The potential applications are: i) provenance and counterfeit, ii) intellectual property (IP) management, and iii) insurance.

1.2.6 Governance

Blockchain-enabled applications could change the way governments at the local or state level operate by disintermediating transactions and record keeping. The safety, automation, and accountability, which blockchain proposes for handling public records, could hinder corruption and make government services more efficient.

1.2.7 Healthcare management

Various blockchain opportunities exist, for example, user-oriented medical research, sharing patients' medical data, automated health claims adjudication, public healthcare management, longitudinal healthcare records, online patient access, drug counterfeiting, clinical trial, and precision medicine.

1.3 Blockchain Challenges

Blockchain permissionless networks like Bitcoin and Ethereum face an inherent scalability limitation that restricts the rate at which the blockchain network can process transactions [14]. Solving the scalability problem will significantly help with blockchain mainstream adoption. Various solutions are proposed in the literature to scale blockchains without changing their trust assumptions, such as layer-one solutions and layer-two protocols [15]. Within the context of the blockchain, a layer-two protocol is implemented on top of the layer-one protocol, the blockchain or ledger. It is added to enable users to perform the so-called off-chain transactions through private and authenticated communication rather than to broadcast every single transaction on the (parent) blockchain [16].

Off-chain networks have witnessed fast development over recent years. Bitcoin's Lightning Network (LN) [14] and Ethereum's Raiden Network [17] are the most prominent

examples of this emergent area of research. Ripple [7] is a permissioned digital payment network using its own cryptocurrency called XRP as the primary transaction medium.

Transactions are executed off-chain without affecting the decentralization model, the privacy, or the trust features of the blockchain, with significantly lower transaction fees. For comparison, while the average on-chain bitcoin transaction fee may cost $25, the average fee on the Lightning (i.e. an off-chain layer on top of the Bitcoin) is a fraction of a cent [18]. To enable payments between any two nodes, payment channels form a network, called a payment channel network (PCN). In which payments can be routed over more than one hop. Thus, an efficient transaction routing protocol has a pivotal role in solving the scalability limitation of blockchain networks.

The main challenge that PCNs face is finding the best route of payment channels between the transaction source and destination with the lowest transaction fees. Also, both transient errors (insufficiently funded channels) and intransient errors (such as offline nodes) are identified as the main reasons for failing payments [19].

Thus, one of the unsolved issues with routing algorithms, and the LN itself, is that a significant number of nodes should be continuously online [20]. This requirement is not present in Bitcoin and virtually all centralized payment systems. However, PCNs provide a measurable incentive for nodes to be online, as an online node could receive transaction fees by routing payments through it.

According to the study in [19], the main reason for failing payments is a temporary channel failure. It is reported when a channel direction has temporarily insufficient available funds. This could be a result of previous transactions that have been locked through the channel. The error indicates the same payment may succeed through the channel at a later time. The following most common error was Unknown Next Peer. This error occurs because a hop along the intended payment route does not connect to the next node.

1.4 Problem Statement

In this work, the main objective is to overcome the blockchain limitations of scalability and high transaction fees using PCNs. We intend to achieve this objective by proposing an efficient and effective transaction routing protocol in PCNs. The proposed protocol has to consider the fast dynamic nature of PCN's. In addition, it should be able to efficiently handle

frequent errors that may result in repetitive routing failures. These errors occur as a result of the lack of online or cooperative nodes needed to form the path from transaction source to destination. Also, a payment channel along the payment path may not have sufficient funds to route the payment, which is another source of frequent routing failure.

1.5 Thesis Objectives and Contributions

The thesis objective aims at transaction route-finding and selection in a PCN that meets certain constraints. A machine learning-based technique is proposed to suggest the possible routes of the payment channels to carry the transactions from the sender to the destination with the lowest transaction fees. It aims to “learn” and “memorize” node and payment channel activities and use it to predict a future route suggestion with the lowest transaction fees as a function of time. Besides, a model is proposed that assesses offline nodes' impact on the routing protocol performance in PCNs.

As such, the objectives of this research can be summarized as follows.

1. Propose an effective and efficient transaction routing protocol in PCNs.
2. Assess the impact of off-chain network dynamics and errors on payment routing protocols.

Consequently, the main contributions of this research can be stated as:

Devising a novel machine learning-based routing protocol proposed for off-chain transaction routing in the payment channel networks. The proposed protocol will predict the best route in the future in case of insufficiently funded channels and offline nodes. Transient and intransient errors that result in payment routing failure are modeled. To the best of our knowledge, this is the first work that proposes a machine learning-based technique in transaction routing in PCN. Also, this is the first work that investigates and evaluates the routing problem in PCNs considering the dynamics in terms of the involved nodes status. These dynamics cover the offline nodes, identified as one of the main reasons for failing payments.

1.6 Thesis Structure

The rest of this thesis is organized as follows. In chapter 2, we present the necessary background information regarding blockchain. Furthermore, the fundamentals of payment channel networks are discussed in this chapter. We also review some of the proposed routing protocols that are relevant to our work. The recent findings of the literature studies that address machine learning applications in packet routing are also covered in this chapter. Chapter 3

presents the payment channel model and the routing problem formulation using a machine learning-based technique. Chapter 4 describes the proposed algorithm and its detailed functionality. In chapter 5, the results of the simulation experiments that validate and evaluate the proposed technique are presented. Finally, this study is concluded in chapter 6, along with providing our ideas for future work.

Chapter 2
Related Work

2.1 Introduction

In this chapter, we present the background necessary for this study along with the previous research efforts concerning the payment channel network routing. First, blockchain technology is introduced, highlighting the main characteristics of this technology. We address significant challenges that hinder the wide adoption of blockchain-based cryptocurrency. Then, we describe layers of the blockchain network and introduce the concept of off-chain layers. We discuss the solution to the blockchain scalability problem using an off-chain layer, forming the concept of payment channel networks. We highlight the significance of routing in efficient off-chain network performance. We also discuss the related work in payment channel network routing. Finally, we investigate the background of using machine learning in communication network routing problems.

2.2 Blockchain and Cryptocurrency

Since the emergence of an economic dimension of P2P networks by an anonymous author called Satoshi Nakamoto in the Bitcoin paper [1] published in 2008, the blockchain applications, the underlying technology beyond Bitcoin, have widely spread in business applications. It is viewed as one of the most critical technologies that will have massive impacts for years to come [2].

A blockchain consists of a distributed electronic database called a ledger that can store any data such as records, events, or transactions. This ledger continually grows as blocks linked to each other as a chain, where each block has a limited storage size. The data stored in each block are verified or confirmed and connected to the previous block, as shown in Figure 2-1, using an algorithm that generates a unique hash function, *H*, which includes a series of letters and numbers. In case any piece of data is altered due to transmission error, for example, the correct hash function will not be generated, and therefore an error will be reported. The hash can easily generate a hash from any input. However, it is challenging to generate the input from the hash. Additionally, two different inputs cannot generate the same hash.

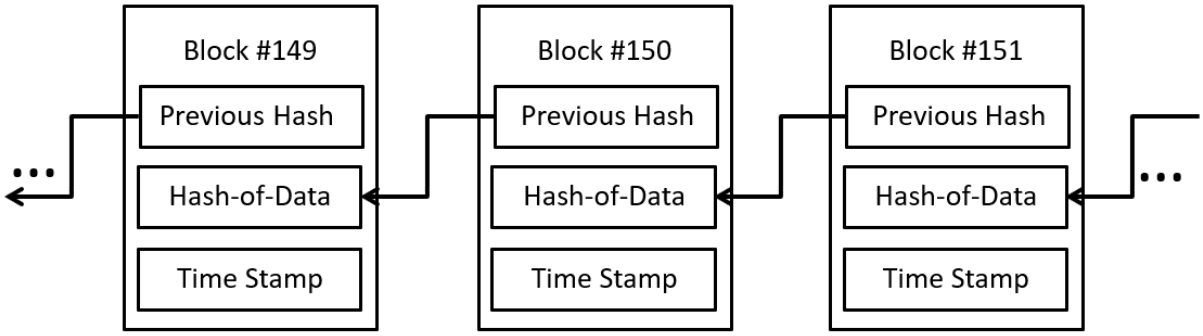


Figure 2-1. A blockchain is a linked list built with hash pointers instead of pointers [3]

Then, the verified block is added to the end of the growing blockchain, in which the data is continuously verified based on a consensus mechanism or a global agreement. In this process, called mining, new transaction verifications are performed and recorded on the global ledger. It is performed by the blockchain members, called miners, who render a competitive computing power service to the blockchain network by solving a hash function and get the chance to be rewarded accordingly. Miners also secure the blockchain transactions spending the same amount of Bitcoin (BTC) more than once, known as a double-spend. Thus, no centralized node or authority is required. Also, not necessarily all the nodes keep a complete copy of the entire database. This architecture allows people to trust the system's output without the need to trust any part of it [21] [22]. An example of some of the Bitcoin contents is shown in Figure 2-2.

Block 485963

Summary	
Number of Transactions	2055
Output Total	4,805.72139485 BTC ($298,374,073.73)
Transaction Fees	1.05055103 BTC ($65,225.83)
Height	485963
Timestamp	2017-09-19 04:11
Size	999,347 bytes
Miner	BTC.TOP
Hashes	
Hash	00000000000000000013942c4215cd92306bbce769cfcb349d0b42f031c994eb
Previous Block	00000000000000000004a5b64638b5d96d367a6d4e0a435fd460f972f1fb8f56b
Next Block(s)	00000000000000000006a36f741f47f7522d23809b97d2e24345488b5fffb0ea9

Figure 2-2. Some details of a Bitcoin block [23]

The hardware and bandwidth limitations impose constraints on the number of transactions per second of the distributed ledger of a blockchain, resulting in a scalability problem. Bitcoin and Ethereum [24] are prominent examples where the transactions per second are around seven for Bitcoin and fifteen for Ethereum [6]. On the other hand, the VISA system can support 47,000 transactions per second. Increasing the block size to handle a similar number of transactions as the VISA system will require a block capacity of eight gigabytes, which imposes large resource burdens on regular user computers [14]. Therefore, the number of miners will significantly decrease, which mandates relying on a central trusted authority with the power of validation. Thus, the primary privilege of the blockchain system of trustless trust would be lost. Moreover, micropayments (i.e. payments of less than a few cents) are considered impractical on the bitcoin network due to the fixed transaction fees and the minimum output size of a transaction.

Scalability is considered one of the critical challenges of blockchains, described as the scalability trilemma, a term coined by Vitalik Buterin, the co-founder of Ethereum [25]. The trilemma claims that blockchain systems can at most only have two of the three properties: decentralization, scalability, and security. Decentralization is the core and the nature of blockchain; security is essential, while the main challenge of scalability is finding a way to achieve all three at the base layer.

Various solutions are proposed in the literature to scale blockchains without changing their trust assumptions, such as layer-one solutions and layer-two protocols [15]. Within the context of blockchains, a layer-two protocol is implemented on top of the layer-one protocol, the blockchain or ledger. It is added to enable users to perform the so-called off-chain transactions through private and authenticated communications as opposed to broadcasting every single transaction on the (parent) blockchain [16].

2.3 The Off-chain Layer

In the literature, four layers are introduced to identify the blockchain components relevant to layer-two [16]. They are depicted in Figure 2-3 and can be summarized as follows:

- *Hardware Layer*: is considered a Trusted Execution Environment (TEE) and substitutes the need for a blockchain clock with a trusted hardware assumption. TEEs can execute sensitive or security-critical application code, tamper-proof from the operating system or other higher-privileged software.
- *Network Layer*: also called layer-zero, is typically a peer-to-peer layer on which blockchain nodes exchange information asynchronously. This layer performance is

crucial to the scalability, security, and privacy of a blockchain. Moreover, layer-zero significantly affects transaction throughput and provides stronger resilience against malicious actors.

- *Blockchain Layer*: also called layer-one, is an immutable append-only chain of blocks that accumulates transactions from parties in a network for public verifiability. A transaction, which reflects an update of the blockchain state, can exchange digital assets between parties or invoke an application (i.e., smart contract).
- *Off-chain Layer*: also called layer-two, allows transactions between users by exchanging authenticated messages through a medium tethered to a layer-one blockchain, but it is not part of it. Authenticated contentions are sent to the parent-chain only in case of a dispute. Hence, the parent-chain judges the dispute and issues the outcome. The security and trust properties of a layer-two protocol are inherited from the consensus algorithm of the parent-chain.

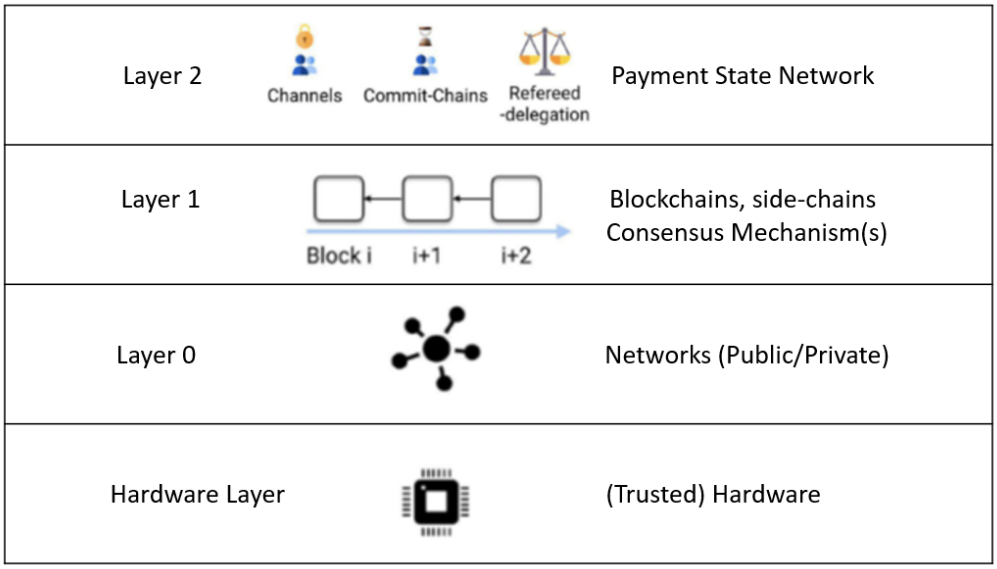


Figure 2-3. Proposed blockchain layers [16]

Different types of layer-two protocols exist, e.g. payment channel designs [26][27]. However, Bitcoin's Lightning Network [14] and Ethereum's Raiden Network [17] are the most prominent examples of this emergent area of research [28][29]. Despite having their own working mechanisms and particularities, both solutions are striving to provide increased throughput to blockchain systems. Bitcoin and Euthereum are the largest cryptocurrencies by market capitalization, respectively, according to CoinMarketCap [30]. This work will focus on the Lightning network as a prominent example of layer-two

protocols and payment channel networks. As of this writing, the Lighting network has 17,439 nodes and 38,482 edges [31].

2.3.1 Payment Channel Networks

A payment channel (aka micropayment channel or transaction channel) generates a financial relationship between two parties that record an updated current balance in a trustless manner and it gets communicated and exchanged off-chain [14]. In payment channel networks (PCNs), a payment channel can be viewed as a temporary joint account between two users or more [32]. The balance is divided among the owners, and the share can be modified upon all parties' agreement. Pre-set rules govern the agreement, e.g. a smart contract, allowing the involved parties to consent to channel updates by exchanging authenticated state transitions off-chain [16]. The channel capacity identifies the total deposit, limiting the maximum amount of payment that can be transferred via this channel.

Payment channels initially emerged to support fast unidirectional payments [33]. Then, Poon et al. proposed the Lightning Network (LN) that supports bidirectional payments [14]. If two parties want to execute a transaction and do not share a payment channel, the payment has to be performed via a series of transactions in different channels, composing a payment path. Several measures have to be undertaken to guarantee to send funds via multiple intermediaries in a network without the risk of intermediary theft of funds. The concept of using staggered timeouts, Hashed Time-lock Contract (HTLC) [34], was integrated into the Lightning Network [14] to construct secure transfers using a network of channels across multiple hops to the final destination. Typically, nodes collect fees for forwarding transactions, thus creating incentives. Basically, three cases exist where the transactions are broadcasted to the blockchain:

i) opening a payment channel,

ii) closing a payment channel, and,

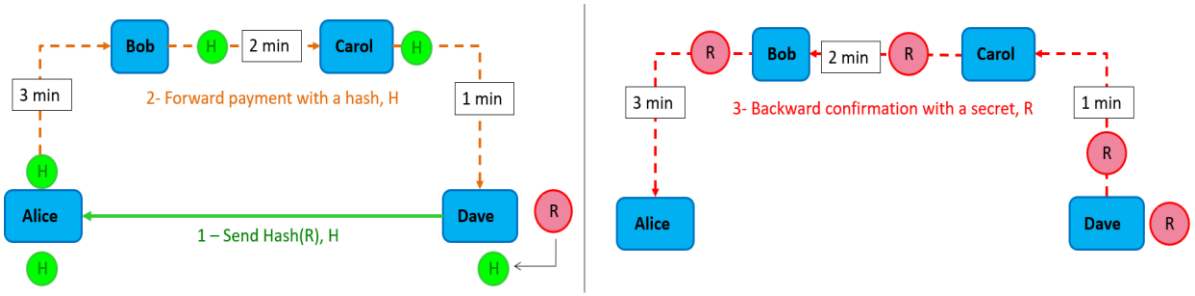
iii) a dispute between the two parties.

It is not necessary to open a channel in case the user will pay infrequently. Anyone may pay the money indirectly by transferring the money to intermediate parties till reaching the destination. In this process, the route is selected to minimize the number of intermediates and the associated fees.

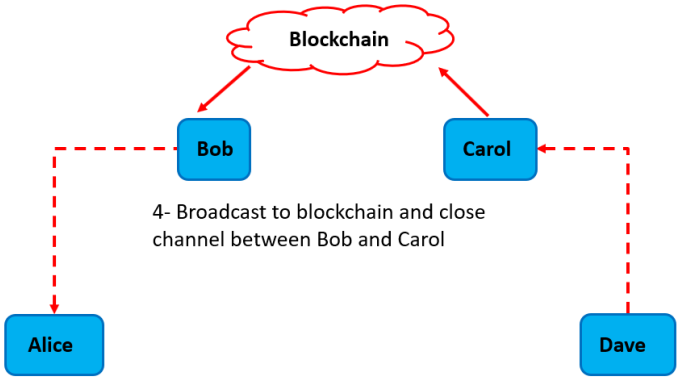
To illustrate this process of indirect payment through payment channels, we use the following example. Assume that Alice wishes to send 0.1 BTC to Dave, as shown in Figure 2-4 (a). She locates a route through Bob and Carol and requests Dave to send her Hash (R), H, to use for this payment, where R is a secret random value generated by Dave. Alice then creates

a contract, HTLC, with Bob that contains *H* and an expiry time of three minutes, for instance. Bob cannot spend the payment without providing *R* to Alice within three minutes. Bob then creates a contract with Carol the same way with an expiry of two minutes. That is, each node along the path should forward the payment the same way but with a decremented expiration time till the destination.

Upon reception, Dave will give the secret value, *R*, to Carol so that she can settle the payment and update the balance by the deadline instead of broadcasting it to the blockchain, as shown in Figure 2-4 (b). All parties along the path know that the disclosure of *R* will allow the disclosing party to pull funds, as agreed in the HTLC. This procedure then occurs step-by-step back to Alice. The whole process occurs off-chain, and nothing is broadcast to the blockchain when all parties are cooperative. In case a party, e.g. Bob, disconnects and refuses to update the balance, the counterparty, e.g. Carol, will be responsible for broadcasting this situation to the public blockchain, which will judge the situation and settle the failed non-responsive channel state, as shown in Figure 2-4 (c).



(a) Payment over the Lightning Network using HTLCs (b) Settlement of HTLCs



(c) Only the non-responsive channels get broadcast on the blockchain, all others are settled off-chain

Figure 2-4. A path of payment channels [14]

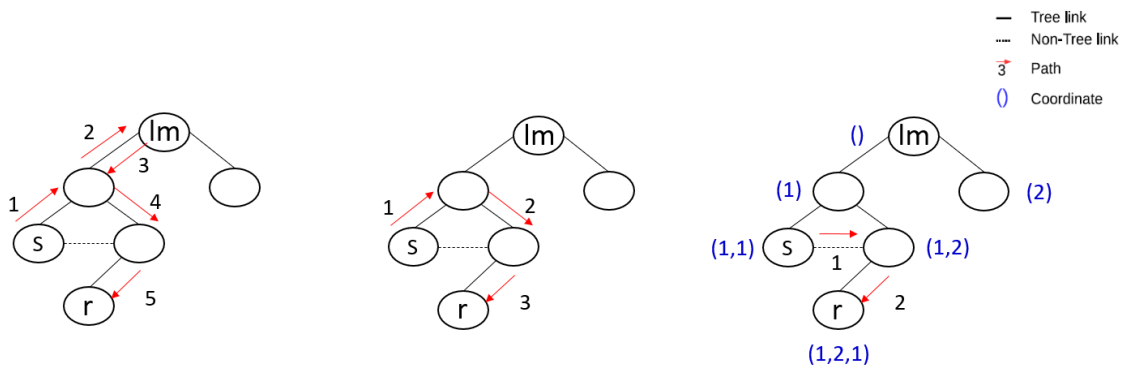
2.4 Off-chain Transaction Routing

Off-chain transaction routing is needed to achieve a successful payment with the minimum intervention of blockchain. Thus, efficient routing in PCNs is crucial to achieving their intended goal of solving the scalability limitations of blockchains. Though the routing problem is well-examined in data transmission, it experiences different challenges when applied to PCNs due to the diverse nature between both types [16]. In particular, node links and capacities are not considered private in computer networks. Moreover, routing of data does not change the state of the communication links.

On the other hand, a successfully executed transaction changes the state of the payment channel. The balance is updated after each transaction. What makes the situation more challenging is that the channel could be depleted in one direction. This imbalance may cause the PCN to become locked as no paths are found with funds in the needed direction, thus, making PCNs more dynamic than computer networks. Nodes may take actions to increase their balance in a payment channel, at the expense of decreasing their balance in another channel, to keep routing payments. This process is called rebalancing. Different rebalancing strategies are implemented in the literature [35][36] to help maintain efficient PCN transaction transfer.

Flare [20] is one of the earliest efforts to study the routing problem in payment channel networks (PCNs) in a decentralized manner. It is a beacon-based routing technique for LNs using the same techniques implemented in mobile ad-hoc networks (MANET). Each node reveals all payment channel capacities to a predefined number of nearest neighbors. It is characterized to have longer routes, higher latencies, and communication overheads. However, the most significant limitation of Flare is its inability to support topology changes [16].

Landmark-based routing was proposed in SilentWhispers [37], in which dedicated nodes, called landmarks, control the routing process and determine the path from a source to a destination such that they are part of the route (i.e. landmark-centered), as shown in Figure 2-5 (a). Any route has to go through the landmark even if the sender and the receiver are on the same branch, which leads to longer routes. The evaluation of SilentWhispers on a real-world dataset reveals low effectiveness and moderate latencies [28].



(a) Landmark-centered (b) Tree only (b) Embedding-based

Figure 2-5. Examples of spanning-tree routing schemes for landmark *lm*, sender *s*, receiver *r* [28]

Embedded-based, also known as distance-based, routing was extended in SpeedyMurmurs [28] by assigning coordinates to each node in the network referenced to a set of landmarks, as shown in Figure 2-5 (c). Distance-based routing suggests the shortest path by learning a vector embedding. In particular, vector embedding relies on a coordinate system, coordinates of nodes are dictated by the root node from a spanning tree. In order to initialize an embedding-based routing system, a spanning tree has to be created. So, nodes close in network hop distance are also close in embedded space or such a coordinate system. Upon transaction execution, nodes locally select the next hop in a payment path by considering all neighboring links with sufficient balances. Then, nodes choose the channel to the node with the coordinate closest to the recipient's coordinate, which is not necessarily part of the spanning tree. If a channel opens or closes, only descendants in the spanning tree have to modify their coordinates, which typically results in an overhead logarithmically scaled to the network size. Nevertheless, SpeedyMurmurs does not consider balances, which results in low effectiveness due to insufficient balances in a dynamic PCN [16].

The Spider Network [38] uses a modified implementation of source routing [39], in which the sender specifies the complete route to the destination. The modifications involve actively accounting for the cost of channel imbalance by preferring routes that rebalance channels. The preferred paths are categorized after applying a decentralized congestion control algorithm to determine the rate to send the so-called transaction units for different payments. Payments are split into transaction-units routed independently on multiple paths over time, similar to packet-switched networks. So, the risk that there is an insufficient balance in the payment direction is minimized. However, this comes at the expense of higher computational

overhead. Besides, the cost-effectiveness of the selected paths is worsened since transaction fees are charged for each path [16].

Maximum-flow algorithm [40], implementing the Ford–Fulkerson method [41], is the base for several studies for off-chain routing in [42], [32], [29], [43]. In graph theory, the maximum flow problem is to route as much flow as possible from the source to the sink. The flow from the source to the sink is initialized to zero at all edges. Then, the algorithm searches for any existing augmenting path. An augmenting path is a simple path from a source to a sink which does not include any cycles and that passes only through positive weighted edges. Upon finding an augmenting path, the current maximum flow is updated to add the flow of this augmenting path. Then, the current capacity of all the edges across the augmenting path are updated to exclude the augmenting path flow resulting in the so-called residual graph. A residual network graph indicates how much more of the flow is allowed in each edge in the network graph. If there are no augmenting paths possible from the source to the sink, then the flow has reached the maximum.

Although max-flow routing is a baseline technique in terms of throughput and transaction success rate, it has a high overhead of communication and latency [44] that is not practical for an actual PCN size. For instance, EdmondsKarp algorithm [45] is a special algorithm of the Ford–Fulkerson method, finding augmenting paths with breadth-first search (BFS) [46]. In BFS, the source node expands the shallowest nodes first of the graph to find the destination and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. The complexity of EdmondsKarp algorithm is $O(VE^2)$ per new transaction.

In Flash [29], payments are differentiated based on a threshold payment value (i.e. a percentage of the whole transaction values). For high transaction values, aka elephant payments, Flash uses a modified version of the EdmondsKarp algorithm that finds the maximum flow using a predefined maximum number of paths, *k*. As the algorithm executes EdmondsKarp algorithm for *k* paths, it has high latency [16]. For low transaction values, aka mice payments, the algorithm uses local routing tables information to do the routing thus avoiding executing an expensive flow algorithm. However, inferring the size of payments, thus the threshold value, is hard in real-world networks [47].

Ad-hoc on-demand distance vector (AODV) routing is proposed in [48] for payment channel networks. Each node maintains a routing table where updates are reflected only when searching for a route. The results indicate high effectiveness for successful path discovery and cost. However, the communication overhead is considerably high when compared to the simulated small graphs. So, scalability in a real-world dataset is unlikely [16].

The execution of the transaction through multiple paths can be achieved in PCNs. This possibility makes use of the mechanism of Atomic Multipath Payments (AMP) proposed for the Lightning network in [49]. The feature is implemented using HTLCs to allow the transaction to split over multiple paths. At the same time, the receiver should wait for the total intended payment prior to settling any of the partial payments. This implies that the transaction should either succeed or fail in its entirety. AMP eliminates the restriction to a single path from sender to receiver with sufficient directional capacity. It reduces the pressure to have larger channels in order to guarantee higher payment flows. Prior routing algorithms Spider Network [38] and Flash [29] consider this mechanism and propose multi-path payments.

To our knowledge, no routing algorithm that fulfills all desired criteria of scalability, effectiveness, and efficiency in real-world PCN dynamic settings [16].

2.5 Machine Learning for Networking

Machine learning (ML) techniques have been applied in different fields. They can not only be used to aid critical analysis of data, called data mining (e.g., features, variables, probability distribution, and transformation), but they also go beyond data mining to predict future events. Research interests have recently increased to use the power of ML for networking. Different techniques have been proposed to tackle various computer networks' problems, such as traffic prediction, congestion control, routing and classification, resource and fault management, QoS management, and network security [50]. Network traffic analysis and prediction is a proactive approach that aims to maintain secure and reliable network communications. The traffic analysis targets avoiding network congestion by allocating the network resources according to the predicted traffic.

ML techniques can be broadly classified into four main categories [50]:

i) Supervised, where labeled training datasets are used to create models to identify patterns or behaviors according to the training dataset. This technique is used in

classification and regression, where discrete and continuous quantities are predicted, respectively.

ii) Unsupervised, where unlabeled training datasets are used to create models to distinguish between patterns (e.g., clustering).

iii) Semi-supervised, where missing labels in the training dataset is inevitable.

iv) Reinforcement learning (RL), where an agent interacts with the environment. Thus, it learns and gains experience by exploration rather than having examples.

Different approaches of machine learning have been proposed to solve routing in wireless sensor networks. Such approaches are usually based on RL, swarm intelligence or neural networks. Nevertheless, these algorithms have different properties, which need to be considered when selecting the application. For instance, ant-based routing is a very flexible algorithm, but it generates a lot of additional traffic. This is because of the forward and backward ants that move through the network. In a WSN scenario, this overhead has to be carefully addressed and compared to that in other protocols.

On the other hand, algorithms that need an offline learning phase, such as offline neural networks, cannot cope with changing properties of the network. As they require very high costs of gathering the relevant data, calculating the routing tree and then distributing the tree roles to the nodes. RL algorithms have proved to work very well for routing and can be implemented at nearly no additional costs. They should be the first choice when looking for a flexible and low-cost routing paradigm [51]. Moreover, some RL proposed techniques feature a good scalability property that implies the solutions' capability to route data in large-scale networks. In contrast, swarm intelligence or neural network techniques feature limited scalability [52].

2.6 Reinforcement Learning in Packet Routing

Reinforcement learning (RL) is a technique where an autonomous agent seeks to associate actions with system states in a trial-and-error manner. The outcome of an action is observed as a reinforcement. Reinforcement learning aims to maximize the total rewards (i.e. reinforcements) or minimize the penalties that an agent receives over time by selecting the optimal action. Reinforcement learning problems are typically modeled as Markov decision processes (MDPs) that consist of a set of states S, a set of actions A, and a reinforcement function $R(s, a)$, as shown in Figure 2-6.

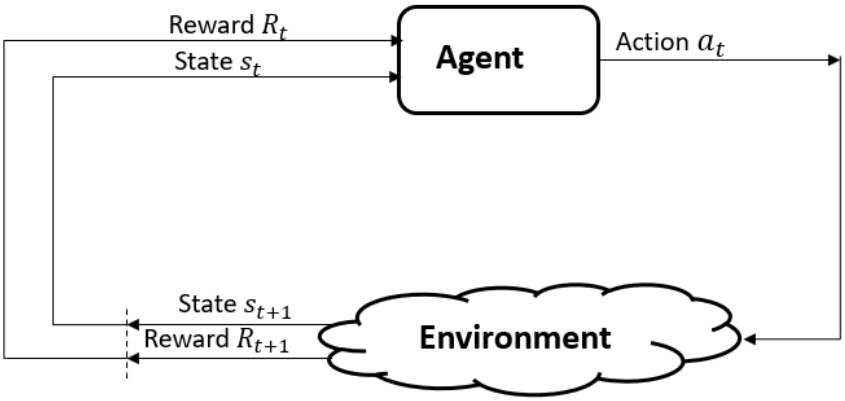


Figure 2-6. The agent-environment interaction in a Markov decision process [53]

RL strategies can be classified into two main categories, namely, model-free (e.g. Q-learning) and model-based learning methods [54]. Using a model of the environment, an agent can predict how the environment responds to its actions. Given a state and an action, a model produces a prediction of the resultant next state and next reward. In case the model is stochastic, there are several possible next states and next rewards, each with a probability of occurrence. Some models produce a description of all possibilities and their probabilities, called distribution models. Other models produce just one of the possibilities, sampled according to the probabilities, called sample models. Given a starting state and an action, a sample model produces a possible transition and a distribution model generates all possible transitions weighted by their probabilities of occurrence [53]. The model consists of an explicit knowledge of the state transition probability function $T(s, a, s')$ and the reinforcement function $R(s, a)$ [54].

Model-based learning methods build an internal model of the environment, calculate the optimal policy and use it to derive the agent. On the other hand, model-free methods do not build an explicit $T(s, a, s')$ and $R(s, a)$ model, but the agent learns directly from experience. Model-based methods are guaranteed to converge faster to the optimal policy as they make use of the data stored in the internal model. However, they are less prevalent in RL due to slower execution time and increased storage size. Nevertheless, in distributed systems where gaining real-time experience is costly, model-based techniques are superior to model-free approaches as the former make use of the gathered data [55]. The upcoming sections will address the two RL approaches as used in communication routing problems.

2.6.1 Model-free Reinforcement Learning in Packet Routing

Q-routing is a distributed adaptive traffic control scheme based on Q-learning (a reinforcement learning technique) to route traffic in dynamically changing traffic and topology networks [56]. In Q-routing, each node has its own controller that takes the routing decisions, while Q-learning depends on the network's global information for solving multistage decision problems. Q-Routing is the online version of the Bellmann-Ford algorithm [57]. An action-value function (i.e. Q-value) estimates the utility for performing a specific action for a particular state. For Q-routing, the utility represents the best link with the lowest $Q_x\,(d, y)$ (i.e., the time estimate to deliver a packet to node *d* from node *x* via a neighbor node *y*) that is deterministically updated as follows.

$$Q_x(d, y) \leftarrow Q_x(d, y) + \alpha\, \Delta Q \tag{2-1}$$

$$\Delta Q = \Big(transmission\ delay + queueing\ delay\ at\ y + min_z Q_y(d, z) - Q_x(d, y)\Big), \tag{2-2}$$

where ΔQ is the difference between the new and the old estimates, α is the learning rate, and $Q_y(d, z)$ is the time estimate to deliver a packet to node *d* from node *y* via a neighbor node *z*. In case the estimated time in the link has increased due to a temporary load condition, this link will not be used again until the other link in the same node has worse estimated time values.

A memory-based Q-learning algorithm was proposed in [58] to memorize the best experiences (Q-values) and their rate of variations (i.e. recovery rate) to be used to predict future traffic trends thus increasing the learning speed. During normal network load, the shortest path is the optimum, while in case of congestion, alternative routes ought to be utilized. As the load decreases in the shortest paths, they recover and get back to be optimum paths. Special packets are occasionally sent using these paths as a controlled exploration activity called probing to recognize this change. The probing frequency is critical since it may increase the congestion in the already congested paths. Predictive Q-Routing (PQ-Routing) [58] is identical to Q-learning in how the Q-function is updated except for the routing policy (2-4). The recovery rate is used for better Q-value estimates rather than only the current Q-values. The recovery rate changes with time and depends on network traffic and the probability of link or node failure. This recovery rate is memorized to be utilized for future routing plans.

$$\Delta t = \text{current time} - U_x(d, y), \tag{2-3}$$

$$Q'_x(d, y) = \max\big(Q_x(d, y) + \Delta t\, R_x(d, y), B_x(d, y)\big), \tag{2-4}$$

where $B_x(d, y)$ is the estimated delivery time from node *x* to node *d* via a neighboring node *y*, $R_x(d, y)$ is the recovery rate for path from node *x* to node *d* via a neighboring node *y*, and $U_x(d, y)$ is the last update time for path from node *x* to node *d* via a neighbor node *y*.

In multi-agent RL, agents collaborate to achieve shared objectives in a challenging learning environment due to huge state space, limited training examples. In opaque transitions, agents are not able to track state transitions subsequent to performing an action. The earliest work that proposes multi-agent RL in network routing problems is Team-partitioned opaque-transition (TPOT) RL in [59].

The update mechanism in (TPOT) RL in (2-5) is similar to Q-routing in (2-1). Nevertheless, the fundamental difference between TPOT-RL and Q-routing is that the immediate rewards in (TPOT) RL is calculated based on the transmission of packet arrival time from the source node to the destination. Hence, immediate reward r in (2-5) is entirely goal-oriented after the packet reaches the destination:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\,[r - Q(s, a)] \tag{2-5}$$

In contrast, in Q-routing, the immediate rewards are based on the packet's arrival at the next node. Therefore, nodes get "expected time-to-go" information from neighboring node *z* to the destination *d*. This means that $min_z Q_y(d, z)$ in (2-1), is updated after each packet is sent to a neighbor only rather than the destination.

2.6.2 Model-based Reinforcement Learning in Network Packet Routing

The earliest work that proposes a model-based RL in network routing problems is [55] that uses collaborative feedback between agents to share the latest optimal policy in a non-stationary environment such as MANET. Optimization problems are decomposed into a set of discrete optimization problems that are solved by collaborating RL agents, forming the so-called collaborative RL, aka multi-agent RL (MARL).

Each agent has routing tables that store route cost information to its neighboring nodes and the last advertised route cost values from those neighbors to each destination in use. Collaborative RL adapts agent behavior in a changing environment to meet system optimization goals with agents using only local state information. Collaborative RL provides

feedback models that map changes in an agent's environment and neighbors onto internal changes in the agent's policy using distributed model-based reinforcement learning. It utilizes a decay function for the advertised cost values to allow agents to exchange the effectiveness of actions they have learned with one another.

2.7 Chapter Summary

This chapter stated the necessary background information from blockchain emergence, description, main features, and significance as a foundational technology. We discussed the scalability challenge as the key challenge of the widespread blockchain adoption. Then, we described the background of payment channel networks introducing the concept of layer-two protocols. We summarized the related work that addresses payment channel routing problems. Finally, we discussed the application of reinforcement learning in packet routing, which is decomposed into model-free and model-based techniques. We aimed to introduce applying reinforcement learning in transaction routing in PCNs.

Chapter 3
System Model and Problem Formulation

3.1 Introduction

This chapter investigates the payment channel network model and the formulated transaction routing problem. First, we introduce the payment channel network model. We then formulate the routing problem. We utilize a machine-learning technique to find the best available path to execute the payment. We then illustrate the payment model and the state diagram that we use to apply reinforcement learning in the PCN routing. We demonstrate the proposed RL model update methodology. Finally, we discuss the difference between the PCN routing problems versus those in communication networks using reinforcement learning.

3.2 Payment Channel Network Model

We model the payment channel network as a directed graph, similar to prior work [28], [38] since funds can flow in either direction and channel balances are dependent on the direction. We define the graph $G = (V,\ E)$, where V is the set of accounts in the network, E is the set of payment channels between a pair of accounts, and a weight function w of the set of edges. A link from a node, z, to a node, v, exists if z can transfer funds to v. The neighbors of the node v can be expressed as $N(v) = \{z \in V:\ (z, v)\ \in E\}$.

Although a node v can have a limited number of direct neighbors $N(v)$, it can 'route' payments to participants to which no direct network connection exists by chaining channels together. This is achieved by forming a payment path p via a sequence of edges $e_1, \dots, e_n$ with $e_k = (v_k^1,\ v_k^2)$, for $1 \le k \le n-1$. Therefore, the routing problem can be defined as the process of finding the minimum cost path starting at the source vertex, and reaching all destination vertices, by using the available graph edges. This path is actually a spanning tree whose vertices include the source (i.e. a root node) and destinations (i.e. leaf nodes that do not have any child nodes). Moreover, function w reflects the value of funds that can be transferred from one node to another node that shares the same edge, e. Hence, the maximum possible transaction value is limited by the path capacity (i.e., the funds available in a path $e_1, \dots, e_n$), which is defined as the $minimum\ w(e_k)$.

A node, *src*, wants a send a payment to node, *dst*, via payment channels, while both are not directly connected. We aim to find a connected path of payment channels in which each

link has sufficient funds in the payment direction for a single path payment. If the payment is divided over multiple paths, the payment is split so that each channel in each path can handle the partial payment. For simplicity, we assume a single path only, while a multiple-path routing is postponed for future work.

Therefore, the objective is to find a set of edges $e_1, \dots, e_n$ for transferring the payment between *src* and *dst* satisfying the following requirements, which are in line with the previous works that address the same problem [20], [28], [16]:

Effectiveness: The routing algorithm should find the paths that maximize the probability of a successful transaction. Upon channel balance change, the algorithm should remain effective.

Efficiency: The traffic generated by route discovery should result in low latency, computation and communication overheads. Besides, low overhead costs should be paid in case of PCN dynamic topological changes.

Scalability: The routing algorithm should maintain effectiveness and efficiency in large-scale PCN with massive amounts of transactions.

Cost-Effectiveness: The routing paths should incur transaction fees that are as low as possible. They should be consistently lower than layer-one protocols.

Node's Privacy: No certainty can be given to whom the sender/receiver are in a transaction. The distribution of funds among the two-channel participants should not be disclosed.

The stated requirements should be fulfilled by the routing algorithm considering the fast dynamic nature of PCNs. As stated in Section 1.4, both transient errors (insufficiently funded channels) and intransient errors (such as offline nodes) are identified as the main reasons for failing payments. Therefore, the routing algorithm should be verified to handle a considerable amount of these errors.

It is worth mentioning that peer-to-peer PCNs differ from common peer-to-peer networks as the connections between peers are predefined and cannot be changed to improve the routing quality. Therefore, they are argued in the literature to be considered friend-to-friend (F2F) networks, which restrict connections to peers sharing a mutual trust relationship [28]. Moreover, opening or closing a payment channel is a costly process. Hence, it is not expected that connections between peers frequently change with the same rate as the communication networks.

3.3 Problem Details and Formulation

We aim to address the fast dynamic nature of PCNs by using machine learning. As described in Section 2.5, RL is the type of machine learning techniques that dominates the area of traffic routing with the aim of achieving good scalability and low overhead. In reinforcement learning, each agent interacts with the environment described by a set of states, *S*. Hence, it learns and gains experience by exploration rather than having examples. The agent performs an action and uses the environment's feedback to maximize the reward for a given state in the form of rewards or penalties. We propose modeling the nodes in a PCN as agents collaborating and sharing information to perform successful transactions. This is achieved by selecting the most suitable action (i.e. a payment channel with a direct neighbor) to route the transaction with the lowest fees.

In the literature, both model-free and model-based RL algorithms are proposed for communication routing problems. Model-based methods are guaranteed to converge faster to the optimal policy as they make use of the data stored in their internal models. Generally, in distributed systems where gaining real-time experience is costly, model-based techniques are superior to model-free approaches.

Therefore, we propose using multi-agent reinforcement learning (MARL) in PCNs. MARL is a model-based technique to "learn" the nodes' behavior (i.e. build a model for the optimal policy π^*) and use it to predict the nodes' status as a function of time for future route suggestions. This implementation is supported by the study in [29] based on real-world transactions in the Ripple network that concludes that cryptocurrency transactions are characterized as highly recurrent in terms of the sender, receiver, and value within a time frame of 24 hours. To allow the transaction source to take the most appropriate action (i.e. a specific payment channel to route the transaction), the time and the amount of the transaction and the destination affect the decision. Thus, the states used to describe the environment should reflect these aspects, unlike the packet routing problem concerned with the destination only.

Based on the above, it is expected to have performance gains when applying RL in PCN routing for two main reasons:

i) Peer-to-peer PCNs can be described as friend-to-friend, and,

ii) real-world transactions in PCNs are characterized to be highly recurrent.

Therefore, RL will be able to “memorize” and “learn” node and payment channel activities.

In communication network routing, the fast convergence feature of model-based techniques is achieved at the expense of slower execution and increased storage size. Nevertheless, these constraints of communication networks are not of the same weight in PCNs. This is because the execution time of a payment routing is of a time frame of seconds or minutes. Also, the transactions are characterized to be highly recurrent, so the memorized data for each node should be of a limited size.

3.3.1 Payment Model and State Diagram

In the context of reinforcement learning, the agent, i.e. the transaction source, *src*, interacts with the environment with the goal of performing a successful transaction. The environment is described by a set of states, *S*, which is a function of the transaction destination, *dst*, time *t*, and transaction value *c*. Each agent can perform one action out of a set of actions, *A*, representing the transaction's execution through a direct neighbor that forms with the agent a payment channel. Furthermore, a specific combination of neighbors could be utilized in case that partitioning the transaction on multiple paths is required, which is left for future work. For simplicity, no transaction is considered to be deferred unless it fails in immediate execution. Otherwise, a "postpone" action should be considered for each agent.

A cost function *Q(s, a)* is proposed to describe the routing quality, where $s \in S$ is the current state, and $a \in A$ is a possible action based on *s*. The cost function is estimated by measuring the route's transaction fees to a destination, *dst*, via a payment channel for a particular transaction value, *c*, and a specific time slot, *t*. The Q-value for each state-action pair is locally stored in lookup tables so that the routing decision is taken based on the node's own experience. The values in the tables are updated locally by the node after each routing decision. The table represents the model of the optimal policy that the agent continually wants to learn and improve.

Each new transaction (TXN) is represented in the proposed model as an episode in the reinforcement learning process, as shown in Figure 3-1. An episode is a sequence of actions that leads the agent from an initial state to a terminal state. The transaction source's goal is to perform a successful transaction at the first time to the transaction destination, *dst*, at an arbitrary time slot, t_0. If the source transaction fails, it would have a maximum number of trials tl_{max}, within the same episode. The episode of the interaction between the agent and environment will terminate in any of the following two cases:

i) Reaching the agent's goal, i.e., a successful transaction, or,

ii) Exceeding the maximum number of trials, i.e., $tl > tl_{max}$.

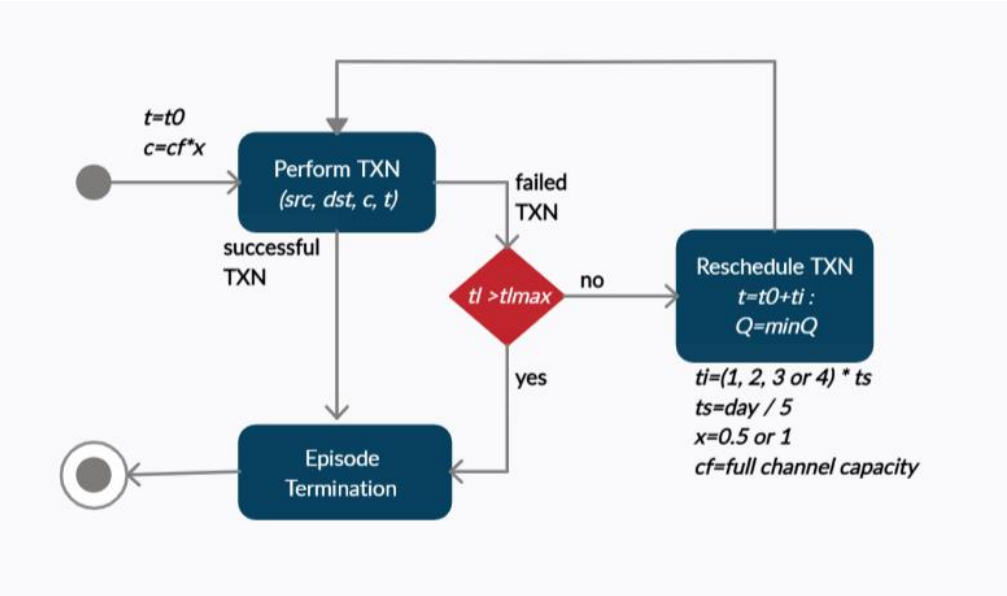


Figure 3-1. State diagram of Payment Model

Each state is described as a combination of a destination, time, and payment value. Since the transactions are characterized to be recurrent every 24 hours, the states that describe the environment need to be reset every day as well. The number of states used to describe the environment enhances the routing table resolution at the expense of increased storage size. For example, a time resolution of 30 minutes requires the storage of up to 48 states in the lookup table per payment slot for each destination. In case the payment value is described by two limits (e.g. half or full payment channel capacity), the number of states is up to 48*2 for each destination.

Therefore, in order not to consume the storage resource, we assume that the payment time is described by one of five values, called time slots. One epoch represents the time of resetting these time slots, which is one day in our model. The transaction value is approximated into two limits, namely, half and full the channel capacity. For example, two transactions with values less than half of the channel capacity at the same slot of time for the same destination will be considered one state in the lookup table. Hence, the state space is approximated into a reduced number of states that is limited to ten per destination. Their values are stored by the agent into a lookup table for each transaction destination due to the product of transaction time slot (i.e. five states) and transaction amount (i.e. two states).

3.3.2 The Proposed Model Update Methodology

Different reinforcement learning algorithms exist, and the convergence rates of some of them are known. The total number of RL algorithms introduced in the literature is vast. For model-based algorithms, R-MAX [60] is considered one of the best algorithms, according to [61] [53], that are guaranteed to find a near-optimal solution in time polynomial in the number of states and actions. It takes the idea of optimistic initialization to its extreme, in which all incompletely explored choices are assumed maximally rewarding, and optimal paths are computed to test that assumption.

In R-MAX, the agent always stores a complete but possibly inaccurate model of its environment and acts based on the optimal policy derived from this model. The model is initialized in an optimistic manner (optimism under uncertainty) with its maximum possible reward R-MAX (i.e. admissible heuristic). Upon the real model's execution, the agent's fictitious model makes a choice between exploration and exploitation implicit. So, we utilize R-MAX to update the RL model in the PCN routing problem. The model is locally memorized by each node in the form of local lookup table.

Let *n(s, a)* denote the number of times in which the agent has taken an action, *a*, starting from a state, *s*. The agent has observed the immediate rewards, *r[1], r[2],…., r[n(s, a)]*, after *n(s, a)* actions. In the proposed model, the node wants to take an action, *a*, of executing the transaction via a specific payment channel starting from a specific state, *s*, described by a payment value limit, slot of time, and a specific destination. The node observes the transaction execution to evaluate the rewards or penalties represented as transaction fees in the proposed model. The immediate rewards, *r*[*n*(*s, a*)], are the transaction fees after a successful transaction execution. In case of failure, the value of the immediate rewards is estimated, as discussed later in this section. Then, the empirical mean reward distribution, $\hat{R}(s,a)$, can be obtained as follows.

$$\hat{R}(s,a) = \frac{1}{n(s,a)} \sum_{i=1}^{n(s,a)} r[i], \tag{3-1}$$

Let *n(s, a, s')* denote the number of times when the agent has taken action *a* from state *s* and immediately transitioned to the state *s'*. In the proposed model, the next state, *s'*, is one of the states used to describe the environment. In this case, *s'* represents the same payment limit by the node itself to the same destination in another slot of time. The node tries the transaction

and waits for the results. In case of a successful transaction, the episode representing the node interaction of the environment terminates, so there is no next state. In case of a failing transaction, the transaction is retried in another slot of time representing the next state. This process is repeated till the maximum number of retrials is exceeded; hence, the episode terminates. So, the transition distribution, $\hat{T}(s,a)$, in our model reflects the probability of from a specific state, s, to another specific next state, s'. Then, the empirical transition distribution $\hat{T}(s,a)$ is given as

$$\hat{T}(s'\backslash s,a) = \frac{n(s,a,s')}{n(s,a)} \text{ for each } s' \in S \tag{3-2}$$

In the action selection step, the agent selects the action $\underset{a'}{\operatorname{argmin}}\, Q(s',a')$ that minimizes the current action value Q(s, .) that represents the transaction fees in our model. As the transaction fees are targeted to be reduced, they can be viewed as penalties that need to be minimized. In contrast, in case the rewards are targeted to be maximized, the agent should select $\underset{a'}{\operatorname{argmax}}\, Q(s',a')$, which is not the case in our model. The update rule is to solve the following set of Bellman equations [57]

$$Q(s,a) = \begin{cases} \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s'\backslash s,a) \min_{a'} Q(s',a'), & n(s,a) \geq m, \\ U(s,a) & otherwise, \end{cases} \tag{3-3}$$

where $\hat{R}(s,a)$ and $\hat{T}(s,a)$are the empirical (maximum-likelihood) estimates for the reward and transition distribution from the state-action pair *(s, a)*, to the state-action pair, $(s',\ a')$, respectively, using only data from the first *m* observations, *n(s, a)* denotes the number of times in which the agent has taken action *a* from state *s*, $0 \leq \gamma < 1$ is the discount factor, which is a constant that determines the relative value of delayed versus immediate rewards and *U(s, a)* is the upper bound of the real value function or admissible heuristic where $U: SxA \rightarrow R$ satisfies $U(s,a) \leq Q^*(s,\ a)$, where $Q^*(s,\ a)$ is the *Q*-value when applying an optimal policy π^*, for all $s \in S$ and $a \in A$. This MDP uses the empirical transition and reward distributions for those state-action pairs that have been experienced by the agent at least *m* times.

In the proposed model, the observations are counted by the node for each state-action pair and stored in the lookup table. The admissible heuristic is the shortest path search, performed on-demand upon each transaction execution trial. This heuristic is used to estimate the transaction fees in case of a transaction failure.

3.4 Routing in Payment Channel Networks vs. Communication Networks Using Reinforcement Learning

The proposed algorithm is similar to the Q-routing of the communication networks in that both are distributed so that each node locally stores the Q-values. Within the context of RL, each node resembles an agent that could be a source node or any node along the path. In case there is a route of two sources to the same destination, the Q-values stored in their own routing tables will be different depending on their own learning experience.

On the other hand, the proposed algorithm is different from the Q-routing of the communication networks in the following aspects:

1. In Q-routing, the state is described by the destination. While in the proposed PCN model, the state is described by a combination of a destination, payment time and a payment value.
2. In Q-routing, the action is described as one of the transaction source direct neighbors. While in the proposed PCN model, the state is described as one of the directly connected payment channels.
3. In Q-routing, the Q-value is interpreted as the time estimate to deliver a packet to node *d* from node *x* via a neighbor node *y*. While in the proposed PCN model, the Q-value is interpreted as the transaction fees estimate to deliver a payment, *c*, at a time slot, *t*, from source, *src*, to destination, *dst*, via a neighbor payment channel described by an action *a*.
4. In Q-routing, the next state-action pair (s', a') is performed by a neighbor node. While in the proposed PCN model, the next state-action pair is performed by the same node at a future time.

3.5 Chapter Summary

In this chapter we introduced the system model and formulated the payment channel routing problem that finds the best available route with the lowest transaction fees to execute the payment. We described the machine learning technique that builds a model using reinforcement learning. The optimum routes are memorized considering the state of the environment upon the transaction execution. We compared our proposed algorithm in the payment channel network to the Q-routing protocol in the communication networks.

Chapter 4

A Machine Learning-Based PCN Routing Technique: The Proposed Algorithm

4.1 Introduction

In this chapter, we introduce a novel model-based reinforcement learning technique for transaction routing in PCNs. The proposed algorithm overview, route discovery, and route selection are described in the context of reinforcement learning. We illustrate the proposed technique by describing a motivating example. Then, we give a detailed description of the overall proposed transaction routing algorithm in PCNs. Finally, we introduce a way to model the impact of transient and intransient errors on the routing protocol.

4.2 Algorithm Overview

We propose a transaction routing protocol in PCNs, based on multi-agent reinforcement learning that we call "MARL-Routing". As shown in Figure 4-1, each node in the network represents an agent that has local lookup tables. The Q-value for each state-action pair is locally stored in such tables so that the routing decision is taken based on the node's own experience. The values in the tables are updated locally by the node after each routing decision. The table represents the model of the optimal policy that the agent continuously wants to learn and improve.

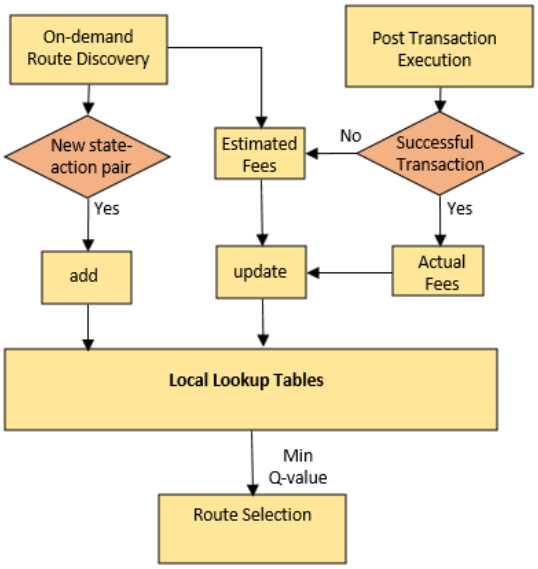


Figure 4-1. MARL-Routing illustrative diagram

To initialize this model, on-demand route discovery is required. We also use it to estimate the updated value in case of transaction failure. Therefore, on-demand route discovery and route selection are the main elements of the proposed algorithm that are detailed in the subsequent sections.

4.3 Route Discovery

Each agent has an internal environment model in model-based reinforcement learning and calculates the optimal policy. The agent relies on the rewards it gains via the interaction with the environment to build its own model. The agent also gathers some information on-demand upon a transaction execution and uses it to update the model. Routing information is collected locally through direct neighbors, including the cost function or the transaction fees to reach a destination. We establish this route discovery process on-demand in order to limit the communication overhead. Upon a transaction execution, the sender finds the shortest path to the destination with sufficient funds to perform the payment.

A variety of algorithms perform the path calculations based on the required performance and the forwarding scheme [62]. For example, widest paths [63], shortest-widest paths [63], [64], and widest-shortest paths [64], among others, are proposed in the literature [65]. The widest path is one of maximum available bandwidth, with bandwidth predicating a bottleneck weight function. Then, a widest-shortest path is the widest path among the set of shortest paths between two nodes. In contrast, a shortest-widest path is the shortest path among the set of widest paths between two nodes [65].

However, we leave the investigation of the best method to perform the path calculations that optimize the routing problem in PCNs in our technique to future work. For simplicity, the transaction fees are assumed to be fixed for all nodes. Thus, the shortest path is equivalent to the lowest number of hops. Our algorithm uses Breadth-First Search (BFS), described in Section 2.4, that expands the shallowest nodes first. BFS is optimal if the path cost is a nondecreasing function of the node's depth [66]. Thus, applying BFS in our implementation is the optimal solution that goes with the assumption of an equally weighted graph to find the shortest path. In the proposed implementation, we include the maximum number of hops and payment value in the route request. Hence, only nodes on paths with sufficient funds in the payment direction would respond.

It is worth mentioning that relying on the shortest path only does not consider channel balances. In particular, routing decisions might contain channels with low balances or implicitly turn bidirectional channels into unidirectional ones, reducing the available routes over time in a dynamic PCN [16][47]. On the other hand, our algorithm uses the shortest path search to help update the nodes' internal model, but the routing decision is taken based on the lowest Q-values. These values are updated so that they reflect the quality of selecting the shortest path, as discussed in the upcoming section.

4.4 Route Selection

In the context of reinforcement learning, each node stores a local lookup table containing Q-values for routing payment to a destination via a specific payment channel. These values are only considered based on the node's own experience, whether the node is the transaction source or an intermediate node in the payment path. The direct interpretation of the Q-value is an estimate of transaction fees that need to be minimized. This is the exact meaning when all previous payments are successfully completed. So, the actual transaction fees ($\hat{R}$ in (3-3)) are averaged and stored in the node's lookup table after transaction execution (i.e. no delayed actions). In our algorithm, the agent goal is to minimize the transaction fees that represent penalties rather than rewards in the context of reinforcement learning.

On the other hand, in case of a transaction failure, no substantial fees are actually charged. Alternatively, the expected fees for the shortest path in addition to an extra constant percentage of these fees (i.e. called the penalty factor, α) are stored in the lookup table after the unsuccessful transaction trial. This penalty factor that is added to the expected transaction fees represents the waste of opportunity of immediate payments. This is in line with the reinforcement learning concept that applies the minimum penalties in successful interactions with the environment. In contrast, higher penalties are applied in case of failure.

Thus, when all the transactions are successful, the Q-value will be minimum. However, in case repetitive failures are encountered using a specific payment channel, the Q-value starts to increase, driving the agent to select another payment channel. Hence, the Q-value is not only an indication of the transaction fees, but the figure involves the cumulative experience of using a specific payment channel to route a transaction to a destination under certain circumstances described by s.

The use of reinforcement learning in our implementation inherently solves the problem of channel imbalance. This problem may arise while using the shortest path criterion only, as

discussed in the previous section. When the payment channel is frequently depleted in the payment direction, this path will be avoided due to increased Q-value. At the same time, reinforcement learning proposes a solution to network underutilization since congested paths will have higher Q-values. Therefore, the agent will try other underutilized routes, even if they are of longer paths.

A route is established only on-demand on a hop-by-hop basis. In hop-by-hop routing, every node maintains a lookup table that indicates the next hops of the routes to other nodes in the network. For a transaction to reach its destination, it only needs to carry the destination address. Intermediate nodes forward the payment along its path based on only the destination address. In other words, the transaction sender selects only the direct neighbor, but it should not be aware of any further node across the path. Similarly, each intermediate node can decide on how to forward the message to the final receiver. Each node along the path appears to its successor as the payment source. Therefore, the sender's real identity should not be revealed, which preserves the sender's privacy. Nevertheless, the proposed algorithm's privacy assessment and its impact on the performance are left for future work.

4.5 A Motivating Example

We use an example to illustrate the proposed technique's functionality using a simple 15-node network model, shown in Figure 4-2.

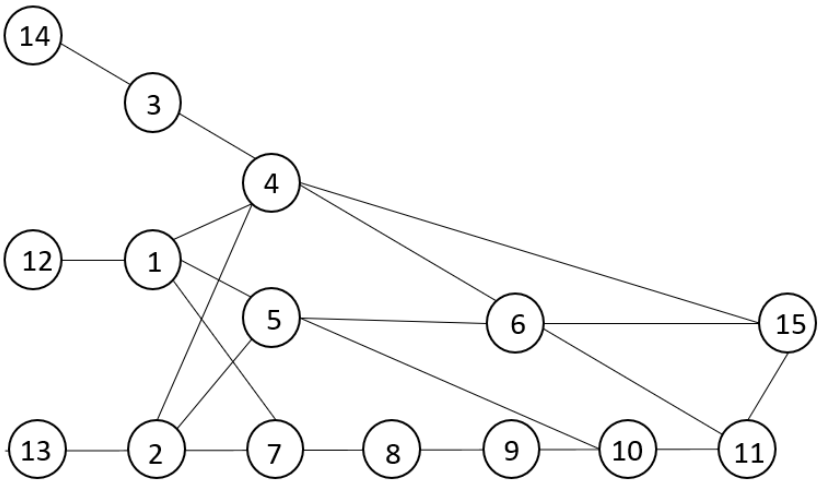


Figure 4-2. Simple 15-node network model

i) Node 1, src, has three direct payment channels via 4, 5, and 7 that can reach node 15, *dst*.

ii) Node 1 performs a route discovery process to node 15 and updates the shortest path table based on the network's current state.

iii) Based on the transaction destination *dst*, transaction value *c*, and slot of time *t*, node 1 selects the corresponding state *s* in the lookup table and selects the action with the lowest transaction fees (i.e. Q-value in our implementation) via node 5.

iv) If the state *s* does not exist, node 1 creates the state-action value based on the shortest path heuristics.

v) Node 5 repeats the same previous steps to forward the payment. All the nodes along the path to the destination will also do the same.

vi) Upon transaction success, node 1 updates the Q-value utilizing the actual transaction fees.

vii) Upon transaction failure, node 1 updates the Q-value utilizing data from the route discovery in addition to a penalty identified by an extra constant percentage of these fees (the penalty factor, α).

   a. Another slot of time is suggested to retry the payment based on the lowest Q-value stored in the lookup table. Alternatively, this is retried after two time slots, if there are no available suggestions.
   b. Node 1 repeats steps (ii) & (iii) and selects node 4 to execute the transaction.
   c. In case the transaction is successful, node 1 updates the Q-value based on the actual transaction fees rather than the advertised values in the route discovery.
   d. In case the transaction fails for the second time, node 1 repeats steps (vii-a) & (vii-b) until the maximum number of trials is reached.

4.6 The Detailed Overall Algorithm

As illustrated in Section 3.5, each new transaction (TXN) is represented in our implementation as an episode in the reinforcement learning process, as illustrated in Algorithm 4-1 and depicted the flow diagram in Figure 4-3. The agent or transaction source interacts with the environment with the goal of performing a successful transaction (Line 1).

In the beginning, *Q(s, a)* has to be initialized with the shortest path to the destination, which represents an admissible heuristic $U(s, a_h)$. Therefore, each agent v will receive information, I_v, collected from its online neighbors (Lines 3-18) / (Figure 4-3: FD-1). This advertised information is intended to conduct route discovery in our model and will be utilized

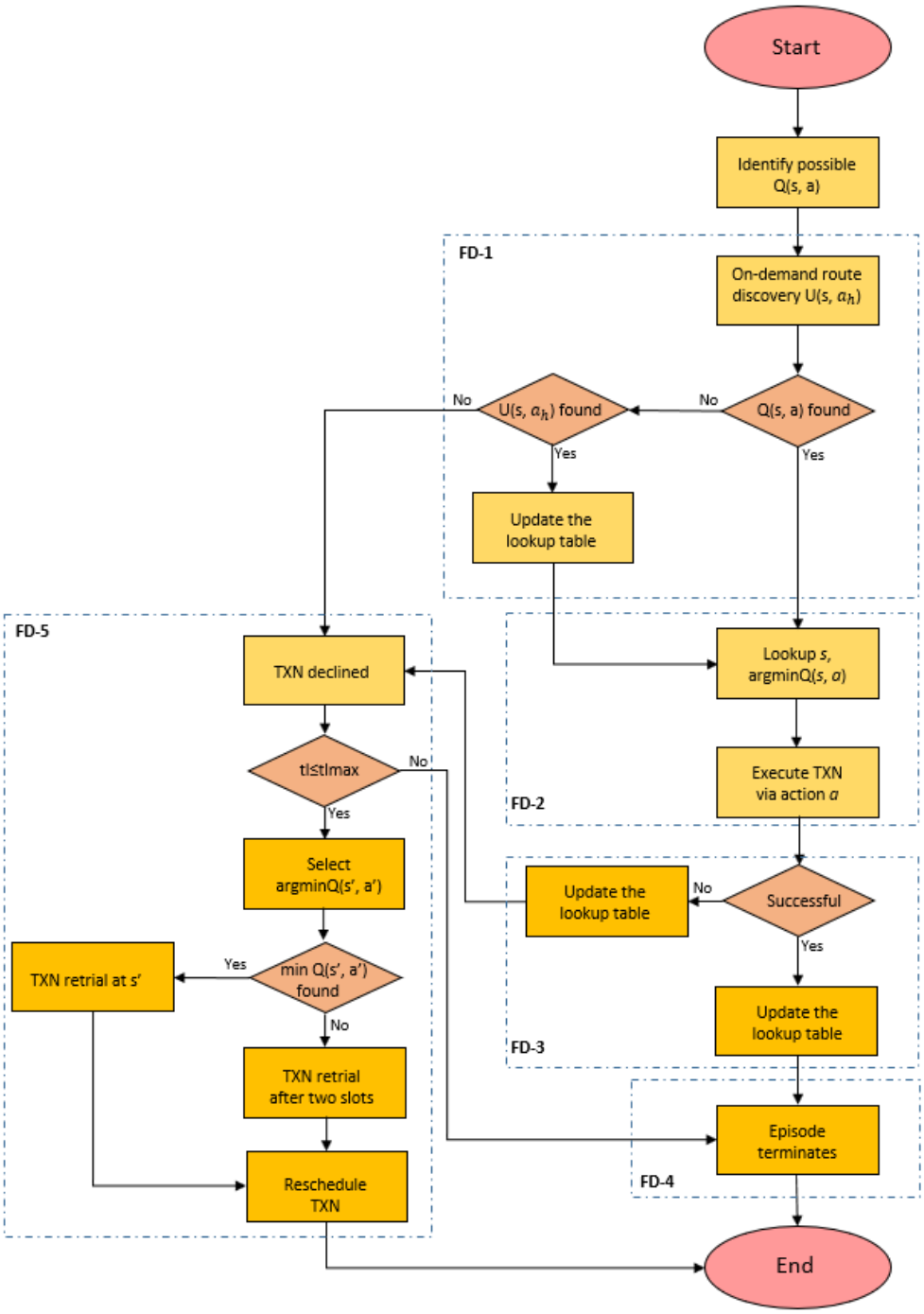


Figure 4-3. MARL-Routing flow diagram

to communicate the shortest path as admissible heuristics to initialize the environment model. The input data during a maximum allowable number of hops, h_{max}, and the value of the transaction, c. The node submits these queries to its neighboring nodes that satisfy the condition that the payment channel capacities must be higher than the transaction value (i.e., $minimum\ w(e_k) > c$). In response, each node will reply with the transaction fees for the destination.

Algorithm 4-1: The MARL-Routing algorithm

Input: *Graph*, γ, m, α, h_{max}, and tl_{max}

Output: $Q(s,a)$

1. **for** all episodes do
2. **do** while (*TXN* !successful && $tl \le tl_{max}$)
3. #Look up the corresponding possible state-action pairs $Q(s, a)$ in *src*'s table
4. #Collect on-demand route discovery information I_v and $U(s, a_h)$
5. **if** $Q(s, a_h)$ is not found
6. $Q(s,\ a_h) \leftarrow U(s,\ a_h)$ in (3-3)
7. $n(s,\ a_h) \leftarrow 0$ in (3-1)
8. $n(s,\ a_h, s') \leftarrow 0$ in (3-2)
9. **if** $U(s,\ a_h)$ is not found
10. # *TXN* is declined
11. # lookup argmin $Q(s', a')$ a'
12. **If** min $Q(s', a')$ is found
13. # retrial at *s'* slot of time
14. **else**
15. #retrial after two slots of time
16. end **if**
17. end **if**
18. end **if**
19. execute action *a* such that
20. $a = argmin_a\ Q(s,a)$
21. $n(s,a) \leftarrow n(s,a) + 1$
22. **if** (successful *TXN*)
23. $rSum(s,a) \leftarrow rSum(s,a) + r(s,a)$
24. **if** $n(s,a) \ge m$
25. $Q(s,a) \leftarrow \hat{R}(s,a)$
26. **else**
27. $Q(s,a) \leftarrow Q(s,a)$ \\ no change
28. end **if**
29. episode terminates
30. **else**
31. *tl*++
32. #select argmin $Q(s', a')$ a'
33. #reschedule *TXN* for retrial
34. $rSum(s,a) \leftarrow rSum(s,a) + U(s,\ a_h)(1+\alpha)$
35. $n(s,a,s') \leftarrow n(s,a,s') + 1$
36. **if** $n(s,a) \ge m$
37. $Q(s,a) = \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s' \backslash s, a) \min_{a'} Q(s', a')$
38. **else**
39. $Q(s,a) \leftarrow Q(s,a)$ \\ no change
40. end **if**
41. end **if**
42. end **do**
43. end **for**

Basically, the agent aims to reduce the transaction fees in both success and failure cases. In a sense, the reward function is converted to a penalty if a failure is targeted to be minimized in our algorithm (Line 20). The agent action, a, is selected to achieve the goal that the cost function is to be minimized, $argmin_a\ Q(s,a)$, for the suggested route (Lines 19-21) / (Figure 4-3: FD-2). In case more than one action has equal Q-values, the one with the highest $n(s,a)$ should be selected. It represents a higher number of times that the agent has taken the same action, which indicates it is more experienced with this action. Upon the transaction execution, a route request is sent by the source via the selected payment channel to know the transaction fees. This request is forwarded on a hop-by-hop basis without revealing the original source identity.

The transaction is considered successful when all nodes along the route from the source to the destination are online and have sufficient funds to perform the transaction, including the transaction fees. On the other hand, the transaction is considered to have failed when at least one node along the route from the source to the destination is found offline, ceases to participate in the transaction routing, or does not have sufficient funds to perform the transaction, including the transaction fees.

The reward $r(s, a)$ in (3-1) represents the actual transaction fees in case of a transaction success (Lines 22-29) / (Figure 4-3: FD-3) or a representation for the waste of payment channel utilization in case of a transaction failure (Lines 30-41) / (Figure 4-3: FD-3). Therefore, the reward summation $rSum(s,a)$ is updated and memorized in the lookup table alongside $Q(s, a)$ after each TXN trial (Lines 23 & 34). In case of a successful transaction, an immediate reward, $r(s, a)$, is based on the actual TXN fees. Therefore, $\hat{R}(s,a)$ is calculated (Line 25). On the other hand, in case of a failure, the agent will assume $r(s, a)$, representing the function of a penalty. It is calculated based on the heuristics collected through advertising $U(s, a_h)$ and in addition to a penalty factor, α. Then the $rSum(s,a)$ is updated (Line 34).

As far as the Transition function $\hat{T}(s'\backslash s,a)$ in (3-2) is concerned, in case the TXN is successful, the episode terminates, and the transition function value becomes zero (Line 29) / (Figure 4-3: FD-4). In case of a failure, $\hat{T}(s'\backslash s,a)$ and *Q(s, a)* will be updated as described in (3-2) and (3-3), respectively (Lines 35 & 37). The next state, s', is transitioned to by the agent to retry TXN in a future slot of time based on the lowest *Q(s', a')* (Line 32) / (Figure 4-3: FD-5). Thus, the transaction is rescheduled. If there are no future suggestions, TXN will be retried after two-time slots as a default (Line 11).

To verify our assumption of considering the shortest path route discovery $U(s, a_h)$ as an admissible heuristic, we need to examine the original definition in a pathfinding problem. A heuristic function is considered admissible if it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is less than the lowest possible cost from the current point in the path. To apply this definition in our problem, TXN fees of the shortest path $U(s, a_h)$ are always guaranteed to be less than the lowest state-action value, $Q(s, a)$, (Lines 6, 25, 27, 37, and 39).

4.7 A Numerical Example

We use a numerical example, as shown in Figure 4-4, to illustrate how the agent selects between two channels *a* and *b* to execute a transaction to *dst*. Assume that the optimistic values stored in the local lookup table for channels *a* and *b* are 6 and 8, respectively. For simplicity, assume the number of transaction retrial, *tl* =0, the penalty factor, α = 0.5, the number of optimistic observations, *m*=1, and the on-demand route discovery shortest path *U*=8.

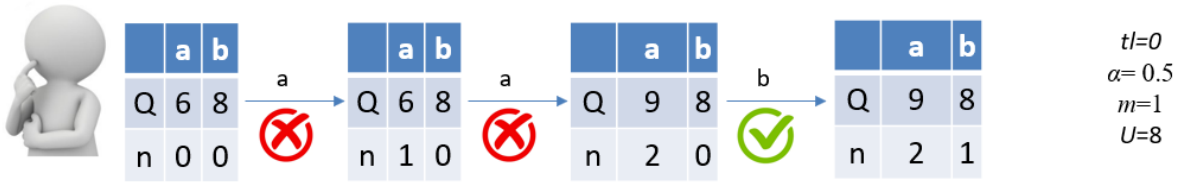


Figure 4-4. The agent selects between two channels based on the lowest Q-value

Action *a* suggests a shorter path than action *b*, but the agent has never tried both actions *a* or *b* (i.e. the number of times the agent selects an action, *n*=0). This information is captured earlier through the on-demand route discovery process and is used to initialize these two state-action pairs. The agent selects action *a* since it incurs lower transaction fees. Two transaction failures increase the Q-value of action *a* and drive the agent to select action *b*. Therefore MARL-routing proposes a solution to channel underutilization (e.g. channel *b*) and channel overutilization (e.g. channel *a*).

4.8 Non-Participating Nodes

One of the unsolved issues with routing algorithms, and PCN itself, is that a significant number of nodes should be continuously online. This requirement is absent in blockchain and virtually all centralized payment systems [20]. Therefore, we aim to assess the impact of non-participating nodes on the routing protocol performance. We introduce a new factor in our

assessment to simulate the inactive nodes during the routing process, called the participation percentage. The node's inactivity could be due to being offline, uncooperative, or the payment channel is depleted in the payment direction.

In case the routing algorithm is fully distributed, such as Max-Flow, Flash, or MARL-Routing, all the nodes across the path from *src* to *dst* have to be participating in considering the transaction successful. Also, the non-participating nodes are excluded from the route discovery process. In landmark-oriented algorithms, such as SilentWhisper and SpeedyMurmurs, all the nodes across the path from *src* to *dst* have to be connected to the tree root or landmark via participating nodes. So that the transaction is considered successful. This condition is in line with the original work of the SpeedyMurmurs algorithm [28] that considers different cases that call for coordinate changes of the existing nodes in the spanning tree. Among others, setting the value of non-zero links to zero is one of the reasons. Then, the on-demand repair mechanism included in the SpeedyMurmurs algorithm urges the child node and all its descendants to choose a new parent. It is aimed to increase the number of non-zero links in the spanning tree and the likelihood of transferring funds. This suggests that in case any node is non-participating, all its descendants should follow the exact behavior of coordinate changes. Otherwise, the node and all its descendants connected to the spanning tree root by a non-participating node will be isolated from the rest of the network.

4.9 Chapter Summary

This chapter introduced the details of the proposed model-based machine learning technique to be used for transaction routing in PCNs. We began by highlighting the overall proposed algorithm functionality, route discovery, and route selection in the light of model-based reinforcement learning. We used the BFS algorithm to find the shortest path to initialize the model and estimate the rewards or the penalties. A motivating example to demonstrate the proposed technique was presented. We described the details of the proposed algorithm for the routing problem at hand. Finally, we introduced the proposed new factor, called participation percentage, to model both the transient and the intransient errors to assess their impacts on the routing protocols.

Chapter 5
Simulation Results

5.1 Introduction

In this chapter, the proposed machine learning-based routing protocol's performance is evaluated through two real-world off-chain network topologies: Ripple and Lightning. Subsequently, we evaluate the simulation results along with five different performance metrics. In particular, the success ratio, routing transaction fees, routing efficiency, transaction overhead, and maintenance overhead are analyzed. Furthermore, we discuss the impact of selected network and simulation parameters on the performance.

5.2 Network Topology

The network topology could substantially influence the routing protocol performance, so it is crucial to consider a variety of possible topologies. It is widely recognized that many real-world complex network systems display some organizing principles, which often lead to either small-world topology or scale-free topology [67] [68]. Small-world networks describe networks with high clustering, few hubs, and short average path lengths. Examples of small-world networks include peer-to-peer file sharing [69].

In scale-free networks, nodes preferentially attach to higher degree nodes leading to few high degree hubs and little clustering. Examples of scale-free networks include the Internet. Scale-free networks have been used in simulation models for timing analysis of the Bitcoin network [70]. Besides, research has concluded that the scale-free model well approximates the Lightning network [71]. Such studies may not be indicative of the networks in this work since the proposed setup does not use the same networks as in these studies.

The evaluation is conducted using two real-world network topologies, namely, the Ripple and the LN networks. The Ripple network, which to the best of our knowledge, is the only payment network whose transaction data are publicly available [28]. Each transaction data entry includes the sender, the receiver, the transaction volume, and transaction time information. We use the same ripple network dataset used in the studies of [28],[38],[29] that propose routing algorithms for the same purpose.

As described in Chapter 1, the Ripple is a permissioned public blockchain network, where transactions are executed on-chain with medium scalability and managed by a privately held company. It is assumed that the on-chain and off-chain transactions characteristics are almost similar since on-chain transactions are moving to be off-chain for rapid settlement with lower fees [29]. Nevertheless, the Ripple payment protocol is not actually implemented in the simulator. We only utilize the network graph's existing snapshots and the actual transactions to assess the different routing protocols.

The second real-world network topology is the Lighting Network. As stated in Chapter 1, it is the most prominent PCN for the largest cryptocurrency by market capitalization, Bitcoin. The available real-world data is the network graph only, without including the transactions. We use the same network topology used in [29], proposed to solve the same problem, where nodes and channels were obtained as a snapshot of the Lightning network on a particular day of December 2018.

5.2.1 Network Topology Parameters

Although topological analysis of the proposed network is beyond the scope of this work, network topology metrics for the Ripple network and the Lightning Network, utilized in the simulation, are as defined in Table 5-1. These parameters are helpful in the performance analysis of the routing protocols. We rely on these parameters in assessing the change in the routing protocols' behavior under the two different simulated topologies, as illustrated in the Results Section 5.6.

Average degree is the straightforward and perhaps also the most critical characteristic of a single node. The degree *ki* of a node *i* is usually defined to be the total number of its connections. Thus, the larger the degree, the more influential the node is in a network. The average of *ki* over all *i* is called the *average degree* of the network and is denoted by $< k >$ [72]. It is the coarsest connectivity characteristic of the topology. Networks with a higher average degree are "better-connected" on average [73].

Clustering Coefficient (C) is the average fraction of pairs of neighbors of a node that are also neighbors of each other, where $C \leq 1$; and $C = 1$ if and only if the network is globally coupled, which means that every node in the network connects to every other node [72]. Clustering expresses local robustness in the graph and thus has practical implications: the higher the local

clustering of a node, the more interconnected are its neighbors, thus increasing the path diversity locally around the node [73].

Table 5-1. Network topology parameters

Parameter	**Value**	Ripple	Lightning
Nodes	Number of nodes in the network	80,591	2,509
Edges	Total number of links in the graph	245,394	34,022
Average Degree	Average number of links connected to a node	6.09	27.12
Clustering Coefficient	Measure of how nodes tend to cluster together in a graph	0.082	0.308
Node with 1 Link	Number of nodes with a single Link	29361	588

5.3 Simulation Model and Payment Generation

We use an existing simulator of payment channel networks to model the transactions' arrival and execution functions [28]. It is a java-based simulator that makes use of the graph analysis tool GTNA (Graph-Theoretic Network Analysis for P2P structured networks) [74].

For the Ripple topology, we use the same Ripple network dataset used in [28]. As performed in the same paper, the chosen transactions are guaranteed to be successful using the Max-Flow (Ford-Fulkerson) [41] as the baseline algorithm. The transactions were executed using Ford-Fulkerson, and only the successful transactions were used in further simulations of this work. Max-flow-based routing is a reference technique in terms of throughput and transaction success rate, but it has a high computational complexity overhead [44]. Since the nodes' behavior is valuable, in terms of the transaction value, time, and the destination, in the proposed routing algorithm, the raw data are reprocessed to include the actual Unix timestamps, unlike the proposed work [28] that uses a time-averaged dataset.

We generate payments for the Lightning Network topology by randomly sampling the Ripple trace for the Ripple topology. In other words, due to the lack of sender-receiver information in the Bitcoin trace for Lightning, and similar to [29], we randomly sample the Bitcoin trace for transaction volumes and sample a sender-receiver pair from the Ripple trace and map it to the nodes in the Lightning topology. Payments arrive at senders sequentially, which is considered a time-averaged behavior. Similar to the Ripple network, the chosen transactions are guaranteed to be successful using the Ford-Fulkerson algorithm.

To simulate the proposed technique, the simulator is modified to include participation functionality for each node. If the node is offline or online but unresponsive during the payment

execution, it is considered non-participating. This, subsequently, results in a transaction decline. Due to the absence of actual dynamic network topology models, different participation assumptions are considered to cover a range of practical scenarios. Based on the target participation percentage of the total number of nodes, each node participation status is randomly selected. The selection is altered at the beginning of each time slot, i.e., five times a day. The random number generation criterion, used to decide the node's status, is weighted by the number of edges connected to each node (i.e., payment channels). So, the more connected the node, the higher probability of being participating. To allow an equal basis comparison, the random function is seeded from the Unix time for each new time slot, which is fixed for all routing techniques.

For simplicity, the transaction fees are assumed to be fixed for all nodes. Thus, the minimum cost function is equivalent to the lowest number of hops. For a fair comparison, we modified the SpeedyMurmurs algorithm to dynamically repair the spanning tree in case of a node being non-participating. Thus, it is excluded from being a parent node the same way as the coordinate change for zero-edge in the original work, as discussed in Section 3.7.

5.4 Simulation Setup

For the Ripple network, the simulation is performed on 328,230 Ripple transactions starting from January 2013 till November 2016. For the Lightning network, the simulation is performed on 363,520 randomly sampled transactions as described in Section 5.3.

The nodes' participation is randomly altered at the start of each time slot. The random function was seeded from the time for each new time slot. In order to assume the participation percentage, it has to be linked to the actual network topology upon executing the transaction. Due to the absence of actual network topology models, different participation assumptions are considered to cover a wide range of practical scenarios. As of this writing, the number of Lighting network active nodes, for example, is almost 50% of 17,439 (i.e. 50% of the total number of nodes) [31].

On the other hand, the number of active addresses that have either sent or received a transaction in the Ripple [75] and Bitcoin [76] is less than 5% of the total non-zero addresses. As discussed earlier in Section 5.2, the on-chain characteristics are moving to be off-chain. Besides, as demonstrated in Section 3.7, this factor also involves transient errors such as insufficiently funded channels. Therefore, a lower participation percentage should be

considered in the simulation. The simulated participation percentages are selected to be 100% and 50%. An additional participation percentage of 5% is considered to demonstrate the routing protocols' resilience.

The simulation parameters are summarized in Table 5-2. The presented results are the averages of 10 experimental runs, using a different set of transactions for each run. The proposed routing technique is evaluated in comparison to Flash and SpeedyMurmurs. The former combines max-flow with the shortest path search to find the suitable path, while the latter is a spanning tree-based routing protocol [28]. Distributed Ford-Fulkerson is used as the benchmark algorithm for the success ratio for the routing protocols under evaluation. In addition, Ford–Fulkerson is the baseline method for several routing protocols in the literature, proposed to address the same problem. Further details about the routing protocols exist in Section 2.4.

For Ford-Fulkerson and SpeedyMurmurs algorithms, we use the same implementations of [28]. For the Flash algorithm, we implement the same algorithm and input parameters used in the original work [77]. It is worth mentioning that the Flash algorithm uses a threshold payment value (i.e. as a percentage of the whole transaction values) to differentiate between low transaction values, aka mice payments, that use k-shortest paths stored in local routing tables. On the other hand, high trasanaction values, aka elephant payments, use a modified max-flow based algorithm for the routing.

Table 5-2. Network and simulation parameters

Parameter	**Value**
All: Maximum number of trials tl_{max}	2
All: One epoch	24 hrs
Ford-Fulkerson: Number of paths	Unlimited
MARL-Routing: Discount factor γ	Ripple: 0.95 / Lightning: 0.35
MARL- Routing: Penalty Factor α	0.1
MARL- Routing: No. of optimistic observations *m*	1
MARL- Routing: Maximum number of hops h_{max}	8
MARL- Routing: Number of states for each *dst*	5 for time x 2 for amount
MARL- Routing: Number of paths	1
SpeedyMurmurs: Number of landmarks	1
Flash: Elephant-mice threshold	90%
Flash: Number of paths "mice payments"	4
Flash: Number of paths "elephant payments"	20

In case of a failure in the algorithms used for comparison, the transaction is randomly retried within one day in any slot of time, including the current slot. While in our algorithm, the transaction is retried in another slot of time within a day, based on the time slot with the lowest Q-value. For SpeedyMurmurs, selecting one landmark is in line with the original work results since increasing the landmarks negatively impacts the success ratio. It is worth mentioning that the number of landmarks in the original work is interpreted as the number of spanning-tree roots, which is equivalent to the number of paths.

5.5 Performance Metrics

Five performance metrics are used to evaluate the proposed algorithm in comparison with the other routing protocols as follows:

Success Ratio: describes the total number of successfully completed payments over the total number of generated payments.

Transaction Fees: represents the average path length of successful transactions. When the transaction is performed via multiple paths, the summation of all these paths is counted.

Routing Efficiency (eff): represents the average ratio of shortest path length over the routing path length (where the path length of unsuccessful paths is infinite) [78]. The routing efficiency is described as follows:

$$eff = \frac{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \frac{sp_{ij}}{p_{ij}}}{n(n-1)}, \tag{5-1}$$

where *i* and *j* are two nodes within the set of n nodes, $sp_{ij}$is the shortest path length from *i* to *j* and $p_{ij}$is the routing path length from *i* to *j*. The ratio $\frac{sp_{ij}}{p_{ij}}$ assumes values in the interval [0, 1]. When the routing is unsuccessful, the path length is infinite, and therefore the ratio is zero, which represents the worst case. When the routing is successful, the path length is greater than zero, and the ratio tends to 1 as the path length tends to the shortest path length, becoming one in the best case. The routing efficiency is the average of this ratio over all the node pairs.

Transaction Overhead: denotes the traffic required for route discovery and collecting information from neighboring nodes.

Transaction Maintenance Overhead: describes the traffic necessary to complete a successful transaction for repeated payments between the same pair of node and destination. The node could be the transaction source or any node along the payment path.

5.6 Results

The proposed routing protocol is assessed in comparison with Flash and SeedyMurmurs using two real-world network topologies as detailed in this section. We use the metrics defined in Section 5.5 for our comparison.

5.6.1 Ripple Network

We conduct the evaluation using actual Ripple network topology and transactions. The impact of different participation percentages, namely, 100%, 50%, and 5%, is assessed for each performance metric.

5.6.1.1 Success ratio

As demonstrated earlier in this chapter, the Ford-Fulkerson algorithm is the baseline in terms of success ratio, under the condition that all nodes are online (i.e., 100% participation), and the success ratio is 100% in that case. However, Ford-Fulkerson shows a decline in the success ratio as the participation percentage decreases to 50% and 5%. This is expected because the non-participating nodes are excluded from the suggested routes.

As shown in Figure 5-1, MARL-Routing is able to get comparable performance to Ford-Fulkerson, in terms of success ratio, at 100%, while it outperforms Ford-Fulkerson at 50% and 5% node participation. This difference results from the embedded intelligence in our algorithm, which is gained from the learning process. In particular, the machine learning-based protocol selects the most suitable time slot for the transaction retry in case of a failure. Therefore, if there is no path of online nodes or channels with sufficient funds between the source and the destination in a specific time slot, the transaction is retried in another time slot within a day, based on the stored highest Q-value.

On the contrary, in Ford-Fulkerson, Flash, or SpeedyMumurs, the failed transaction is randomly retried within the same day. This includes the time slot in which the transaction has already failed due to the lack of participating nodes necessary to form a path to the transaction destination since no specific slots of time have a preference in these algorithms.

The machine learning capability that considers the payments' recurrency feature is why the comparable performance of success ratio in our algorithm with the baseline algorithm or exceeding that of Flash. Although that Flash is a distributed algorithm that combines routing table and max-flow, MARL-Routing outperforms it at 100% participation. The reason is that Flash simply reuses the existing paths if the receiver is in the routing table without considering on-demand route discovery. The authors depend on the recurrency feature of mice payments. Our algorithm uses this feature and also considers the fast dynamic nature of PCNs. It performs on-demand route discovery even if there are several stored Q-values for the same destination. This helps balance the exploration and exploitation and combat the underutilization of some paths.

The SpeedyMurmurs simulation results, when all nodes are participating, is around 90%, which is in line with the average figure in the original work [28] for the same number of transaction retries. However, the spanning tree reconstruction of SpeedyMurmurs is not able to cope with a further reduction of participating nodes of the network. This leads to a significant decline in the success ratio at participation percentages of 50% and 5%. The rationale beyond the SpeedyMurmurs deterioration is that, unlike a fully distributed approach in MARL-Routing and Flash, not only the nodes, along the path from the sender to the receiver, have to be participating. Nevertheless, they have to be connected to the landmark (i.e. the spanning-tree root) with participating nodes.

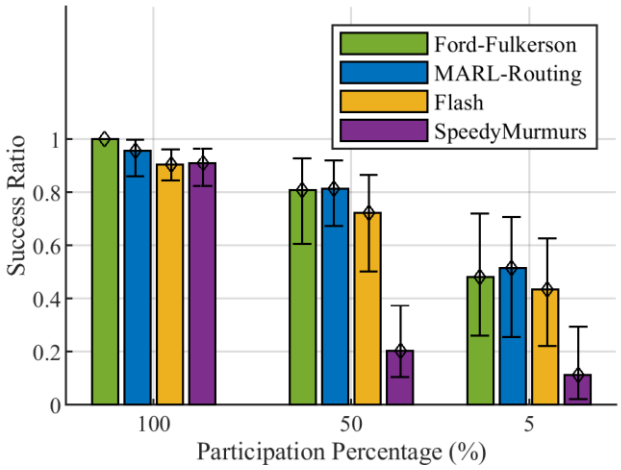


Figure 5-1. Success ratio at various participation percentages in Ripple network

5.6.1.2 Average Transaction Fees

It is appropriate to recall the assumption in Section 5.3 that the transaction fees are considered to be fixed for all nodes. Accordingly, where multiple paths are required to complete

the transaction, the summation of the number of hops for all the paths has to be counted to represent the transaction fees.

As shown in Figure 5-2, the Ford-Fulkerson algorithm, which works on finding all the possible paths, gains a considerable rise in the transaction fees of the successful transactions when compared to the other algorithms. Since the number of intermediate nodes in Ford-Fulkerson is unlimited, there is a slight increase in transaction fees as the number of non-participating nodes increases. The algorithm tends to find the maximum flow that the network allows from a source to a destination. This concept implies that if the number of non-participating nodes increases, the average path gets longer since the offline nodes cannot be counted in the path.

On the contrary, both the SpeedyMurmurs and our algorithm work on finding the shortest path and using a single path in the current simulation. Therefore, reducing the successful transactions' average path length, thus the associated fee, occurs. This is because those transactions are successfully executed when the sender and receiver are closer in terms of the number of hops. Flash relies on routing tables that store k-shortest paths in more than 90% of the payments categorized as mice payments. This results in low transaction fees that are comparable to the SpeedyMurmurs and our algorithm.

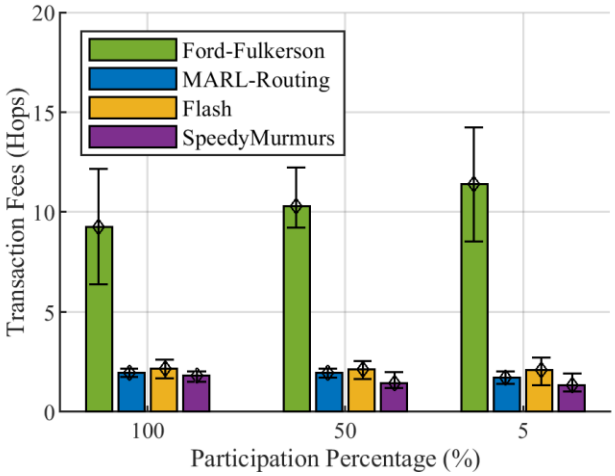


Figure 5-2. Transaction fees of successful transactions in Ripple network

Although the SpeedyMurmurs algorithm shows lower transaction fees, this can be justified by the lower success ratio. For example, at 5% participation, our algorithm's average path length involves the execution of transactions of paths of 1, 2, 3, and 4 hops. In contrast, at

the same participation percentage, the successful transactions in the SpeedyMurmurs consist of the payments where the source and the destination are mainly one hop apart. Longer paths may be available, but they are not recognized due to taking both algorithms' greedy actions based on the shortest path. Hence, the successful transactions' average path length could not be assessed in isolation from the success ratio or the number of the simulation paths.

5.6.1.3 Routing efficiency

The routing efficiency measure allows the routing navigability assessment with a score that integrates both the concepts of success ratio and average path length of successful transactions. It provides a unique solution when both performance metrics suggest conflicting results. As shown in Figure 5-3, MARL-Routing outperforms Ford-Fulkerson, Flash, and SpeedyMurmurs at all participation percentages.

The Ford-Fulkerson algorithm is generally characterized by a high success ratio and long average path length, while the opposite features apply to the SpeedyMurmurs. Our algorithm and Flash relatively combine the tradeoff between a high success ratio and a low average path length. The machine learning feature in our algorithm results in a superior performance to that of other algorithms. Therefore, MARL-Routing is able to surpass the other algorithms in terms of routing efficiency.

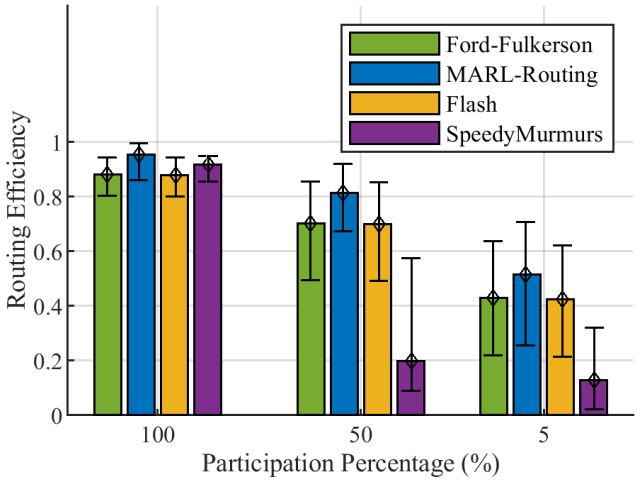


Figure 5-3. Routing efficiency in Ripple network

5.6.1.4 Transaction overhead

Nodes exchange messages to collect route discovery information in order to execute a transaction, as described in Section 4.6. Since MARL-Routing, Flash, and Ford-Fulkerson

algorithms are fully distributed routing techniques, the number of messages is expected to be quite large upon transaction execution. It is appropriate to recall that the fully distributed payment network is a crucial feature of blockchain, which our algorithm is designed to maintain. This decentralized approach will be at the expense of increased transaction overheads compared to any landmark-oriented algorithm such as SpeedyMurmurs.

On the contrary SpeedyMurmurs experiences an enormous overhead in case of a coordinate change is initiated in the spanning tree, called stabilization overhead [28]. This is so frequent due to the fast dynamic nature of PCNs, as stabilization overhead is required to maintain necessary state information. As described in Section 2.4, the stabilization overhead is logarithmically scaled to the network size. Nevertheless, the stabilization overhead metric does not apply to the other algorithms since all the traffic is generated on-demand upon transaction execution.

Flash performs route discovery for k-shortest paths for mice payments if the destination is not in the routing table. Also, the route discovery is performed for the transaction retrial in case the stored paths have failed to execute the transaction. This aids in limiting the transaction overhead in Flash.

On the other hand, MARL-Routing gains transaction overhead on-demand upon each transaction execution. Nevertheless, this overhead is rationalized by searching for the minimum sufficient paths only rather than a predefined number of paths. However, as shown in Figure 5-4, our algorithm outperforms Flash and reduces the transaction overhead by a factor of 4.5 at 50%.

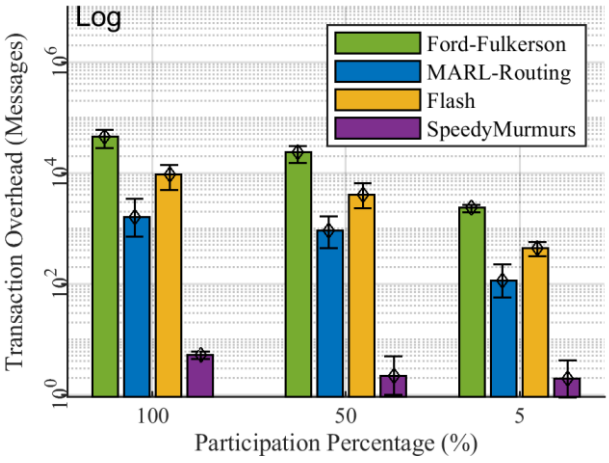


Figure 5-4. Transaction overhead at various participation percentages in Ripple network

5.6.1.5 Transaction maintenance overhead

This metric considers the average traffic generated, which is deemed necessary to perform a successful transaction. In MARL-Routing, the amount of traffic generated is not equal for each transaction towards this average. In case there is a new combination of state-action pair (i.e. payment category, slot of time, destination, and payment channel), the transaction's share in the maintenance overhead is relatively high. In case the combination is not new and was previously used to make the routing decision towards a successful transaction, this figure is minimal. Similarly, Flash reuses existing paths stored in the lookup tables for repeated transactions to the same destination for mice payments. These payments are around 90% of the executed transactions. At the same time, higher payment values rely on a max-flow algorithm, which generates a massive amount of traffic upon each transaction execution even if the transaction is repeated.

In MARL-Routing, the transaction maintenance overhead metric is different from the transaction overhead for successful transactions. The transaction overhead also counts the traffic generated for collecting data from neighbors to enhance the routing table accuracy and balance between the exploration and exploitation. This route discovery information is unnecessary for a successful payment if the node has existing Q-values in the lookup table and uses it to take the routing decision in a repeated transaction.

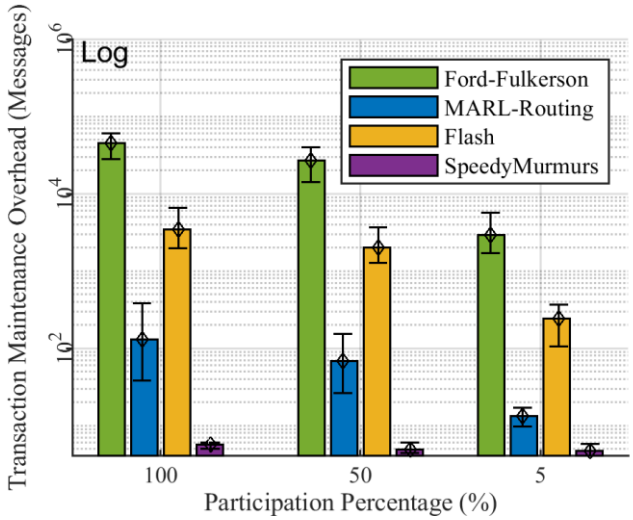


Figure 5-5. Transaction maintenance overhead of successful transactions in Ripple network

As expected, Ford-Fulkerson experiences a massive transaction overhead, as depicted in Figure 5-5. The performance of SpeedyMurmurs is also expected since the maintenance

overhead of the successful transactions depends on their shortest path lengths. Averaging this parameter over the total number of transactions should not be assessed in isolation from the success ratio, as discussed earlier in this section. MARL-Routing outperforms Flash as a fully distributed technique and reduces the transaction maintenance overhead by a factor of around 4 in all participation percentages. Lower transaction maintenance overhead suggests faster successful transaction execution.

5.6.2 Lightning Network

As described earlier in Section 5.2, we use a real-world network implementation of the Lightning network. While the transactions are generated by randomly sampling sender-receiver from the Ripple dataset, we use the actual Lightning transaction values. So, the relationship between the transaction volumes with respect to the payment channel capacities is still maintained. We sample one million transactions and verify their success using Ford-Fulkerson, resulting in 363,520 transactions. As the number of nodes used in the simulated Lightning implementation is 25 times smaller than the Ripple, we modify the simulated nodes' participation to 100%, 50%, and 20%. Accordingly, there is a sufficient number of nodes to be participating in the routing process.

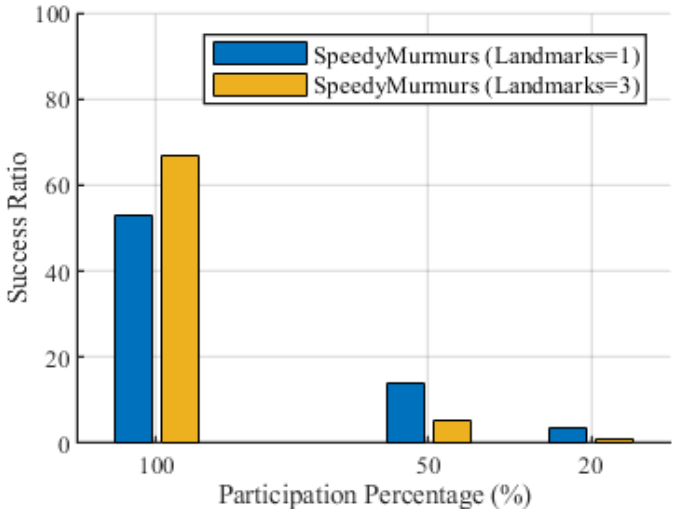


Figure 5-6. SpeedyMurmurs success ratio at different number of landmarks in LN

In order to identify the number of landmarks (i.e., spanning tree roots) in the SpeedyMurmurs algorithm, we compare the routing success ratio at the values of one and three, see Figure 5-6. This parameter is also interpreted as the number of paths. The higher number of landmarks, the higher the success ratio only when all nodes are online. Then, the performance deteriorates at lower participation percentages. For a fair comparison, we select

one landmark in the simulation in the subsequent sections. Since non-participating nodes' impact is the focus of our work, we opt for the figure that achieves the best success ratio at lower participation percentages. The network and simulation parameters are described in Table 5-2.

5.6.2.1 Success ratio

As expected, since the Ford-Fulkerson algorithm is the baseline for success ratio and transaction volumes, it gives the highest success ratio, as shown in Figure 5-7 in all participation percentages. At 100% participation percentage, the gap between the baseline algorithm and the other algorithms (i.e. MARL-Routing, Flash, and SpeedyMurumrs) increases when compared with the same performance metric in the Ripple network in Figure 5-1. This can be justified as a result of the different topology parameters of the simulated Lightning against the Ripple network, as reflected in Table 5-1. The Lightning topology has higher values of clustering coefficient and degrees as well as a lower number of single-link nodes. This gives Ford-Fulkerson and Flash an advantage of finding multiple paths between the source and the destination. On the other hand, MARL-Routing gives a comparable performance to Flash using only one path. As stated in chapter 4, we leave using multiple paths in MARL-Routing for future work.

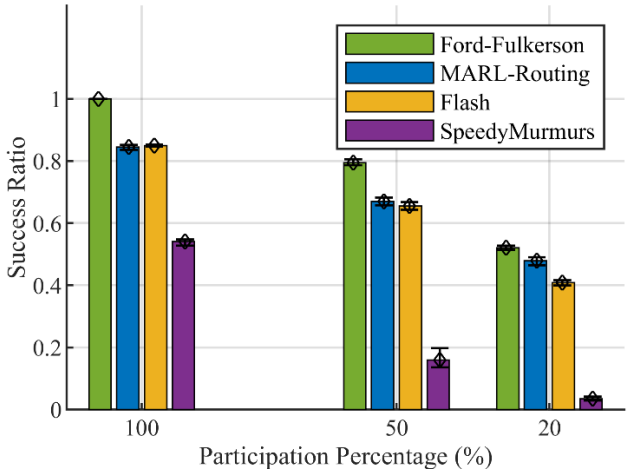


Figure 5-7. Success ratio at various participation percentages in LN

The multiple-path impact in the LN is also in line with the SeedyMurmurs results in Figure 5-6, demonstrating that using three paths via three landmarks enhances the success ratio at 100% participation. However, the same behavior cannot be maintained at lower participation

percentages. This is because all the nodes along the three paths have to be participating, which is more challenging as the participation percentage decreases.

Eventually, our algorithm shows robustness against increasing the number of non-participating nodes. The gap to the baseline algorithm in terms of the success ratio decreases from almost 17% at full nodes' participation to 12.5% at 20% participation percentage. Similar to the Ripple network, a significant decline is observed in the SpeedyMurmurs at participation percentages of 50% and 20%.

5.6.2.2 Average transaction fees

Similar to the Ripple network, the transaction fees of the successful payments are the highest for the Ford-Fulkerson algorithm, see Figure 5-8. The participation percentage effect is insignificant on all the algorithms due to the high average degree and clustering coefficient parameters of the simulated Lightning network. Multiple paths exist between the source and the destination with various link capacities. In case of losing one or two paths, other paths of the same distance and capacities are available to perform the transaction. The higher transaction fees of Flash compared to MARL-Routing indicate that Flash relies more on multi-paths in the transaction routing.

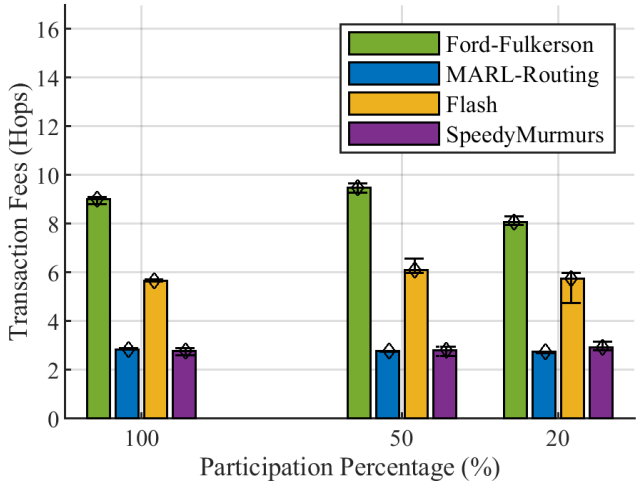


Figure 5-8. Transaction fees of successful transactions in LN

5.6.2.3 Routing efficiency

Similar to the Ripple network, our algorithm relatively combines a high success ratio and a low average path length. Therefore, it is able to surpass the other algorithms in terms of routing efficiency, as shown in Figure 5-9.

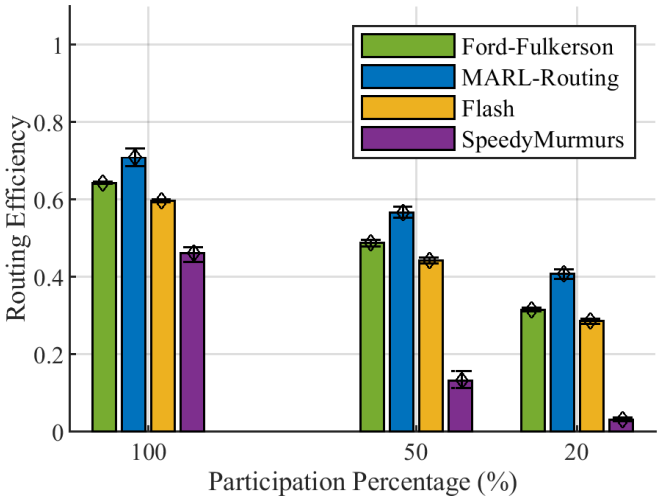


Figure 5-9. Routing efficiency in LN

5.6.2.4 Transaction overhead

Similar to the case of the Ripple network, MARL-Routing surpasses the other fully distributed algorithms, Ford-Fulkerson and Flash, in terms of transaction overhead, at 100% and 50%, as shown in Figure 5-10. MARL-Routing's increase of transaction overhead at lower participation percentages is dominated by the overhead that results from failed transactions which is worsened when combined by the higher connectivity of the LN. This is caused by the on-demand route discovery that will search for all the available paths. At the same time, the other routing algorithms' performance is consistent in all participation percentages.

It is worth mentioning that SpeedyMurmurs shows low transaction overhead upon transaction execution because it relies on a coordinate system (i.e. spanning tree-based). However, it results in an overhead that is logarithmically scaled to the network size, in case of coordinate changes, which is not captured by this metric.

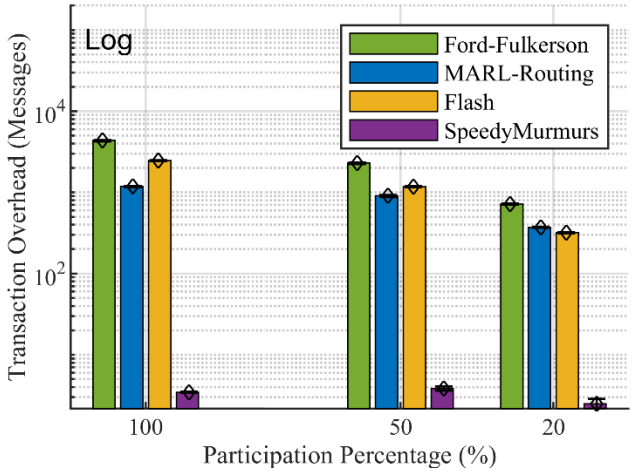


Figure 5-10. Transaction overhead at various participation percentages in LN

5.6.2.5 Transaction maintenance overhead

As depicted in Figure 5-11, the transaction maintenance overhead is a significant enhancement of MARL-Routing over the other fully distributed algorithms, which is similar to the performance in the Ripple network. The transaction maintenance overhead in MARL-Routing is mainly dominated by the on-demand route discovery, only when it is used in the routing decision without relying on the previously stored values.

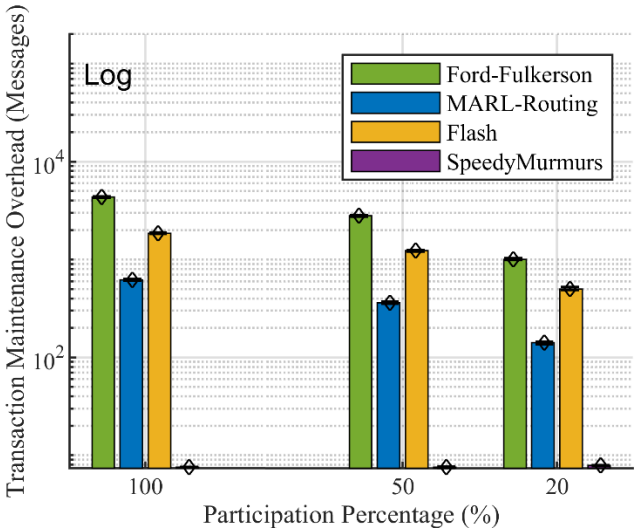


Figure 5-11. Transaction maintenance overhead of successful transactions in LN

5.6.3 Results Overall Discussion

As highlighted earlier in this chapter, the network topology may significantly impact the routing protocol performance. This fact is very present in the simulated results for two real-world network topologies: Ripple and Lightning. Ford Fulkerson is the benchmark algorithm in terms of success ratio. Nevertheless, MARL-Routing manages to surpass the Ford Fulkerson algorithm in the Ripple network at lower participation percentages. The machine learning feature embedded in MARL-Routing helps the algorithm outperform the other protocols as the number of single nodes increases in the network.

The increase of single-link nodes gives MARL-Routing an advantage over the other algorithms. In particular, if the transaction sender, or receiver, is a single-link node, it means it is connected to the payment channel network through one node. In case this node is non-participating, the transaction will immediately fail, and it is better to be repeated in the future. This makes MARL-routing excels as the number of the single-link nodes increases. The number of single-link nodes in the simulated Lightning topology, 23.44%, is less than the same parameter in the simulated Ripple topology 36.43%. Therefore, at 50% and 20% participation percentages, MARL-Routing is not able to perform close to the benchmark algorithm in terms of the success ratio in the Lightning network.

Moreover, MARL-Routing uses a single path in the transaction routing while Ford-Fulkerson and Flash use multiple paths. This gives them an advantage for both algorithms over MARL-Routing, if the transaction value exceeds the payment channel capacities for the available paths. This advantage is quite evident in the LN, where the simulated transactions rely more on multiple paths than the transactions of the Ripple network. Applying multiple paths in MARL-Routing is left for future work.

MARL-routing clearly outperforms the SpeedyMurmurs algorithm in terms of the success ratio. MARL-routing has the highest routing efficiency among the other algorithms under comparison in all participation percentages.

The average successful transaction fees are comparable for both MARL-routing and SpeedyMurmurs, while both slightly outperform Flash. Being a fully distributed method, MARL-routing demonstrates an advantage over Flash and Ford Fulkerson in terms of transaction and maintenance overheads. This is opposite to the landmark-based algorithm, SpeedyMurumus, which experiences massive overhead due to spanning tree modifications.

Eventually, MARL-routing shows a good tradeoff among success ratio, transaction fees, routing efficiency, transaction overhead, and transaction maintenance in both real-world network topologies: the Ripple and the Lighting networks.

5.6.4 Impact of the selection of γ

Figure 5-12 depicts the effect of the discount factor, γ, on the success ratio at the 20% participation percentage in MARL-routing for the LN. It results in the best performance at values around 0.4. When the value is closer to one, the agent prioritizes rewards in the distant future. However, a discount factor closer to zero indicates that only rewards in the immediate future are considered, implying a shallow lookahead. Hence, the performance is best in moderate values in between the two extremes. If the discount factor meets or exceeds 1, the action values may diverge.

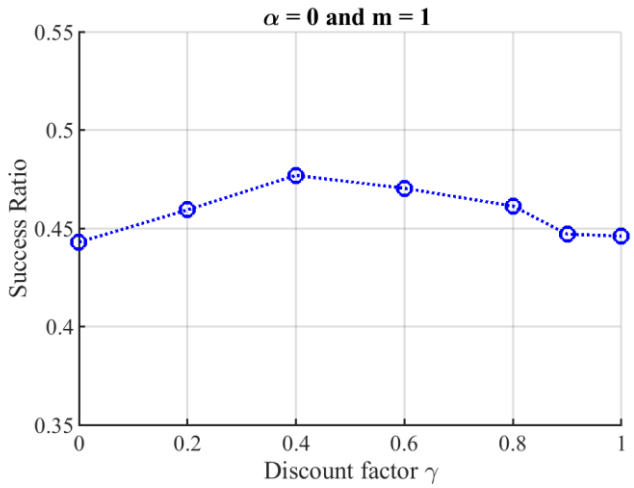


Figure 5-12. Success Ratio vs. discount factor γ

5.7 Chapter Summary

In this chapter, we presented the performance evaluation results of the proposed routing protocol in payment channel networks. The simulations were conducted using two different real-world network topologies and datasets, which were used in previous research works addressing the same problem. We presented the simulation results for both networks showing the impact of the non-participating node percentage on the success ratio, the transaction fees, the routing efficiency, the transaction overhead, and the transaction maintenance overhead. We conducted an overall overall discussion on the results of MARL-routing performance in comparison with maximum flow-, routing table-, and landmark-based routing protocols. We illustrated the change in the routing techniques' performance between both networks regarding

the difference in network topology parameters. Finally, we justified the selected discount factor used in the simulation setup of MARL-routing.

Chapter 6
Conclusion and Future Work

6.1 Thesis Summary and Conclusion

This thesis investigated the recent efforts in the literature that deal with transaction routing in payment channel networks. The usage of PCNs is one of the most promising solutions proposed to solve the blockchain scalability challenge. We introduced a novel machine learning-based technique to find the possible routes between the transaction source and destination. We applied a model-based reinforcement technique and multi-agent reinforcement learning to build an internal model for each node via routing tables to continuously learn and improve the routing efficiency.

The aim of the resulting model is not only limited to suggesting the route of the lowest transaction fees but also to consider the quality of using a specific path. In particular, the algorithm learns to avoid paths that encounter frequent transaction failure due to offline or non-responsive nodes and repeated payment channel imbalance. This is achieved by using on-demand route discovery upon transaction execution that continuously updates the model based on the latest state of the network, characterized to be highly dynamic. Simultaneously, reinforcement learning proposes a solution to network underutilization since congested paths are avoided by the transaction source. Therefore, the node tries other underutilized routes, even if they are of longer paths. Moreover, due to the model built by each node to describe the network, our algorithm is able to suggest the best time in the future to retry the transaction in case of a failure. This is applied randomly in the other protocols.

We modeled the frequent transient and intransient errors, identified as the main reasons for failing payments, by introducing a simulation parameter called participation percentage. We aimed to model the nodes' inability to immediately participate in a successful payment due to being offline, non-responsive, or temporarily lacking adequate funds. Hence, the node is excluded from the path selection, even if it is suggested by the routing protocol.

We verified our algorithm using two real-world network topologies. We ran the simulations on a vast number of transactions. In addition, we applied different values for participation percentages that represent real-world cases. We compared our algorithm to other routing protocols proposed to solve the same problem. We used various performance metrics

to conduct the evaluation, namely, success ratio, transaction fees, routing efficiency, transaction overhead, and transaction maintenance overhead. Our algorithm showed a superior performance that significantly balances all the performance metrics, while this is achieved using only one path for the payment routing.

6.2 Future Research Potential

Since the payment channel network is a thrust area of research to solve the blockchain scalability challenge, considerable improvements and further investigations are to be studied for the proposed protocol as follows.

- On-demand discovery technique: A variety of algorithms perform the path calculations based on the required performance and the forwarding scheme. The best method to perform the path calculations that optimize the routing problem in PCNs needs further assessment.
- Multi-path routing: The feature of multi-path payment is applicable in payment channel networks. The utilization of this feature in the routing can potentially result in higher success ratio and transaction volume.
- Privacy guarantees in PCNs: A public blockchain does not entail strong privacy guarantees. The link between a sender and receiver of payments as well as tracing back the origin of coin can be tracked on a public blockchain. Similarly, broadcasting a transaction may reduce the privacy aspects of a PCN. Also, probing messages may impose privacy risks. Privacy risks have to be thoroughly assessed and addressed in used routing protocols.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” October, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[2] A. Webb, “8 Tech Trends to Watch in 2016,” *Harvard Business Review*, Dec. 08, 2015. [Online]. Available: https://hbr.org/2015/12/8-tech-trends-to-watch-in-2016.

[3] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, "Bitcoin and cryptocurrency technologies: a comprehensive introduction." *Princeton University Press*, 2016 Jul 19.

[4] C. Catalini and J. S. Gans, “Some Simple Economics of the Blockchain,” *Social Science Research Network*, Rochester, NY, SSRN Scholarly Paper ID 2874598, Apr. 2019.

[5] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, 2016 Jun.

[6] D. G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, no. 151, pp. 1-32, 2014.

[7] “Ripple: Global payment solutions-instant processing.” [Online]. Available: https://ripple.com/.

[8] “The strategic business value of the blockchain market | McKinsey.” [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/blockchain-beyond-the-hype-what-is-the-strategic-business-value.

[9] F. Casino, T. K. Dasaklis, and C. Patsakis, “A systematic literature review of blockchain-based applications: Current status, classification and open issues,” *Telematics and Informatics*, vol. 36, pp. 55–81, Mar. 2019.

[10] K. Christidis and M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[11] “The Energy Web Blockchain – Energy Web Foundation.” [Online]. Available: https://energyweb.org/blockchain/.

[12] H. Kadry, “Blockchain Applications in Midstream Oil and Gas Industry,” *International Petroleum Technology Conference*, Jan. 2020.

[13] M. Haferkorn and J. M. Quintana Diaz, “Seasonality and Interconnectivity Within Cryptocurrencies - An Analysis on the Basis of Bitcoin, Litecoin and Namecoin,” in *Enterprise Applications and Services in the Finance Industry*, Cham, 2015, pp. 106–120.

[14] J. Poon and T. Dryja, “The bitcoin lightning network: Scalable off-chain instant payments," Jan. 2016. [Online]. Available: https:// bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf.

[15] A. Hafid, A. S. Hafid and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE Access*, vol. 8, pp. 125244-125262, 2020.

[16] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, “SoK: Layer-Two Blockchain Protocols,” 2019. [Online]. Available: http://eprint.iacr.org/2019/360

[17] “Raiden network.” [Online]. Available: https://raiden.network/.

[18] “*CoinDesk*: A guide to saving on bitcoin’s high transaction fees,” Feb. 26, 2021. [Online]. Available: https://www.coindesk.com/saving-bitcoin-high-transaction-fees.

[19] F. Waugh and R. Holz, “An empirical study of availability and reliability properties of the Bitcoin Lightning Network,” *arXiv:2006.14358 [cs]*, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.14358.

[20] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy and O. Osuntokun, "Flare: An approach to routing in lightning network white paper,” July, 2016. [Online]. Available: https://pdfs.semanticscholar.org/4392/ 166a1194010c844ec915694fd5c56da94301.pdf

[21] J. J. Sikorski, J. Haughton, and M. Kraft, “Blockchain technology in the chemical industry: Machine-to-machine electricity market,” *Applied Energy*, vol. 195, pp. 234–246, Jun. 2017.

[22] A. M. Antonopoulos "Mastering Bitcoin: unlocking digital cryptocurrencies," O'Reilly Media, Inc."Dec. 2014.

[23] “Blockchain.com Explorer | BTC | ETH | BCH.” [Online]. Available: https://www.blockchain.com/explorer.

[24] V. Buterin, “A next generation smart contract & decentralized application platform,” *Ethereum Foundation*, 2013.

[25] “Sharding-FAQs,” *Ethereum Wiki*. [Online]. Available: https://eth.wiki/sharding/Sharding-FAQs.

[26] “[ANNOUNCE] Micro-payment channels implementation now in bitcoinj.” [Online]. Available: https://bitcointalk.org/index.php?topic=244656.0.

[27] “bitcoinj.” [Online]. Available: https://bitcoinj.org/.

[28] S. Roos, P. Moreno-Sanchez, A. Kate and I. Goldberg, “Settling payments fast and private: Efficient decentralized routing for path-based transactions,” in *Proc. Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2018.

[29] P. Wang, H. Xu, X. Jin and T. Wang, "Flash: Efficient dynamic routing for offchain networks," in *Proc. the 15th Int. Conf. on Emerging Networking Experiments and Technologies (CoNEXT '19)*, New York, NY, USA, Dec. 2019, pp. 370–381

[30] "Global Cryptocurrency Market Charts," *CoinMarketCap*. [Online]. Available: https://coinmarketcap.com/charts/.

[31] "Real-Time Lightning Network Statistics." [Online]. Available: https://1ml.com/statistics

[32] R. Yu, G. Xue, V. T. Kilari, D. Yang and J. Tang, "Coinexpress: A fast payment routing mechanism in blockchain-based payment channel networks," in *Proc. 27th Int. Conf. on Computer Communication and Networks (ICCCN)*, July, 2018, pp. 1-9.

[33] "Working with micropayment channels." [Online]. Available: https://bitcoinj.org/working-with-micropayments.

[34] C. Decker and R. Wattenhofer, "A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels," in *Stabilization, Safety, and Security of Distributed Systems*, Cham, 2015, pp. 3–18.

[35] R. Khalil and A. Gervais, "Revive: Rebalancing Off-Blockchain Payment Networks," 823, 2017. [Online]. Available: http://eprint.iacr.org/2017/823

[36] L. M. Subramanian, G. Eswaraiah, and R. Vishwanathan, "Rebalancing in Acyclic Payment Networks," in *2019 17th International Conference on Privacy, Security and Trust (PST)*, Aug. 2019, pp. 1–5.

[37] G. Malavolta, P. Moreno-Sanchez, A. Kate and M. Maffei, "SilentWhispers: Enforcing security and privacy in decentralized credit networks," in *Proc. Network and Distributed System Security Symposium (NDSS)*, Feb. 2017.

[38] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. Fanti and P. Viswanath, "Routing cryptocurrency with the spider network," in *Proc. the 17th ACM Workshop on Hot Topics in Networks*, Redmond, WA, USA, Nov. 2018, pp. 29–35.

[39] C. A. Sunshine, "Source routing in computer networks," *SIGCOMM Comput. Commun. Rev.*, vol. 7, no. 1, pp. 29–33, Jan. 1977.

[40] L. R. Ford, "Network Flow Theory," Jan. 1956. [Online]. Available: https://www.rand.org/pubs/papers/P923.html.

[41] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399-404, 1956.

[42] E. Rohrer, J.-F. Laß and F. Tschorsch, “Towards a concurrent and distributed route selection for payment channel networks,” in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, Springer, Cham, Aug. 2017, pp. 411-419.

[43] S. Mazumdar, S. Ruj, R. G. Singh and A. Pal, "HushRelay: A privacy-preserving, efficient, and scalable routing algorithm for off-chain payments," in *Proc. IEEE Int. Conf. on Blockchain and Cryptocurrency (ICBC)*, Toronto, ON, Canada, May 2020, pp. 1-5.

[44] A. V. Goldberg and R. E. Tarjan, “A new approach to the maximum-flow problem,” *J. ACM*, vol. 35, no. 4, pp. 921–940, Oct. 1988.

[45] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," *MIT Press*, 2009.

[46] C. Y. Lee, “An Algorithm for Path Connections and Its Applications,” *IRE Trans. on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.

[47] V. Sivaraman *et al.*, “High Throughput Cryptocurrency Routing in Payment Channel Networks,” in *Proc. 17th Symposium on Networked Systems Design and Implementation*, 2020, pp. 777-796.

[48] P. Hoenisch and I. Weber, “AODV–Based Routing for Payment Channel Networks,” in *Proc. Int. Conf. on Blockchain*, Jun. 2018, pp. 107-124.

[49] O. Osuntokun, “[Lightning-dev] AMP: Atomic Multi-Path Payments over Lightning,” Feb. 06, 2018. [Online]. Available: https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html

[50] R. Boutaba *et al.*, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” *Journal of Internet Services and Applications*, vol. 9, no. 1, Dec. 2018.

[51] R. V. Kulkarni, A. Förster, and G. K. Venayagamoorthy, “Computational Intelligence in Wireless Sensor Networks: A Survey,” *IEEE Communications Surveys Tutorials*, vol. 13, no. 1, pp. 68–96, First 2011.

[52] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1996–2018, Fourthquarter 2014.

[53] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," *MIT Press*, 2018.

[54] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence* research, vol. 4, pp. 237-285, May 1996.

[55] J. Dowling, E. Curran, R. Cunningham and V. Cahill, “Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing,” *IEEE Trans. on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 35, no. 3, pp. 360–372, May 2005.

[56] J. A. Boyan and M. L. Littman, “Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach,” *Advances in neural information processing systems*, pp. 671-678, 1994.

[57] R. Bellman, "Dynamic Programming," *Princeton University Press*, 1959.

[58] S. P. M. Choi and D.-Y. Yeung, “Predictive Q-Routing: A Memory-based Reinforcement Learning Approach to Adaptive Traffic Control,” *Advances in Neural Information Processing Systems*, pp. 945-951, 1996.

[59] P. Stone, and M. Veloso. "Team-partitioned, opaque-transition reinforcement learning." *Proceedings of the third annual conference on Autonomous Agents*, pp. 206-212, 1999.

[60] R. I. Brafman and M. Tennenholtz, “R-MAX-A general polynomial time algorithm for near-optimal reinforcement learning,” *Journal of Machine Learning Research*, vol. 3, pp. 213–231, Oct. 2002.

[61] A. L. Strehl, L. Li and M. L. Littman, "Reinforcement learning in finite MDPs: PAC analysis," *Journal of Machine Learning Research*, vol. 10, no. 11, Nov. 2009.

[62] Y. Yang and J. Wang, “Routing metrics design for multihop wireless networks,” in *Proc. of Communication & Networking Technology (CNT) Symposium*, 2007.

[63] Zheng Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” *IEEE J. Select. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.

[64] Qingming Ma and P. Steenkiste, “On path selection for traffic with bandwidth guarantees,” in *Proceedings 1997 International Conference on Network Protocols*, pp. 191–202, Oct. 1997.

[65] J. L. Sobrinho, “Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet,” in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, Apr. 2001, vol. 2, pp. 727–735 vol.2.

[66] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," 3rd ed. Upper Saddle River, Pearson, 2009.

[67] R. Albert and A.-L. Barabási, “Statistical mechanics of complex networks,” *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 47–97, Jan. 2002.

[68] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, no. 6684, Art. no. 6684, Jun. 1998.

[69] A. Iamnitchi, M. Ripeanu, and I. Foster, “Small-world file-sharing communities,” in *IEEE INFOCOM 2004*, Mar. 2004, vol. 2, pp. 952–963 vol.2.

[70] T. Neudecker, P. Andelfinger, and H. Hartenstein, “Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network,” in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 358–367.

[71] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, “Discovering Bitcoin’s Public Topology and Influential Nodes,”, May 2015. [Online]. Available: *https://allquantor. at/blockchainbib/pdf/miller2015topology. pdf*

[72] Xiao Fan Wang and Guanrong Chen, “Complex networks: Small-world, scale-free and beyond,” *IEEE Circuits Syst. Mag.*, vol. 3, no. 1, pp. 6–20, 2003.

[73] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, K. C. Claffy, and A. Vahdat, “The Internet AS-Level Topology: Three Data Sources and One Definitive Metric,” *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 17–26, Jan. 2006.

[74] B. Schiller and T. Strufe, “GTNA 2.0-A framework for rapid prototyping and evaluation of routing algorithms,” in *Proc. Summer Computer Simulation Conf. (SummerSim)*, 2013.

[75] “Charts,” *Coin Metrics*, Jul. 03, 2020. [Online]. Available: https://coinmetrics.io/charts/

[76] “Got 10 BTC? You’re now in the top 0.5% of 30 million bitcoin addresses,” *Cointelegraph*, Jul. 03, 2020. [Online]. Available: https://cointelegraph.com/news/got-10-btc-youre-now-in-the-top-05-of-30-million-bitcoin-addresses

[77] *"*Offchain-routing-traces-and-code,*" GitHub. [Online]. Available: https://github.com/NetX-lab/Offchain-routing-traces-and-code*

[78] A. Muscoloni, J. M. Thomas, S. Ciucci, G. Bianconi and C. V. Cannistraci “Machine learning meets complex networks via coalescent embedding in the hyperbolic space,” *Nat. Commun.*, vol. 8, no. 1615, 2017.