Layer 2 Protocols in Bitcoin

Luis Esteban Oleas-Chávez

Layer 2 Protocols in Bitcoin

Luis Esteban Oleas-Chávez

September, 2024
Version: Final

Universitat Autònoma de Barcelona

Departament d'Enginyeria de la Informació i de les Comunicacions
Universitat Autònoma de Barcelona
Línia de recerca: Codificació, Compressió i Seguretat

Layer 2 Protocols in Bitcoin

PhD in Computer Science

Luis Esteban Oleas-Chávez

A dissertation submitted to the Universitat Autònoma de Barcelona in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in Computer Science.

Supervisors: Dr. Jordi Herrera-Joancomartí
Department of Information and Communications Engineering
Universitat Autònoma de Barcelona

Dr. Cristina Pérez-Solà
Department of Information and Communications Engineering
Universitat Autònoma de Barcelona

September, 2024

Universitat Autònoma de Barcelona

Departament d'Enginyeria de la Informació i de les Comunicacions

Layer 2 Protocols in Bitcoin

Luis Esteban Oleas-Chávez

Supervisors: Dr. Jordi Herrera-Joancomartí and Dr. Cristina Pérez-Solà

September, 2024

Submitted to Universitat Autònoma de Barcelona in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Committee:

Dr. Jordi Castellà Roca
Dr. Guillermo Navarro Arribas
Dr. Magdalena Payeras Capellà
Dr. Carles Garrigues Olivella (Substitute)
Dr. Helena Rifà Pous (Substitute)

Supervisors:

Dr. Jordi Herrera-Joancomartí
Dr. Cristina Pérez-Solà

LaTeXstyle:
`http://cleanthesis.der-ric.de/`.

Declaration

We certify that we have read this thesis entitled "Layer 2 Protocols in Bitcoin" and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Cerdanyola del Vallès, September, 2024

Dr. Jordi Herrera-Joancomartí (Advisor)	Dr. Cristina Pérez-Solà (Advisor)

Committee:

Dr. Jordi Castellà Roca
Dr. Guillermo Navarro Arribas
Dr. Magdalena Payeras Capellà
Dr. Carles Garrigues Olivella (Substitute)
Dr. Helena Rifà Pous (Substitute)

Program: PhD in Computer Science
Department: Departament d'Enginyeria de la Informació i de les Comunicacions

Abstract

As a decentralized cryptocurrency, Bitcoin builds on the blockchain by distributing the ledger among peers to share a global awareness of transactions made on the network without relying on intermediaries. However, the recording of transactions on the blockchain lacks dynamism and is far from the speed offered by well-established commercial payment systems such as Visa and Paypal.

Due to the discouragement of adopting Bitcoin as a payment method, a novel parallel payment network, referred to as the Lightning Network (LN) for Bitcoin, burst onto the scene to boost on-chain transactionality by making off-chain payment transactions fast and scalable. LN therefore overcomes the hurdles of scalability and dynamic financial processing, as well as lessening the burden on the Bitcoin network chain.

Transitioning this restricted Bitcoin network into a viable payment method depends on how LN compares to traditional payment systems. We aim to analyze LN and propose improvements on two fronts. First, we provide an analytical approach to define the value of some contract parameters. Finally, we determine alternative metrics to evaluate the centrality of this peer-to-peer network (P2P).

Thus, to provide an improvement to LN, we evaluated the impact of adjusting the contract parameters used in multi-hop payments into the security and performance of the network. Also, we proposed a graph-based model for the LN, and a set of centrality metrics to measure node centrality within this model. Hence, the main goal behind this research is to enhance the reliability of well-established protocols deployed on the LN.

Resumen

Como criptomoneda descentralizada, Bitcoin se basa en la cadena de bloques al distribuir el libro de contabilidad entre pares para compartir un conocimiento global de las transacciones realizadas en la red sin depender de intermediarios. Sin embargo, el registro de las transacciones en blockchain carece de dinamismo y está lejos de la velocidad que ofrecen los sistemas de pago comerciales consolidados como Visa y Paypal.

Debido al desaliento de adoptar Bitcoin como método de pago, una novedosa red de pagos paralelos, conocida como Lightning Network (LN) para Bitcoin, irrumpió en escena para impulsar la transaccionalidad dentro de la cadena al hacer que las transacciones de pago fuera de la cadena sean rápidas y escalables. Por lo tanto, LN supera los obstáculos de la escalabilidad y el procesamiento financiero dinámico, además de reducir la carga sobre la cadena de la red Bitcoin.

La transición de esta red Bitcoin restringida a un método de pago viable depende de cómo se compara LN con los sistemas de pago tradicionales. Nuestro objetivo es analizar LN y proponer mejoras en dos frentes. Primero, proporcionamos un enfoque analítico para definir el valor de algunos parámetros del contrato. Finalmente, determinamos métricas alternativas para evaluar la centralidad de esta red peer-to-peer (P2P).

Por lo tanto, para brindar una mejora a LN, evaluamos el impacto de ajustar los parámetros del contrato utilizados en pagos de múltiples saltos en la seguridad y el rendimiento de la red. Además, propusimos un modelo basado en gráficos para LN y un conjunto de métricas de centralidad para medir la centralidad de los nodos dentro de este modelo. Por lo tanto, el objetivo principal de esta investigación es mejorar la confiabilidad de protocolos bien establecidos implementados en la LN.

Resum

Bitcoin és una criptomoneda descentralitzada que basa el seu funcionament en la utilització d'una cadena de blocs que distribueix, entre tots els participants del sistema, el llibre comptable on hi figuren totes les transaccions realitzades a la xarxa. L'interès principal d'aquest sistema és que permet el manteniment d'aquest registre de transaccions sense dependre d'intermediaris. No obstant això, l'enregistrament de les transaccions en la cadena de blocs no ofereix el dinamisme esperat i està lluny de la velocitat que ofereixen sistemes de pagament comercials consolidats, com ara Visa o Paypal.

Aquetes mancances indicades fan que l'adopció de Bitcoin com a mètode de pagament sigui poc viable i, per aquest motiu, s'ha desenvolupat una xarxa de pagament de nivell superior, anomenada Lightning Network (LN) que permet augmentar el volum de transaccions que el sistema pot gestionar fent transaccions de pagament fora de la cadena, oferint una opció ràpida i dinàmica i, per tant, superant així els obstacles d'escalabilitat de la xarxa Bitcoin.

En aquest treball pretenem analitzar la LN i proposar millores en dos fronts. En primer lloc, oferim un enfocament analític per definir el valor d'alguns paràmetres del sistema. Finalment, determinem mètriques alternatives per avaluar la centralitat de la xarxa de pagaments que forma la LN.

Així, per oferir una millora a la LN, avaluem l'impacte d'ajustar els paràmetres dels nodes de la LN utilitzats en els pagaments multi-salt en la seguretat i el rendiment de la xarxa. A més, proposem un modelat de la LN basat en teoria de grafs que permet definir un conjunt de mètriques de centralitat per determinar la centralitat dels nodes de la xarxa dins d'aquest model. Aquestes mesures permeten establir el grau de centralització dels diferents nodes i de la LN en el seu conjunt.

Acknowledgement

Another professional and personal stage in my life has to come to an end that could not have been possible without the help and support of many people. I want to take a moment to thank each of them for their help and advice.

First, I would like to thank my advisor, Dr. Jordi Herrera-Joancomartí, for allowing me to research with him and for his mentoring and support. Furthermore, thanks for his guidance in overcoming the struggles during this student phase. Also, I would like to thank my co-advisor, Dr. Cristina Pérez-Solà, for her wisdom in addressing me in the right direction when I lost track of my research and her patience in explaining concepts or ideas that got me off guard. Special thanks to the Ph.D. follow-up committee for the advice given after each follow-up presentation. Also, their opinions and points of view on the progress reports submitted gave me the confidence to continue my work.

Finally, a warm thank you to my family, to my parents Luis† and Susana† for being the source of who I am today, their dedication and hard work are the foundation of my determination to achieve my goals, as well as to my sisters María Luisa y Diana for their words of support and encouragement. Above all, I want to thank the love of my life, my wife Jacky, who has been my companion, my friend, my inspiration, and my strength along this journey, which, together with my sweet Lizzy†, are my world.

List of Figures

List of Tables

Contents

Part I

Preliminaries

1 Introduction

Before software should be reusable, it should be usable.

— **Ralph Johnson**
(Computer Scientist, UIUC)

1.1 Introduction

The launch of Bitcoin in 2008 led to the rise of the Lightning Network (LN) more than a decade later to ease the burden on its blockchain. A transition was crucial since, shortly, the general acceptance of Bitcoin as a payment method could be affected by its efficiency [1, 2, 3]. In Bitcoin, a group of transactions, a nonce, and the hash of a preceding block define a block. The maximum size for each block is 1 MB [4], which causes Bitcoin to suffer scalability problems since around seven transactions per second (tps) can be allocated with this block size [5]. Furthermore, Bitcoin's transaction processing performance is quite poor, since block generation takes roughly 10 minutes [6].

Transaction fees are part of the Bitcoin system to incentivize block confirmation. Higher fee transactions are more likely to be included in a block for mining compared to those with lower or no fees that may be delayed or not processed. This confirmation process introduces competitiveness in the system, where miners try to include those transactions with higher fees. However, before submitting a transaction, a user must specify the transaction fee, which could result in overspending with high fees or long confirmation times with low fees. To outbid other Bitcoin users who pay lower fees when traffic is heavy, users frequently bid up the transaction fees.

A drawback for which Bitcoin is known is its slow transaction throughput [7]. Payments in Bitcoin take a random time until they are confirmed, that is, from when a block is generated until it is added to the blockchain. This randomness comes from the limited block size, the fluctuation in block solve timing, and the free selection of unconfirmed transactions by the miners [8]. It results in not knowing how long it takes to confirm a transaction. To address these drawbacks, LN emerged as a feasible alternative in which its deployment as a secondary layer compromises neither the decentralized network nor security nor privacy. Therefore, LN can be deployed as a side-kick network without interfering with the normal operation of cryptocurrencies, as in the case of Bitcoin.

Notwithstanding, LN has some drawbacks such as payment reliability, centrality, and routing [9], privacy [10], skewness [11] and overload [12] issues, to name a few. This thesis aims to improve the security and performance of the already established LN and raise awareness about the centrality of the network. Our efforts aim to determine how the structure of such a network is shaped to ensure it is a reliable payment method. Similarly, in this thesis, we propose new metrics to measure the centrality of LN. By analyzing node centrality, we can determine if a network is decentralized or controlled by a handful of users and how resilient it is to different attacks. Also, we provide a guide on how to select the values of the LN client parameters.

1.2 Research Objectives

Since there is a desire to increase trust in LN as a payment method, this thesis aims to improve its perception of security and performance. As well as give insights on how to comprehend the centrality of this parallel payment network. We propose the following objectives to achieve our main goal:

- To understand the structure of LN and the functionality of its payment method.
- To assess the impact of adjusting LN contract parameters in terms of security and performance.
- To outline, model and analyze the metrics that best define the centrality of LN.

1.3 List of Contributions

This thesis provides the following contributions:

- The definition of a set of metrics for assessing the performance and security of the Lightning Network.

- The application of the above mentioned metrics to analyze the performance and security of the Lightning Network with respect to different client configuration parameters.

- The proposal of optimal configuration parameters taking into account both security and performance.

- The definition of a graph-based model for representing the Lightning Network.

- The proposal of metrics to measure node centrality in the Lightning Network using the previously defined model.

- The evaluation of node centrality over real Lightning Network snapshots.

1.4 List of Publications

The publications produced by this thesis are listed below:

- Oleas-Chávez Luis Esteban, Pérez-Solà Cristina, and Herrera-Joancomartí Jordi. "On the Selection of the LN Client Implementation Parameters." In Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2020 International Workshops, DPM 2020 and CBT 2020, Guildford, UK, September 17–18, 2020, Revised Selected Papers 15, pp. 305-318. Springer International Publishing, 2020.

- Oleas-Chávez Luis Esteban, Pérez-Solà Cristina, and Herrera-Joancomartí Jordi. "Apples and Oranges: On How to Measure Node Centrality in Payment Channel Networks." IEEE Access 10 (2022): 55469-55487.

1.5 Thesis Structure

The structure of the rest of this thesis is as follows:

The thesis is split into three main sections, a preliminary one (Chapters **2**, **3** and **4**) introduces the preliminary concepts and definitions on which this thesis is based in terms of Bitcoin and LN, as well as a slight review of the state of the art on these topics.

Subsequently, a contributions section (Chapters **5** and **6**) provides some analysis on LN related to multi-hop payment channels and network topology. Chapter 5 describes scenarios with different parameters of the contracts to evaluate the impact of these parameters of the network in terms of performance and security. Afterwards, Chapter 6 presents a different perspective to evaluate the centrality in LN using classical and alternative centrality metrics and network properties. The last section, Chapter **7** presents the conclusions of the thesis and discusses the future work.

Bitcoin and Lightning Network 2

Computer science empowers students to create the world of tomorrow.

— **Satya Nadella**
(CEO of Microsoft)

This chapter outlines the basic definitions of the main topics, specifically Bitcoin and LN, in which this research fits into the remainder of this thesis. First, we will present a brief overview of Bitcoin and its scalability problem, given its limited capacity to handle a large number of transactions. Afterward, we will delve into the most relevant aspects of LN, such as nodes, channels, and policies, to name a few.

2.1 Bitcoin

Bitcoin, introduced in 2009, was the first decentralized cryptocurrency [13] as an open-source project made by an individual using the alias Satoshi Nakamoto. Bitcoin records transfer history on a blockchain, an append-only data structure composed of a chain of blocks. Users of the system share this data structure that resembles a ledger, in which each entry is a transaction that tracks the transfer value between users.

Bitcoin relies on its complete detachment from third parties, for instance, financial institutions, to make online payments. Also, it solves the dilemma of trusting a central authority by publicly announcing transactions where the participants agree on the order history of transactions received. Based on cryptographic proof, the Bitcoin payment system allows users to make payments directly through a network without a trusted party. This payment system is helpful when entirely irreversible transactions are needed, a feature that standard payment do not offer.

Irreversible transactions would protect both sellers from fraud as well as buyers through routinely implemented escrow mechanisms. This P2P network, in which Bitcoin settles its implementation, solves the double-spending problem where the same coin is paid simultaneously twice or more. The timestamp of transactions solves such a problem by converting transactions into a continuous hash-based Proof-Of-Work (POW) chain. The only way to change the record formed in this process is by redoing the POW, which consists of miners solving a very computationally intensive puzzle that costs them money in the form of high consumption of energy and computational resources.

Bitcoin overcomes the consensus problem when reaching an agreement on a decentralized network using POW. Such achieved consensus in the system, even with potentially malicious participants, allows solving the double-spending problem. A transaction happens when an owner transfers the digital signature of the previous transaction hash with the subsequent owner's public key and adds it to the coin's tail. Therefore, a chain of digital signatures forms a coin. Similarly, to validate the chain of ownership, the payee may check the signatures; however, it cannot prove that the payer did not double-spend the coin. The confirmation of a transaction occurs after its inclusion in the blockchain inside a block created in a mining process performed by miners.

As long as the number of trusted nodes outnumbers any group of nodes that attack cooperatively, the system is secure overall. Thus, Bitcoin will become more ubiquitous due to its ability to enforce immutability on a sequentially ordered append-only ledger [14].

2.1.1 Scripting language

Within the Bitcoin system, the Bitcoin scripting language, known as ***SCRIPT***, is a loop-free, non-Turing-complete, stack-based language [15] that enables users to build personalized Bitcoin transactions and smart contracts. Since the stack is the foundation of the language, it has no conditional statements or variables. The execution of operations is carried out exactly once, linearly following the Last In First Out (LIFO) scheme [16] without taking backward leaps. It is a non-Turing complete script because it is not feasible to compute arbitrarily powerful functions and lacks loops and conditional statements. Based solely on the number of instructions in the script, it can help to estimate the processing time. Also, the scripts that users submit in the transactions must be executed directly by the miners. Users cannot submit scripts that could run indefinitely or in an infinite loop. Depending on the number of instructions it contains, the Bitcoin script will always execute in a finite number of steps. The script length after the instruction pointer sets the upper constraint on execution time. For nodes validating blocks, this constraint protects against denial of service attacks [17].

The Bitcoin script language also allows the definition of smart contracts modeled as stateless cryptographic protocols [18]. The script serves various purposes, including enabling the validation of transactions, implementing sophisticated protocols and decentralized applications, and offering additional functionality to blockchain scripts [19, 20]. Instead of simply transferring funds from one address to another, the script allows users to set restrictions on how the funds may be used, enabling more intricate and adaptable transactions. Users can set up more complex payment channels, time-locked transactions, and multi-signature wallets with Bitcoin scripts. Due to its stringent design principles, the script offers a high level of security, yet it is relatively easy compared to other smart contract systems such as Ethereum. Formal verification of critical scripts is necessary to validate financial transactions. In that sense, the validation scheme consists of two scripts [21]. The former is an *input script* that provides data and credentials that authorize the transaction. The latter is an *output script*, the composition of which serves both to establish the validation system and to verify that the data supplied enables the transaction. The following are the primary properties of the Bitcoin programming language:

- All Bitcoin scripts are limited to two possible results. It may yield an error, or it may run successfully. In the transaction validation process, the entire transaction will be deemed invalid and should not be approved into the blockchain if there is any error during the script's execution.

- Since a single byte represents each instruction, the Bitcoin scripting language is incredibly compact, with just 256 instructions possible. As of right now, 75 of them are reserved, and 15 are disabled; hence, they are meaningless and may be assigned at a later time.

- The instructions that manage basic logic and arithmetic, error-throwing, and cryptography management, such as signature verification and hash functions, are all included in the Bitcoin scripting language.

As part of the script, it has two different instructions: the first is $data\ instructions$, which contains some value, such as the sender's signature or the public key that generated that signature, surrounded by angular brackets. The other is operational codes (OP_CODE) with specific operations such as signature validation, hash functions, and transaction validation, to name a few. OP_CODEs modify the value on the stack when pushing data or performing tasks within a pubkey script or signature script, as well as add their outcome there. The set of OP_CODE controls the behavior of the transactions in this stack-based language. Based on OP_CODE functionality, they are classified into numerous groups:

- ***Arithmetic:*** User can execute mathematical operations as is multiplication with OP_MUL opcode.

- ***Cryptography:*** These opcodes can be used by users to apply cryptographic instructions to the target data. One such is $OP\_HASH256$, a set of instructions that uses the cryptographic technique SHA-256 to hash the input twice to increase security.

- ***Data manipulation:*** The manipulation parameters for the given data can be specified using these opcodes. $OP\_NUM2BIN$, which transforms a numeric number into a predefined-length byte sequence, illustrates manipulation.

- ***Flow Control:*** These opcodes are useful to ascertain the script's flow. For instance, OP_VERIFY flags a transaction as invalid when the top stack is false.

- ***Bitwise logic:*** This collection of opcodes is useful when a command that executes in response to predefined input data is required. OP_INVERT is one example, which flips every bit in an input.

- ***Constants:*** This class of opcodes allows one to put a given amount of data into the stack. One such is $OP\_1NEGATE$, which pushes the number -1 into the stack by executing a command.

- ***Stack:*** Users use this opcode to rearrange objects on the stack as they see fit. As one example OP_2DROP eliminates the top two stack elements.

The main advantage of the script is its simple and non-Turing complete design that simplifies the validation process of closed scripts linked to transactions. However, it also suffers from disadvantages, as with the simplicity of the script. Its drawback is that some script flaws might lead to the loss of bitcoins. This flaw has been systematically investigated and found to exist in various forms [22]. Despite their initial lack of Turing completeness, it has also been proved that Bitcoin scripts can achieve it with certain limitations. Taking this into account, it could make it possible to add more intricate applications and protocols on top of Bitcoin transactions [23, 19]. Formal verification tools such as ScriFy have been created to improve the security and accuracy of Bitcoin scripts. These tools offer a framework for validating script programs and reducing language-related risks [23].

To prevent the creation of scripts that deviate from the ones deemed conventional, several restrictions have been put in place in practice. Some of these restrictions are: i) a 512 byte restriction per element and 10,000 byte limit per script on the length of the scripts [24] ii) a large number of disabled opcodes [25] iii) a block size restriction, currently set at around 1MB [26] iv) restriction on the maximum number of opcodes, which is 201 for the majority of them [27]. Nevertheless, it seems that the stack size is not restricted, which is technically irrelevant. The reason is that it is implicitly constrained by the script length limitation [17].

2.1.2 Scalability Problems

Today, transactions carried out by consumers require agile payment methods that must be effective in a fraction of the time. Although Bitcoin relies on the broadcast of each transaction between peers, eventually, this mechanism could limit financial processing. It may ultimately decrease its acceptance as a reliable payment method in global commerce. This drawback becomes more evident when all nodes in the network are aware of every global transaction.

When comparing Bitcoin to the Visa payment system, the difference in payment processing is quite substantial. Visa [28] at its peak can handle 65,000 transactions per second (tps) and, on average, 24,000 tps, which is about 150 million transactions each day. Instead, the Bitcoin financial processing problem becomes more acute because size and frequency restrain the extensive use of blocks (records that contain the transaction) as a payment method. On average, it takes 10 minutes to confirm each new transaction. Moreover, the confirmation processing limits transactionality by increasing fees and slowing down its normal handling. This process is estimated at 3.3 and 7 tps [7] with a one-megabyte block limit.

As a comparison of transactional capacity, it would take about 11 gigabytes per bitcoin block approximately every ten minutes, with unlimited block sizes and 300 bytes per bitcoin transaction on average, to match the maximum transaction volume for Visa of around 65,000 tps. This transactionality represents more than 600 terabytes of data per year. Based on this comparison, most Bitcoin network participants would not be able to store the whole blockchain or even operate with adequate bandwidth. If the acceptance of this payment system were global, it could result in a centralized network with only the nodes and miners who can afford it, or it would crash the network in the worst case.

Therefore, as of today, the Bitcoin network would not surpass the transactionality of Visa and PayPal. Especially in the case of miner centralization, where few validators could guarantee the accuracy of the ledger, and there would be few participants validating the blockchain as a result of the mining process.

2.2 Lightning Network

For Bitcoin to reach a higher number of transactions per second than Visa, transaction processing must occur outside the blockchain. In that sense, LN is a second-layer protocol for Bitcoin designed to route micropayments between parties via peer-to-peer two-way payment channels. The LN protocol, which facilitates faster and more cost-effective transactions outside the blockchain, aims to alleviate Bitcoin congestion and enhance scalability [29, 30]. To achieve such properties, LN functions by enabling users to establish direct payment channels with one another, allowing for fast and secure transactions with reduced fees [31]. Therefore, the main goal of LN, proposed by Joseph Poon and Thaddeus Dryja, is to solve throughput, costs, and slow transaction time of Bitcoin[5].

Some of the main advantages that LN offers are scalability, low energy requirements, support for micropayment, and speed. The Bitcoin blockchain scalability issues stem from block size limitations, which LN solves by processing off-chain transactions securely and privately. Similarly, LN reduces nodes' energy usage by taking transactions off-chain, supporting sustainability by reducing the energy needed for transactions compared to Bitcoin network operations. Furthermore, LN enables fast micropayments, with transaction outputs over 100 times lower than Bitcoin. Efficient transaction processing is crucial to the viability of LN, as a lack thereof may cause blockchain to lose market share. Lastly, transactions on LN are faster and more efficient thanks to the payment channel, which is a two-party consensus mechanism. This property makes LN a crucial component of the Bitcoin ecosystem.

The main LN properties are instant payment, low cost, and cross-blockchains. Blockchain smart contracts enable secure, lightning-fast payments without requiring on-blockchain transactions or being concerned about the confirmation times of blocks. This property allows instantaneous payment, with its speed expressed in seconds or milliseconds. Due to LN transacts and settles off-blockchain, it enables low fees. LN could handle atomic swaps to allow instant off-chain transactions between blockchains with various consensus rules. So, they should use the same cryptographic hash function for trustless custody to avoid third-party custodians. Due to these properties and advantages, LN is now among the most important representatives of Payment Channel Networks (PCN) compared with networks such as Liquidity, Stacks, RootStock (RSK), and ChainX, to name a few. Although its growth halted unexpectedly on November 28, 2022, due to Bitcoin's depreciation by 57.794% and a decline in the number of nodes and channels, it is still growing steadily.

Although LN adds a crucial component in improving the capabilities of the Bitcoin network, it has its drawbacks. LN provides efficient transactions once payment channels are established, but the setup process is complex. Users must transfer funds to LN and lock them in a channel, which incurs high costs. Funds locked in a channel are still at risk during transactions, as they may get stuck due to technical issues or be taken by a counterparty if the user goes offline. Watchtowers and LN service providers mitigate these offline risks, but they introduce a vector of centralization into the network. In that regard, LN could mirror the hub-and-spoke model of traditional financial systems, where banks and institutions act as intermediaries. This issue concerns fraud, fees, hacks, and price volatility among businesses investing in LN nodes. A more functional limitation of LN is that payment channels are not seamless, as they are only between two parties.

2.2.1 LN Nodes

Although LN is a relatively new solution, its growth is constant [32] due to users that embrace this scaling technology seamlessly. LN offers its users cheaper and faster transactionality, compared to bitcoin fees that increase as the price increases. As of today, there are 16,370 nodes [33] connected to LN with an average node capacity of ₿0.2839, worth some $7,374.18. Under the structure of layer-2 networks, LN runs on top of Bitcoin and is composed of a set of nodes and payment channels that ensure that LN remains decentralized and secure. LN nodes represent users that connect through payment channels by running a node implementation to carry out instant payments. For that, LN uses source routing and offers gossiping and route discovery mechanisms to help nodes locate routes with minimal fees.

When new nodes join the P2P network, they connect to existing nodes after the initial bootstrapping mechanism[1]. Once a node obtains the addresses of some nodes in the LN, it can open a channel that establishes a direct network connection. Through the open channel, the node can send, receive, or forward payments to other directly or indirectly connected nodes, generate payment invoices, to name a few, and then close them when necessary. Nodes that are not directly connected to the sender can use several hops to send payments to a recipient node. Fees are collected by nodes aiding in forwarding payments. The confidentiality given to Lightning nodes is through the encryption of the communication between nodes. Likewise, to increase security, nodes must authenticate to avoid malicious intrusion. A public key serves as an identifier that allows the protocol to establish an encrypted and authenticated connection between peers to route payments through the network securely. The nodes use onion routing, a cryptographic technique that protects the identity of the payer and payee, to guarantee the privacy and security of the

[1]Mechanism to let nodes without contacts discover them as an initial process

transactions. Table 2.1 provides a brief description of the node with its features and fields:

The nodes connect between them by any implementation: LND (Lightning Network Daemon) [34], C-Lightning [35], and Eclair [36], each with different programming languages and parameter values. The specification of each implementation dictates which values are public and which are accessible through gossip messages. For network security reasons, others are kept private. Because of these public values, it is possible to discover which implementation a user uses, since users maintain the default configuration of nodes and channels [37]. LND is the most popular client on the LN, with approximately 87% of the nodes classified as being predominant in most countries. Instead, C-Lightning and Eclair have lower usage than LND, with approximately 11% and 2% of the nodes, respectively. Above all, regardless of the LN implementation, the nodes try to join large hubs.

Developed by Lightning Labs, LND is a feature-rich, robust back-end software implementation of the LN protocol written in Go. It allows users to create, manage, and route payments through LN channels. LND enables users to set up their nodes as transaction intermediaries, providing a secure, scalable infrastructure for participating in this network ecosystem with privacy and transaction integrity. Eclair, a Scala-based LN implementation, provides a feature-rich HTTP API for easy integration by application developers. Defaultly configured for mainnet, Eclair can also run on testnet or regtest/signet. Core Lightning, previously C-lightning, is a C++-based implementation of the LN protocol that is lightweight, customizable, and standard-compliant. It requires a fully connected bitcoind to relay transactions on the network. As recommended in [38], users seeking paths with low maximum latencies should use C-Lightning. Those users seeking shorter paths with high success rates should use LND. Finally, those users seeking low-cost paths regarding fees paid to intermediary nodes that forward payments might use Eclair.

As pointed out in [39, 37], at least 81 countries have at least one LN node, of which the United Kingdom, United States, Germany, Canada, and France are the most relevant countries with the most nodes. However, France has the highest channel capacity among its nodes, even though the United States has the most open channels and has the highest total channel capacity. Moreover, most of the node population is in North America at 44.8%, Europe at 43.1%, and Asia at 6.2%. Instead, with a minimal node population, Oceania has 2.2%, South America has 0.8%, and Africa has 0.6%, with the rest being 2.3% of undetermined location. The global use of LN nodes offers a glimpse into the acceptance of LN as a viable payment method through its micropayment capabilities despite having restrictions on payment amounts given by channel capacity.

Parameter	Definition	Range / Allowed values	Type
`last_update`	last time the node was available on the network.	prol. Gregorian ordinal	uint32
`pub_key`	public key of the current node which among other is used for: create a shared secret, construct a route and a packet, handshake exchange. This value is announced through the *node_announcement* message as the node's identifier.	valid hash value of 33-byte compressed secp256k1	string
`alias`	field that describes the current node with an descriptive name, if applicable^A.	either empty or a value^B	32-byte:UTF-8 string
`color`	the color of a node^A.	range of colors	3-byte:string hex format
`addresses`	field that enables a node to indicate to other nodes that it is open to receiving connections by providing a list of address descriptors.	address descriptors (address:port): ipv4^C, ipv6^C and Tor v2 and v3 onion service	addrlen*byte:array node of addresses
`network`	standard used to establish and maintain a network communication.	tcp	string
`addr`	identifier of a node on a network.	ip address:port	string

`FEATURES`	`BOLT #9`			
Name	**Description**	**Context**	**Dependencies**	**Bits**
`option_data_loss_protect`	used when extra channel_reestablish fields is either required or supported. This field allows to a node to detect that it is fallen behind to avoid a total loss of funds. As a result, it might at least recover non-HTLC funds by forcing a node to remove the ongoing commitment transaction from the chain.	IN	–	0/1
`initial_routing_sync`	a complete routing information dump needed by a sending node, i.e. node needs a full copy of the routing state of the peers. When negotiated via init, this field can be overridden by the *gossip_queries* feature.	I	–	3
`option_upfront_shutdown_script`	when opening channel commits a shutdown scriptpubkey. This field is useful when the node was compromised somehow, thus, it wants to pre-commit to *shutdown_scriptpubkey*.	IN	–	4/5
`gossip_queries`	a sophisticated gossip control. When negotiated via init, it allows an extended number of inquiries for gossip synchronization, so node can indicate that it supports these types of queries with the *gosip_queries_ex* feature bit.	IN	–	6/7
`var_onion_optin`	used when variable-length routing onion payloads is required or supported.	IN	–	0/1
`gossip_queries_ex`	additional information included in gossip queries.	IN	gossip_queries	10/11
`option_static_remotekey`	static key for remote output. When peers negotiated it, it can be applied to all commitment transactions, as well, the node has to set to a right point the parameter *my_current_per_commitment_point*.	IN	–	12/13
`payment_secret`	payment_secret field supported by node, i.e. set to the payment secret specified by the recipient, this prevents probing attacks from nodes along the path.	IN9	var_onion_option	14/15
`basic_mpp`	basic multi-part payments received by a node. This field causes a delay to allow other partial payments to combine, however, it must be reasonably bounded to avoid a denial-of-service.	IN9	payment_secret	16/17
`option_support_large_channel`	to create large channels. This allows other nodes to know which nodes will take *funding_satoshis* $\geqslant 2^{24}$.	INC+	–	18/19

* I : presented in the init message.
* N : presented in the node_announcement messages
* C : presented in the channel_announcement message.
* C- : presented in the channel_announcement message, but always odd (optional).
* C+ : presented in the channel_announcement message, but always even (required).
* 9 : presented in BOLT 11 invoices.
* A : field that allows to provide intelligence services and to customize node's appearance in maps and graphs.
* B : possible entry point for injection attacks during persisting and rendering. It needs to be sanitized before being used in HTML/Javascript context.
* C : node must assure that the address is a routable one.

Tab. 2.1: Node description

2.2.2 LN P2P network

In the LN exists a ***P2P gossip network*** [40] used to send information. This network differs from the Bitcoin P2P network and the network that creates the channels. The gossip network is a superset of the channel network, where LN nodes share their existence and channels, including details on how to contact them and their forwarding fees. Furthermore, LN uses this gossip network to discover other nodes and their channels, providing information about a peer's alias, features they support, and how to reach them. The gossip network provides information on channels, blockchain verification, and peer fees. This information enables nodes to create a network graph for calculating payment routes using this information. The gossip network, where peers frequently update their fees, can appear noisy and resource-intensive. Some statistics can be calculated by analyzing the graph's data, such as the total number of public nodes, their channels, and capacity. Lightning nodes prevent spam attacks by only broadcasting gossip messages from nodes with at least one public channel. It implies that a node must own Bitcoin and cover on-chain transaction fees.

As noted in [41], LN nodes route payments using a local channel graph to find a path to the destination. They synchronize their graph views by sending update messages through this gossip network with a staggered broadcast mechanism, potentially taking over 10 minutes for messages to reach all nodes. A mechanism in this network is the *Node Discovery*, which nodes use to connect with others in the P2P network by broadcasting their ID, host, and port. Initially, one message allows node discovery, in which peers exchange $node\_announcement$, a gossip message, to offer further information about the nodes besides its public key. However, node data updates may result in several $node\_announcement$ messages. A node should open at least one channel when it first connects to the network to be known; otherwise, it will be ignored to prevent trivial denial of service attacks.

LN brings together the nodes participating in the network using a DNS Seed as a discovery mechanism. The nodes gathering depends on their implemented specification and the processing of type[2], AAAA[3], or SRV[4] broadcast by users. The query types indicate conditions to receive a desired result. Thus, those conditions are key-value pairs separated by *dot-separated* (.) subdomain components. The key

[2]**RFC 1035 - Domain Names [42]:** It is a naming resources mechanism to use the names across hosts, networks, and administrative organizations to name a few.

[3]**RFC 3596 - DNS Extensions to Support IP Version 6 [43]:** It is a tracking protocol for a set of Internet standards that intends to outline the adjustments required for the Domain Name System (DNS) to enable hosts to run IPv6.

[4]**RFC 2782 - A DNS RR for specifying the location of services (DNS SRV) [44]:** It is a monitoring protocol for a set of Internet standards that specify a DNS Resource Record (RR) that identifies the location of the server(s) for a certain protocol and domain.

is a single letter that belongs to a specific action, while the value is the required condition expressed on the query. Table 2.2 explains the different key-value pairs:

Key	Value	Default	Definition
r	realm byte	0	To specify the domain to support by the returned nodes
a	address types	6 (both IPv4 and IPv6)	To be used essentially on SRV queries to specify what address type are returned
l	node_id	null	To query for a specific node instead of a random nodes
n	number of desired reply records	25	

Tab. 2.2: Key-Value Pairs of Query Semantics in Node Discovery

LN provides a bootstrap mechanism that lets nodes without contacts discover them as an initial process. LN also provides an assisted node location mechanism that allows gathering the current network address of known peers through supporting nodes. This implementation allows a node to obtain information from subdomains since the DNS is a seed root domain. It is advised to review [45, 46] to get a more insightful idea about this mechanism.

On the other hand, the ***P2P channel network*** in LN serves the establishment of a payment channel between two Bitcoin users in conjunction with a transaction, facilitating off-chain cryptocurrency exchanges independently from the Bitcoin blockchain [47]. The network addresses scalability concerns and enhances the feasibility of frequent micropayments [48]. This network structure allows the creation of payment channels where any two users are able to carry out an endless number of transactions swiftly and without fees. Transactionality is possible, even if they are not directly connected, using multi-hop transactions to route payments through intermediary nodes[47]. In that context, the onion routing protocol in the LN is a technique used to enhance privacy and scalability by routing transactions through multiple nodes.

The route that a packet travels from its origin node is through knowledge of the public key of both the intermediate nodes and the destination node. Knowing those public keys allows the origin node to create a shared secret using Elliptic-curve Diffie-Hellman (ECDH). In this case, ECDH generates a pseudo-random stream of bytes to obfuscate the packet. It also generates a series of keys to encrypt the payload in addition to computing Hash-based message authentication codes (HMACs). The use of HMAC is to ensure packet integrity at each hop by using the SHA256 hashing algorithm.

A premise of this protocol is to protect the sender's identity, for which the hops only see an ephemeral key delivered by the origin node. Each hop must blind the ephemeral key before forwarding it to the next node to achieve an unbindable route. In this way, the origin node appears anonymous, although the destination node becomes public. A more than notable policy of this protocol is that it keeps the version of the packet format and the routing mechanism. In case of receiving a higher version packet, the node must report a route failure to the origin node and discard the packet.

Even though the onion routing protocol provides privacy and partial anonymity, LN could benefit from a network layer-level implementation of High-speed Onion Routing called HORNET [49]. It could help to reduce latency and provide end-to-end anonymity.

2.2.3 LN Channels

LN uses payment channels as its basic building block [50]. In order to facilitate initial funding, an off-chain transaction mechanism known as a payment channel locks funds in a 2-of-2 multisignature address. Through a network of payment channels, these channels let users send Bitcoin transfers with low latency. The channels significantly improve transaction speed and reduce blockchain congestion [51, 52]. While the channel capacity (total bidirectional balances) is public information, the specific balance distribution within the channel is kept confidential for privacy reasons. Some of the benefits that LN channels provide to the network:

- ***Scalability:*** LN improves scalability by drastically reducing the strain on the Bitcoin network by executing several transactions off-chain.

- ***Low fees:*** Users can save transaction costs as they are not subject to standard Bitcoin network fees.

- ***Speed:*** Transactions within a channel are practically instantaneous since they do not need to wait for on-chain transaction confirmations.

Bitcoin growth peaked on April 12, 2021, with a market price of $61,193.55 [53]. Bitcoin's reciprocal counterpart, the LN network, has experienced more activity than ever. The total number of payment channels on LN is around 68,275 with an average channel capacity of ₿0.0683, equivalent to $1,802.94 [33]. Channels operate through three stages: opening, operating, and closing. In a channel, two parties lock coins, exchange transactions, and broadcast the latest state to the blockchain. After a node connects to the P2P network, it can open an LN channel, which engages a bidirectional connection with other nodes to exchange funds.

Channels can be created and closed by node agreement if not indicated otherwise. Before two nodes can fully interact in a channel, they must establish the channel with one transaction and close it with another. For that, both transactions require locking funds on the Bitcoin blockchain. A set of rules governs the establishment of a new payment channel. These rules state that only with the permission of both nodes can spend locked funds. After payment, it is permissible for both nodes to modify the balance within the channel multiple times, as long as the channel stays open under pre-set rules. However, both nodes broadcast their most recent balance to the Bitcoin blockchain when they mutually decide to close the channel.

Once an LN node establishes one or multiple payment channels, the node's user can perform payments through those channels. However, to be able to perform payments to nodes that lack a direct channel with a given node, it needs to find existing channels on the network. LN specification offers a channel-discovering mechanism that allows a node to create a structure with the network's topology. The node stores this structure and updates it locally in a JSON file. When a node has one or some open channels and is aware of the network topology, the node can perform a payment or interact as an intermediate node in a route of multi-path payment.

On the other hand, a valuable feature of micropayment channels is the delayed broadcast of the state of a transaction at a later time. Contracts are constructed to accomplish such property, with one party accountable for broadcasting transactions before or after a particular date. Similarly, to validate data and order events, the network can use clocks for decentralized consensus [54] and states [55]. Contracts can use Bitcoin transaction scripts that create timeframes in which certain broadcast states can later be invalidated. The transaction malleability soft-fork in bidirectional payment channels is essential in the case of LN to achieve almost infinite scalability and mitigate intermediate node default risks. Table 2.3 provides a brief description of the channel with its features and fields:

2.2.4 LN Channel Lifecycle

The lifecycle of a payment channel consists of four phases: discovering channels, opening a channel, making a payment in the network, and closing the channel. Discovering a channel enables nodes to maintain a local network view for finding routes to desired destinations. Opening a payment channel involves a couple of nodes agreeing to open a channel with some funding. The channels create a multi-signature account to set that amount as channel capacity. The payment channel will be open upon the addition of the channel funding transaction to the blockchain. Payers can make payments to payees through direct channels or by routing on the network. In Section 2.2.5, we detail the payment phase in the LN. When users want to close a channel, the final phase is recorded on the blockchain. Figure 2.1 depicts the lifecycle of a payment channel.

Parameter	Definition	Range/Allowed values	Type
`channel_id`	identification of the channel generated from the funding transaction through the use of an exclusive-OR to combine the parameters: *funding_output_index* (2-byte) and *funding_txtid*. The field can be used when operating in parallel with multiple channels. The *funding_signed* message introduces the *channel_id* field as the channel identifier and can be used with different messages such as: *funding_locked*, *shutdown*, *closing_signed*, *update_add_htlc*, *commitment_signed*, etc.	32-byte	string
`chan_point`	the funding transaction's id and the channel funding transaction's output.	32 bytes (funding_txid) + 2 bytes (output_index)	string
`last_update`	last time the channel was active on the network.	proleptic Gregorian ordinal	uint32
`node1_pub`	nodes appear on a lexicographical sequence so a message passes signature verification of channel_announcement and for channel_update messages.	valid hash value	string
`node2_pub`	nodes appear on a lexicographical sequence so a message passes signature verification of channel_announcement and for channel_update messages.	valid hash value	string
`capacity`	total capacity defined for the channel between two nodes. When refers to *htlc_maximum_msat* is a static value over the life of the channel, but it does not indicate the real-time channel capacity in each direction. This static value makes it possible to prevent a significant data leak and network spam.	denominated in satoshis	int64

Tab. 2.3: Channel description

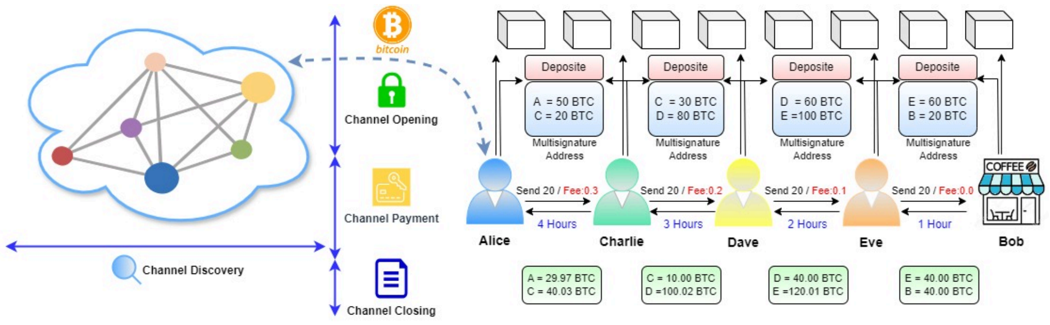


Fig. 2.1: Channel Lifecycle diagram

***Channel Discovery* -** A node can confidently establish a comprehensive and localized understanding of the network's topology through a channel discovery mechanism. It can also enable seamless and efficient identification of optimal routes to desired destinations. Two gossip messages are necessary to support the channel discovery:

- *channel_announcement* message contains information regarding new channels between two nodes.
- *channel_update* message, which updates information about the channel.

However, after channel establishment, only one valid $channel\_announcement$ message is required for any channel, unlike a channel update that expects two messages. The following parameters and messages, along with those mentioned above, complement the components of this mechanism:

- $short\_channel\_id$: provides to the funding transaction a one-of-a-kind description.

- $channel\_update$ **Message:** is appropriate for relaying payments rather than sending payments. When a node builds a route, it includes the estimated amounts and expiration of the Hash Time-Locked Contracts (HTLCs)[5] from the destination to the origin; that is, it calculates these values backward. The payment request consists of the exact values of $amount\_msat$ (initial value) and $cltv\_expiry$ (minimum value) to be used for the last HTLC of the route.

- $channel\_announcement$ **Message:** reveals information about the channel owner that links the Lightning node key associated with the Bitcoin key on the blockchain. Even if the node sends a channel announcement message, it will remain inoperative until at least one of the parties, via a $channel\_update$ message, publishes its expiration and fee levels.

- $announcement\_signatures$ **Message:** acts as an opt-in method to notify the rest of the network about the channel announcement between the two endpoints of a channel. The first step to creating such a message is to construct the $channel\_announcement$ message that belongs to a newly created channel. After this procedure, the message could be sent along with the announcement signatures.

Channel Opening **-** The channel establishment process starts with an $open\_channel$ message from the sender and an $accept\_channel$ message from the receiver, following a successful handshake agreement. The sender then creates a funding transaction and a commitment transaction with two versions, sending a $funding\_created$ message with the outcome of the funding output and the receiver's signature. The receiver generates a signature for the commitment transaction version, sending it as a $funding\_signed$ message to the sender.

Upon receiving the $funding\_signed$ message, the channel sender broadcasts the funding transaction to the Bitcoin network. Both parties wait for confirmation in the blockchain before sending the $channel\_ready$ message to establish the channel. The exchange messages are shown in Figure 2.2.

[5]It refers to a payment routed over many channels.

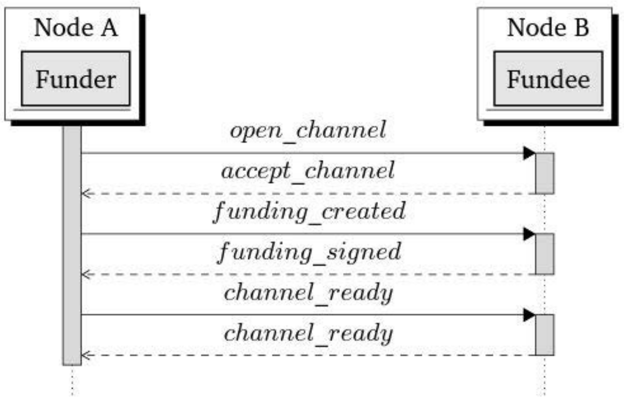


Fig. 2.2: Channel opening diagram

Channel Closing **-** Instead of a unilateral closure, it is advisable that, through negotiation, the connection be mutually closed to have instant access to the funds as well as negotiate lower fees. The process by which peers adopt it is as follows: i) A node announces that it intends to empty the channel and will not admit additional HTLCs. ii) Final channel closure negotiation starts once all HTLCs are resolved. When two nodes try to close a payment channel, to close said channel, the nodes exchange some messages, as shown in Figure 2.3, the same ones detailed below:

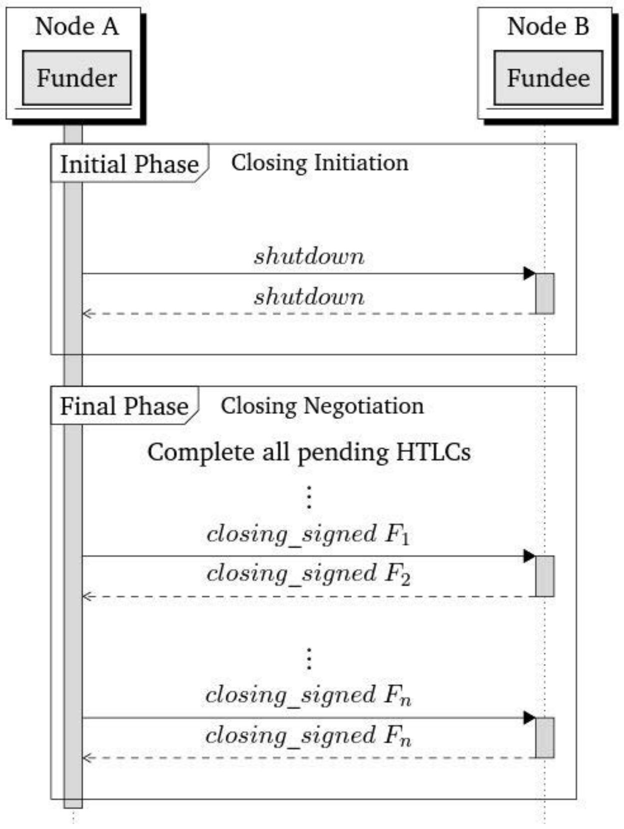


Fig. 2.3: Channel closing diagram

- *shutdown* **message:** indicates the initiating closing process, given by either node or both. At the time a sender did not send a $funding\_created$ message and a receiver did not send a $funding\_signed$ message, both peers cannot send a shutdown. However, they can send a shutdown before a $channel\_ready$ message, which must occur before a funding transaction reaches a $minimum\_depth$. After a shutdown, the sending node must fail to route any HTLC added.

  On the other hand, if a sender did not send a $funding\_signed$ message and a receiver did not send a $funding\_create$ message, both peers cannot send a shutdown. Instead, they should fail the connection. All these actions follow the premise that in case of terminating a channel connection, there must not be new HTLCs added or accepted.

- *closing_signed* **message:** occurs once the shutdown concludes and the channel is free of HTLC. The closing fee negotiations begin as the last commitment transactions run out of HTLC. Then, a back-and-forth negotiation continues until both nodes settle on the fees or some node rejects the channel. Furthermore, the fee negotiation is repeated on reconnecting to prevent a saving state and to deal with fees that shift between disconnection and reconnection.

2.2.5 LN Payments

Alice establishes a payment channel with Bob. Alice completes the process by locking 50 BTC to the channel. Three transactions are involved in the setup of the payment channel: Alice and Bob, each with a commitment transaction and a funding transaction sent to the blockchain network. Each transaction holds the owner's Bitcoins, its hash, a secret generated randomly, and the channel counterparty's secret hash. One can think of the commit transaction as the channel's current balance. Bob sends Alice an invoice for 20 BTC, which Alice pays, resulting in two new commitment transactions. Each user creates a hash and a new secret and sends the secret and the hash of the previous transaction to the counterparty. The new balances of Bob and Alice are held by two new commitment transactions. Alice's new state is as follows: after 1,000 blocks, Bob has 20 BTC, and Alice has 30 BTC. After 1,000 blocks, Bob's new state is: Alice gets 30 BTC, and Bob gets 20 BTC. Figure 2.4 shows a direct payment between Alice and Bob in LN. Furthermore, Table 2.4 provides a brief description of the parameters in the channel policies with their features and fields used to make direct or multi-hop payments:

Through a sequence of off-chain transactions, LN enables fast and affordable transactions. The following information explains the steps required to make payments (Payment Process), the transactions that take place (Transactions Created and

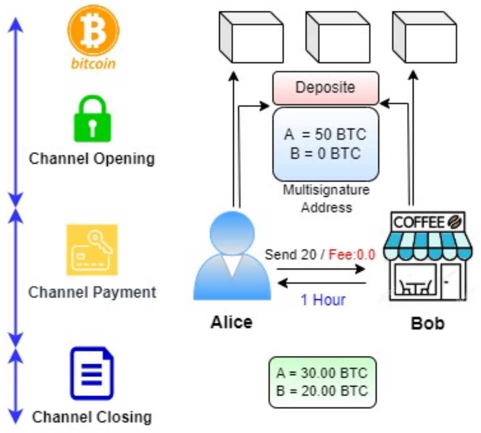


Fig. 2.4: Direct channel payment

Parameter	Definition	Range	Default values (lnd/clight /eclair)	Affects	Type
`time_lock_delta`	Value required for the channel when HTLCs are forwarded. It may be implicitly enforced by *htlc_signature* if received HTLCs are spent or offered HTLCs expire.	Integer	144	In	uint32
`min_htlc`	A conditional payment that indicates a minimum HTLC in milli satoshis accepted by an initiator in a transaction. It refers *htlc_minimum_msat*.	–	default 0	In	int64
`fee_base_msat`	The channel base fee is the amount it will charge for any HTLC.	mil-lisatoshis	–	Out	int64
`fee_rate_milli_msat`	The effective fee rate. Regarding *fee_rate_per_kw*, it refers to the initial fee rate in satoshi to be paid for HTLC transactions and commitment to be included immediately in a block.	mil-lisatoshis	–	Out	string
`disabled`	It indicates whether a policy is disabled or not for the peers.	True False	False	In/Out	boolean
`max_htlc_msat`	In addition to setting the *option_channel_htlc_max* of *message_flags* to 1 to signal the presence of the field, this specifies the maximum HTLC it will send across the channel for a single HTLC. It refers to *htlc_maximum_msat*, a static value over the channel's life, but does not indicate the real-time channel capacity in each direction.	limited to $2^{32}-1$ mil-lisatoshis	–	Out	int64
`last_update`	Last time a policy was updated.	–	proleptic Gregorian ordinal	In/Out	uint32

* In : When a node receives a message in which involves the referred field
* Out : When a node sends a message in which involves the referred field

Tab. 2.4: Parameters in the channel's policies

Exchanged), how trustlessness is ensured (Trustless Mechanisms), and why network monitoring is necessary for nodes (Monitoring the Network).

Payment Process - The payment process includes four stages that range from opening a channel to completing the payment with its transactions and mechanisms. If a payer wants to send a given number of Bitcoins to any network user, it must first find a direct path to that user with at least that amount of Bitcoins on every direct channel. In a direct payment, even when balances are kept confidential, the success percentage of any payment between neighbors is not compromised [56]. On the other hand, when a node cannot find a direct channel, the payment has to be routed on the network in a multi-hop path that we will cover in more detail in Section 2.2.6.

1. *Channel opening* - As indicated in Section 2.2.4, parties must first create a *funding transaction*, the first on-chain transaction, to finance the multi-signature address. The payment channel is deemed open following a few blockchain confirmations of the funding transaction. Additionally, two commitment (off-chain) transactions hold the amount of Bitcoins each party in the channel owns.

2. *Creating a payment* - After a channel opening, a payer can send an invoice to a payee by exchanging *commitment transactions*, where every transaction modifies the channel's balance and, thus, its current status. When a new commitment transaction is created to send one Bitcoin from the payer to the payee, the payment reduces the payer's balance by one Bitcoin while increasing the payee's balance by the same number of Bitcoins, that is, this is a *balance update*.

3. *Hash Time-Locked Contracts (HTLCs)* - An HTLC refers to a payment routed over many channels [57]. The HTLC is a *conditional payment* that makes trustless payments possible. The payee of an HTLC must provide cryptographic proof (preimage) within a certain amount of time to claim the funds. Moreover, HTLC guarantees the release of locked values after a predetermined amount of time or by giving a secret that generates a predetermined hash. HTLC ensures atomic cross-channel transfers by locking coins from sender to receiver with contract conditions. Two key conditions used are:

   i) *Hash Locks (HL):* limits output spending until certain data is disclosed, enabling the combined spending of many outputs with the same HL.

   ii) *Time Locks (TL):* limits Bitcoin spending to a specific future date or block height to ensure that contracts are carried out within the network.

   Payments may be routed across several channels and nodes thanks to *route finding*. Each node adds an HTLC to its channel with the subsequent node, guaranteeing that only payment claims are possible when the preimage is released. Routing algorithms have a significant influence on LN efficiency since they determine whether to favor short pathways or channel capacity based on the approach chosen.

4. _Payment completion_ - In the $preimagerevelation$, the preimage is shown by the payee to collect the funds from the prior node. This preimage is then broadcast backward along the path, allowing each intermediate node to claim its funds. Commitment transaction validation involves revealing and re-hashing the original preimage to confirm data integrity. This mechanism is the foundation of data integrity and safe transactions in blockchain systems like Bitcoin and LN. If the payee fails to reveal the preimage within the allotted period, the transaction is invalidated, and the funds are refunded to the sender. This mechanism prevents funds from being locked indefinitely if a recipient is unable or decides not to satisfy the transaction's conditions.

Transactions Created and Exchanged - In LN, every transaction is off-chain and updated locally via payment channels connecting nodes. The most current balance is sent to the blockchain when a channel node chooses to close it. Thus, several transactions follow the payment process, however, the most relevant are:

1. _Funding Transaction_ - To exchange Bitcoins via LN, two parties create a payment channel by locking funds on the blockchain. Funds are deposited into a 2-of-2 multi-signature address by both parties, requiring both signatures to spend the funds. They can use the channel to exchange Bitcoins after the double-signed funding transaction is received and confirmed by the blockchain. On the other hand, either side can send a settlement transaction to the network to close the channel, storing the sum of off-chain transactions on the blockchain. The funding transaction amount denotes the maximum amount of funds that a pair of nodes can transact through the channel. This transaction serves as a representation of its maximum capacity.

2. _Commitment Transactions_ - Upon opening the channel, both parties can begin signing transactions between them as often as desired—these transactions, referred to as commitment transactions, occur off-chain. Then, each party exchanges commitment transactions to send a payment. Despite not being instantly broadcast to the blockchain, these transactions represent the most recent channel balance following each payment. Instead, they are both signed and held. Transactions are cheap and instant because they do not need to be mined or spread throughout the Bitcoin layer-1 blockchain network. Both parties keep a local copy of the ledger for their balance, updating it after each transaction. The channel's state is updated with each transaction, preventing fraud by not allowing parties to refer back to old states when settling on the blockchain.

3. _Conditional Transactions_ - These transactions guarantee that the payment may only be routed and redeemed if the preimage is disclosed. For that, HTLC is

used to stop cheating in the system. Transactions can be routed via several nodes thanks to HTLCs. HTLCs allow two parties to deal through intermediate channels rather than a direct route. HTLC is a normal Bitcoin transaction that contains a smart contract, which is a unique script. It is a conditional payment with a temporary lock on the transaction. HTLCs generate conditional payments in Bitcoin, making them a potential mechanism.

4. *Closing Transactions* - Both parties may sign a closing transaction known as a settlement transaction once they have agreed to settle the funds. The final transaction will be recorded into the blockchain and mined. The closing transaction will reflect the total amount of the two users' final settlement balance. For LN to leave the channel, none of them have to cooperate. The relationship can be terminated by either the payer or the payee choosing to close the channel. By keeping one of them from going offline and locking the other's funds within the channel stops fraud.

Trustless Mechanisms - Several mechanisms attempt to provide trustless payments without diminishing LN security. In that sense, a mechanism is the commitment transactions where parties own signed but unbroadcasted transactions. These transactions, if necessary, can be broadcast to terminate the channel and settle the blockchain balances. Another one is a penalty mechanism that serves to deter cheating. By offering a secret key, the opposing party can obtain all the funds in the channel if one tries to broadcast an out-of-date commitment transaction. Moreover, a safe mechanism of routing payments via several nodes without needing to trust intermediaries is provided by HTLCs, which ensures that payments may only be claimed if the preimage is disclosed.

A key protocol in LN is the Onion Routing Protocol. The usefulness of this protocol, as specified in [58], lies in routing payments from an origin node to a destination node through private communication to provide privacy in this public network. Hops route a packet through some of the intermediate nodes. Sphinx, known for its proven safe mix [59], is the basis for constructing this routing scheme, which is additionally extended with per-hop payloads. Intermediate nodes know which node to forward the packet to by removing a layer of encryption and verifying its integrity before forwarding the message.

As a privacy constraint, the intermediate node only knows its predecessor and successor. That means such a node does not know anything about the other nodes that conform to the route, as well as it does not know its length or its position on it. To increase channel security, the obfuscation of the packet takes place on each hop to deter any network-level attack by associating packets that belong to the same route. However, there is the possibility of carrying out an attack using traffic analysis to associate packets.

Monitoring the Network - The nodes monitor the network to ensure each node behaves appropriately without incurring attacks, denial of service, or misbehavior. Nodes must furthermore keep an eye on the network to spot any attempts by a counterparty to disseminate an out-of-date commitment transaction. Watchtowers are outside services that can keep an eye on a user's network and react to efforts at fraud. A timely response is necessary for the penalty mechanism. The non-cheating party must promptly supply the secret key to retrieve the funds and punish the cheater if an outdated transaction is broadcast. Similarly, to guarantee that nodes can react appropriately to any fraudulent behavior, nodes maintain track of the most recent status of the channel, including all commitment transactions and HTLCs.

2.2.6 LN multi-hop payment

As mentioned previously, LN allows transactionality between a pair of users even when they do not share a direct connection. A mechanism that creates multi-hop routes to transmit payments through each LN user is necessary for a scenario without a direct payment channel. In a payment route, the transaction traverses all the users, defined in a path, with sufficient funds, which charge a nominal fee to compensate for their work to relay the transaction to subsequent users. The source node constructs the payment route by using route discovery, which depends on an updated topology of the LN network. Most LN implementations follow this procedure, providing a computed route with the hops and fees the source node will use for the payment. However, the source node can determine the payment route with the information available in the LN network. It is worth mentioning that two stages shape the ***multi-hop payments*** in LN. First stage, the process of creating a set of contracts called HTLC takes place, which locks the funds that satisfy these contracts. Second stage, in the payment process, an atomic exchange occurs, in which either the payment succeeds in all the hops or cancellation of the contracts takes place, i.e., the funds are redeemed and are again available to the channels.

The relevance of HTLC for payments lies in the bond it creates between a payer known as Alice (A) and a payee called Bob (B). To make a payment, B has to provide not only the preimage of a hash value but also the digital signature. Once B provides these two elements, it can reclaim the funds locked by A. However, if B cannot commit a preimage before the expiration date set by A in the contract, A can retrieve the funds once it provides the digital signature. In a more detailed description of the payment routing process, B sends A a hash value $h(x)$ computed from a randomly generated value x, known as the preimage. Now, A can pay to B through one or more nodes, in this case, say through hops (Charlie) C and (Dave) D on a route as follows: $A \longleftrightarrow C \longleftrightarrow D \longleftrightarrow B$. With the hash $h(x)$ received from B, A can generate an HTLC that routes the payment to C. Then, C routes the received HTLC containing the same hash to D and this to B. Once B receives this message, it reveals the preimage to D, which also reveals it to C and A so that both the payer and the hop can redeem the funds.

Consequently, when a payment occurs between a payer and a payee, HTLC guarantees the payment is complete. As the first state of a multi-hop payment, there is an *HTLC Establishment*, where the payer creates the contract HTLC that must reach the payee through intermediate hops. From there, there are a couple of states where payment will be routed based on the actions taken by the payee or intermediate hops. The *HTLC Fulfillment* state occurs when the payee reveals the preimage x to the intermediate hops until it reaches the payer so that everyone can redeem their corresponding funds. The other state is *HTLC Failure*, in which the preimage does not arrive within the timelock set in the contract either to the payer or to the intermediate hops, so there are no changes in the balance of the channels.

Furthermore, it is wise to assume that, as with most payment methods, LN also suffers from various attacks. In the case of LN, an attack vector can arise when there is a cancellation of payments in the second stage of a multi-hop payment. Subsequently, the attacker can lock the funds of one or more users by proceeding with the first stage of the payment and then canceling it. Another way to attack users is to withhold payments for a lengthy time during the first stage. This attack aims to increase the damage to the nodes economically. As a countermeasure, and as noted in [60], when LN users hold the values of some main parameters at their default settings, the cost of the attack is significantly low. However, setting values different from their default values while mitigating the attack reduces the performance of the multi-hop payment network. Similarly, if there is an adjustment of the HTLC negotiation parameters, there will be an increase in the cost of the attack.

With each successful transaction, there is a shift in the channel balance after the processing of the mutually signed commitment transaction, which is known as a ***payment execution***. In this step, each hop charges the payment amount and the fees on the route, which are part of the commitment transaction. But a channel might have either a zero balance or the flow of the payments follows a single direction. Consequently, the channel can fall into a step of ***unbalancing***. Then, once the user deems that there is no need for the created channel, the user might choose to close it, in a step called ***channel closing***, by sending a settlement transaction to the blockchain.

For simplicity, Figure 2.5 shows an example of a multi-hop payment between Alice and a coffee shop run by Bob, with Charlie, Dave, and Eve as hops. As seen, these hops charge a routing fee to forward a payment to its destination. Additionally, for ease of depiction, the $cltv\_expiry$ displayed in hours is expressed as block height with a decreasing value between hops.

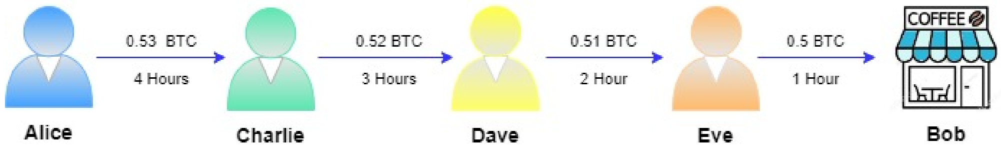
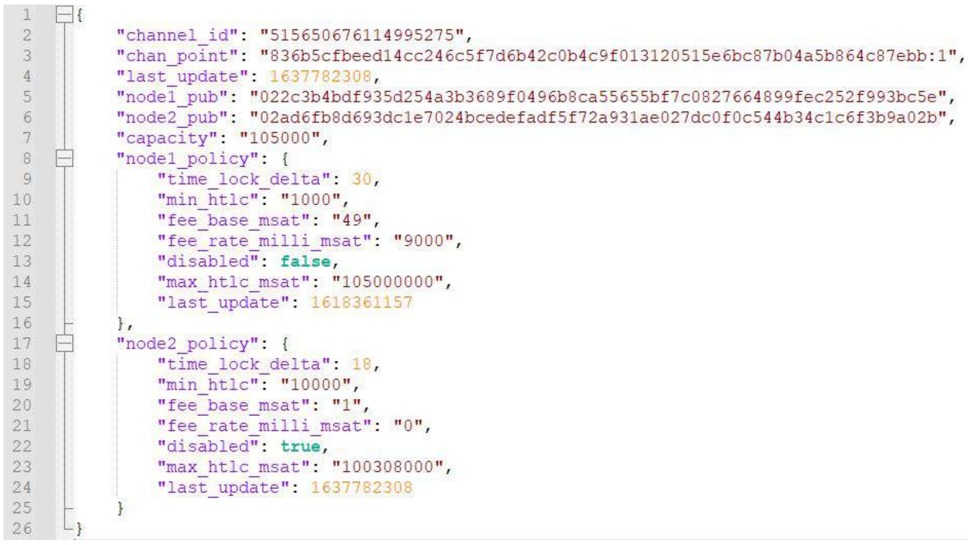


Fig. 2.5: Multi-hop payment

2.2.7 LN Channel parameters and policies

Once both nodes establish a channel ready to transact, they exchange data about the channel open and the fee policies [50]. The data of the channel includes several parameters that describe it. Among those parameters are $channel\_id$, $last\_update$, $node1\_pub$ and $node2\_pub$ (channel nodes), $capacity$, and $node1\_policy$ and $node2\_policy$ (node policies). In that regard, a node policy in the LN is a set of parameters and rules that govern the operation and behavior of a payment channel between two nodes, established and enforced by its participants. Figure 2.6 shows a payment channel between two nodes that was captured and exported from an LND client implementation. It presents the channel parameters as well as the node policies with their setting parameters. The following parameters describe the channel



```
{
    "channel_id": "515650676114995275",
    "chan_point": "836b5cfbeed14cc246c5f7d6b42c0b4c9f013120515e6bc87b04a5b864c87ebb:1",
    "last_update": 1637782308,
    "node1_pub": "022c3b4bdf935d254a3b3689f0496b8ca55655bf7c0827664899fec252f993bc5e",
    "node2_pub": "02ad6fb8d693dc1e7024bcedefadf5f72a931ae027dc0f0c544b34c1c6f3b9a02b",
    "capacity": "105000",
    "node1_policy": {
        "time_lock_delta": 30,
        "min_htlc": "1000",
        "fee_base_msat": "49",
        "fee_rate_milli_msat": "9000",
        "disabled": false,
        "max_htlc_msat": "105000000",
        "last_update": 1618361157
    },
    "node2_policy": {
        "time_lock_delta": 18,
        "min_htlc": "10000",
        "fee_base_msat": "1",
        "fee_rate_milli_msat": "0",
        "disabled": true,
        "max_htlc_msat": "100308000",
        "last_update": 1637782308
    }
}
```

Fig. 2.6: Node Policies in a Payment Channel

and node policies with their parameters:

- *channel_id:* is unique identifier of a channel

- *capacity:* is the total balance of a channel, calculated by adding individual balances of parties. The capacity is set at the channel's opening and cannot be altered without closing the channel.

- *Channel nodes:* are the public keys of the parties, with parameter names $node1\_pub$ and $node2\_pub$, participating in the channel. The node that began opening the channel is $node1\_pub$.

- *Node Policies:* Node policies refer to $node1\_policy$ and $node2\_policy$ and represent the fees that both parties set to forward a payment when they are part of a route as intermediaries.

  - *time_lock_delta:* assigns to a transaction an expiration date, which otherwise may cause it to become permanently pending. The unit of measurement is blocks, where its maximum time lock value is the number of blocks that should be mined in a period of 14 days.

  - *min_htlc:* is the lowest amount of HTLC that a node will tolerate. This option is static, meaning that it is set at channel opening and stays that way until channel closure.

  - *fee_base_msat:* is the fixed amount a node charges to send payment regardless of payment amount and is part of each HTLC.

  - *fee_rate_milli_msat:* describes the amount that is charged as a fraction of the payment's total value. The proposed charge will increase with the amount of the payment.

The node policies used on a payment channel depend on the parameters set by the user that opens it. However, the user can set those parameters according to its needs. According to the three LN implementations [61, 62, 63], the available parameters that can be set are the amount of satoshis to commit to the channel, initial amount of satoshis to push to the remote side, fee rate, private/public channel, minimum amount of millisatoshis for incoming HTLCs on the channel, the base fee, and proportional fee to transfer payments. The main topics that a channel policy usually addresses are:

- ***Routing Liquidity:*** Nodes can configure a limit on the amount of funds they will allocate for payment routing. This aspect covers the upper and lower bounds on the amounts they permit for payments coming in and leaving out via their channel.

- ***Fees:*** The fees charged to a node to route payments through its channel are configurable. The fees may be a percentage of the total payment or a fixed rate.

- ***Expiration time:*** LN payments are time-sensitive. Thus, each node can establish an expiry time for payments that flow over its channel. A payment may expire and need to be retried if it takes too long to process.

- ***Channel Management:*** Nodes may choose to close channels due to a variety of reasons, including modifications in the channel balance, structured fees, or general network health.

- ***Policy Updates:*** Over time, nodes might modify their channel policies. The motive could be network circumstances or their operating requirements. Therefore, the nodes may alter their fee amounts or payment limits.

When a channel is up to send a payment, in the case of a unique hop, the fees set on the channel do not intervene in the payment. Otherwise, the fees are considered along the route except for the last hop, which is the receiver of a payment that does not charge for it. However, a node can establish the parameters a payment route must limit. For instance, on [64, 65, 66], the sender sets parameters such as the timelock for the final hop (CLTV delta), an upper limited amount of time to attempt to fulfill the payment, and the maximum amount of satoshis set as the payment fee. On the other hand, to forward a payment, a node charges by receiving a payment with the fee set on its channel, which sends the payment to the next hop. Policies are essential for routing payments on LN, influencing efficiency, cost, and reliability. Node operators can manage resources and participate effectively.

Layer 2 Protocols Categories and Subcategories

3

Computers are good at following instructions, but not at reading your mind.

— **Donald Knuth**
(Computer scientist and mathematician - "father of analysis of algorithms")

Despite most efforts to provide a wholesome solution after a decade of Bitcoin staging, blockchain cannot fulfill the three critical aspects of a ledger: security, scalability, and decentralization, known as the blockchain trilemma [67]. On the one hand, scalability and security impede decentralization, and increased scalability threatens security. On the other hand, security and decentralization are necessary and fundamental due to the nature of blockchain. However, scalability remains a challenge because of its consensus protocol as a main drawback.

Since one of the main functionalities of blockchain lies in the handling of transactions, scalability refers to the number of transactions handled in a period of time, which in the case of Bitcoin is between 3.3 and 7/tps. Furthermore, the lack of balance between these critical aspects limits the widespread adoption of cryptocurrency technology in the industry, especially for its scalability. For instance, high decentralization characterizes public blockchains, which can restrain security threats but with minimal processing of transactions per second. In contrast, transactional throughput on private blockchains is high, although they are centralized and cannot repel some blockchain-related attacks [68].

Following the implementation of the Bitcoin blockchain and its subsequent success, most scalability solutions focus on improving *Layer-1* or deploying *Layer-2*. Concerning Layer-1, the solutions attempt to change the structural attributes of the blockchain. These solutions aim to counteract inadequate transaction rates and substantial latencies in transaction processing. The scope of action of the proposed solutions points to the operating principles underlying blockchains, such as: modification of block data [69, 70], consensus mechanisms with alternative proposals [71, 72, 73, 74], Directed Acyclic Graphs (DAG)-based solutions [75, 76], or splitting the network into fragments [77, 78, 79]. Other approaches, like [80], offer a performance improvement to the chain rule in Bitcoin through a ghost rule. Instead, [81] discusses consensus mechanisms and proof-of-work in scalable blockchains. Nevertheless, Layer-1 solutions usually lack backward compatibility and are fundamentally flawed because they require modifications to the core design elements of blockchains. Therefore, these solutions make them complicated to deploy in reality [82].

Almost simultaneously, other approaches for Layer-2 protocols address the blockchain scalability issue without altering its consensus mechanism. These approaches enhance the rates of transaction processing and fees by reducing the utilization of the sluggish and expensive blockchains that underpin them. In that sense, the blockchain only fosters trust and resolves conflicts among participants in Layer-2. Consequently, only a certain number of transactions are sent to the main chain, and participants execute unlimited transactions off-chain on an authenticated network. Additionally, to provide security, transactions at Layer-2 follow either of two directions: delayed finality as in commit chains [83] or collateral finality as in payment channels [5, 84, 85].

Although blockchain can have open (permissionless) or restricted (permissioned) access, its expressiveness is derived from the scripting language that blockchain supports, such as in Bitcoin with an incomplete Turing script [13] or in Ethereum with a complete Turing script [86]. While permissioned and permissionless blockchains can be used to create Layer-2 protocols, the relevance of blockchain expressiveness is crucial when developing protocols of Layer-2 above Layer-1.

3.1 Layer-2 Protocols Categories & Subcategories

Layer-2 protocols take it for granted that only legitimate transactions will be recorded in the ledger, no matter the underlying blockchain. To achieve it, several kinds of protocols make up Layer-2, with their characteristics, procedures, requirements, and so on, where each one falls into any of these types: cross-chains, side/child chains, hybrids, and channel solutions.

3.1.1 Cross-chains

Besides the scalability issue, many blockchains suffer from interoperability issues due to their lack of flexibility and application portability. It also has diminished scalability due to the transition of transactions between blockchains [87]. To overcome this issue, cross-chains [88, 89] act as a means to communicate assets between different blockchains through an established procedure of mutual trust and two approaches comprehend this solution:

Notary scheme: For this approach, an entity called a notary [90, 91], actively monitors a set of blockchains for transactions to create a simile on one chain when a comparable event occurs on another chain. Representatives are Coinbase and Binance, which are crypto exchanges.

Blockchain of blockchains: Also known as the Internet of Blockchains [92, 93, 94], it emphasizes both interoperability and customization through building an ecosystem where blockchains share not only data but also tokens. The exchange is made through a platform that communicates the chains but does not act as a central entity. This platform is a core chain that also enables the reuse of network, data, consensus, and a contract layer to create custom and specific applications, resulting in interoperable blockchains.

3.1.2 Side/child chains

The protocols [95, 96] transfer not only computational processing to lessen on-chain load to a parallel distributed ledger but also assets to diverse blockchains. However, this independent ledger uses proof-of- either authority or stake as a consensus mechanism to process the transactions. Similarly, side chains communicate with the on-chain via a bridge that can be used to exchange funds. Its usefulness depends on how quickly the transactions are processed and its capacity to exchange data rapidly with on-chain. However, these protocols do not comply with agile processing due to their centralized mining power and confirmation. It is also diminished by periods of competition in accessing funds in the chains. Nevertheless, two approaches form part of this solution:

Commit chains: Unlike payment channel solutions (Section 3.1.4), which lock funds in open channels without the ability to reuse them beyond their scope, commit chains [83, 97] address this scalability issue through non-custodial operators. The operator starts and maintains a chain of commits, whereas the smart contract hinders the operator from engaging in inappropriate behavior. Although there are no on-chain transactions registered with a commit chain, participants must log in regularly to view checkpoints that are the most recent status of their account balances. Also, even when participants are disconnected, they continue to receive funds, similar to on-chain transactions. However, the level of security in these commit chains depends on their on-chain, given by their consensus mechanism.

Rollups: Similar to commit chains, rollups [98, 99] follow a non-custodial approach. The aim is to reduce the on-chain processing burden through techniques that compress data and smart contracts to scale the on-chains. In batches outside of the on-chain, transactions are processed and then aggregated for verification within the on-chain. The Merkle root also called the state root, remains in the smart contract, which is up-to-date on-chain based on the status of the Rollup. When a batch of transactions is performed, it updates the balances and computes a new state root. Transactions are compressed when someone publishes a batch, so the batch contains this compressed data with the previous and current state roots. Two types of rollups exist, depending on how they prevent fraud and validate the new state root:

- ***Optimist Rollups:*** Its approach is optimistic, assuming that the validity of a transaction is voided when challenged. This approach provides scalability, as there is no computation involved to verify transactions. However, the contract keeps track of updates to the root state and its batch hashes.

- ***zk Rollups:*** This approach, instead of voiding only those challenged transactions, suspects each one of them. As a result, each batch comprises a cryptographic validation proof. Therefore, of the executed batch transactions, their outputs must match the new state root.

3.1.3 Hybrid solutions

These solutions modify some essential properties of Layer-2 solutions to improve their protocol scalability. These solutions aim to either minimize on-chain dependency or eliminate peer-to-peer trust requirements by using a secure resolution mechanism. A couple of approaches are part of this solution:

Bisection protocols: The goal of these protocols [100, 101] is to enhance the mechanism for resolving disputes. The protocols achieve it by minimizing the load on Layer-1 by engaging in off-chain computations.

TEE-based solutions: The Trusted Execution Environment (TEE) has the primary goal of safeguarding data integrity and confidentiality loaded into an area of the CPU such as [102]. For blockchain scalability, these solutions [103, 104, 105] leverage the safeguarding of integrity provided by TEEs to remove the need to use on-chain guarantees when establishing peer-to-peer trust. However, TEEs have their susceptibilities and uneasiness that could be inherited in TEE-based solutions.

3.1.4 Channel solutions

Channels [106, 84, 5], as a key Layer-2 protocol, provide scalability and privacy by setting up private means for transactions between two users. Transactions, though handled off-chain, maintain an identical level of security to that of an on-chain transaction. However, a predetermined and mutually agreed-upon set of rules is established for the purpose of ensuring the security of transactions. Two types of channels form this solution:

State Channels: The principal premise of this solution [106] is that two or more users can exchange or transfer states for use in any arbitrary program, such as auctions or voting, to name a few. For this kind of channel, its establishment is through a smart contract where users join the branched channel of states exchange states. These state exchanges are useful off-chain due to their speed compared to on-chain exchanges. Moreover, via a contract, the on-chain receives the channel's final state upon the completion of every transaction.

Payment Channels: The scalability goal [107, 108] of blockchains is to handle payments with almost instant confirmation, cheaper fees, and limited transactionality. To achieve this, payment channels [5, 84] come into the picture as the adaptation of state channels for payment applications. As an initial design, payment channels were one-way channels [109], but they eventually evolved to two-way channels [84] so that both users could send and receive payments. In that manner, these channels process payments instantly. Also, channels prevent users from broadcasting each transaction on-chain and, thus, waiting for its confirmation. Still, this solution has drawbacks. Specifically because of the fund locking in the creation of the channel, which is not instantaneous due to the confirmation required from the on-chain. In consequence, to improve payment channels, there are proposals such as channel factories, virtual channels, payment channel hubs, and payment channel networks to enhance this channel solution:

- ***Channel Factories: [110, 111]*** Its premise centers on the locking of funds by many participants to finance a factory, intending to create channels for every pair of depositors. In cases where a direct channel is necessary, all depositors reallocate funds to create such a new channel. Although this approach does not require financing and establishing distinct payment channels for every pair of depositors, on-chain confirmation is still necessary after the creation of a factory.

- ***Virtual Channels: [112, 113]*** A virtual channel's existence, which resembles a direct channel, is determined by the locking of funds for a fixed time by all intermediaries between a payer and its payee. The set-up of such a channel implies a new virtual channel has to be established for every intermediary, and then it must supervise its closure. Its principal usefulness is that interacting with the on-chain is unnecessary while creating and closing a virtual channel.

- ***Payment Channel Network (PCN): [5]*** Payment channels initially required a direct channel to make payments, but this hampered scalability to some extent. To reverse this, PCN allows the creation of a network of channels in which a payer, without a direct channel to a payee, forwards a payment using intermediate nodes. In that way, intermediate nodes can earn an incentive through small fees. A key component in PCN is HTLC, a structure with payment conditions. The payer locks the funds until the transaction meets a locking condition, in which case the payee may use the funds again. Another restriction of the conditional lock is the expiry time, which encourages the intermediaries and the payee to achieve a quicker lock resolution. Overall, these transactions with conditionals must be atomic. Thus, the intermediaries participating in the transaction provide the security of said funds [114, 10].

- ***Payment Channel Hubs: [115, 116]*** This alternative aims to optimize PCN by employing a node hub. The node re-transmits the payment to nodes connected in a star topology in which this particular node is in the center. Through this setup, PCN could lessen the overhead involved in routing by interconnecting hubs to reduce the length of routes. This, hand in hand, reduces the cost of routing and additional expenses for the channels. It also reduces funds locked by single nodes. However, a hub requires locking in a significant amount of funds that increases with the number of channels and transactions.

State of the Art

4

Users do not care about what is inside the box, as long as the box does what they need done.

— **Jef Raskin**
about Human Computer Interfaces

LN is a channel payment network that aims to solve Bitcoin's scalability problem using off-chain transactions and has a market value of over \$336 million[1]. The 13,630 nodes and 51,863 payment channels make LN the largest deployed PCN [5] that uses Bitcoin as its underlying blockchain [13]. Due to its increasing popularity, there is a significant number of attacks on the LN in the literature that exploit its design vulnerabilities. Some manually discovered vulnerabilities have come to light, but today, there is no in-depth systematic analysis of the LN security.

4.1 Attacks over the LN

Of the different types of attacks on LN, one that receives the most attention is the ***griefing attack***, which aims to expose HTLC vulnerabilities. Such an attack intends to block as many channels as possible to stall payments. For instance, [117] provides a case of griefing attack where in a payoff between $Alice$ and Bob with hops through $Charlie$ and $Dave$ as in $Alice \rightarrow Charlie \rightarrow Dave \rightarrow Bob$, Bob shares with $Alice$ a hash $H = Hash(x)$. $Alice$ sends a payment with conditions to $Charlie$ to lock c coins during a time T_1. Similarly, $Charlie$ repeats the same process to lock c coins for a time T_2. In the end, $Dave$ sends the conditional payment to Bob to lock c coins for a time T, where $T_1 > T_2 > T$. Within the time T, Bob has to release x to collect the c coins from $Dave$. If Bob does not follow through, $Dave$ closes the channel before the timeout period and retrieves the locked funds from the contract. Bob does, however, succeed in locking c coins in every of the other payment channels for the upcoming T time [114].

Another griefing attack is related to timing assumptions necessary for HTLCs due to the atomic locking of funds and subsequent on-chain settlement. This process

[1] `https://1ml.com/statistics`

entails locking up funds from sending a transaction until the block ends. Based on the example above, when Bob claims funds from $Dave$, the protocol must restrain race conditions where the other nodes can withdraw funds if Bob is unresponsive. A race condition would occur when Bob claims funds from $Dave$, but $Charlie$ seeks to close his channel with $Dave$ before $Dave$ can obtain funds from $Charlie$. A solution is to use connectors [118] on payment channels to perform atomic swaps between them that are safeguarded based on an assumption of synchrony when there is an unresponsive counterpart.

In contrast, the attack performed on [119] overloads the payment channels with unresolved requests (HTLCs) until their expiration time. The channel is locked from receiving further payments if the number of unresolved requests reaches its maximum. The authors consider three versions in which an attacker 1) blocks as many channels with high liquidity as possible, 2) disconnects all the possible pairs of nodes, and 3) tries to separate individual nodes from LN. As countermeasures, the authors propose reducing route length, setting the maximum number of simultaneous payments based on the degree of trust, enforcing fast HTLC resolution, and avoiding loops.

Similarly, [60] reproduces a lockdown of the balance as a result of misbehavior by nodes connected to a specific channel. In a multipath payment, an attacker blocks intermediate nodes to give the adversary a dominant position in the network. The attacker's goal is to collect information from the nodes or increase the profits of a specific gateway node. Countermeasures to the attack aim to increase the value of the ratio of capacity blocked by the attack to the capacity required to carry it out. As such, loops on a payment path should be forbidden or at least minimized; in that case, the length of the cycles has to be greater than two to make the attack more difficult. Other approaches are reducing the maximum length of a payment path even at the cost of performance and adjusting LN parameters.

The authors of [120] present the idea of node isolation and channel exhaustion and demonstrate how the LN is vulnerable to these attacks. So, to remove a given number of nodes, an attacker could pursue a centrality-based strategy; instead, to achieve high efficiency, an attacker could use a higher-ranked min-cut strategy. As a result, these attacks can affect the network's average payment flow and payment success rate. However, using rate-limiting techniques in the client implementation could reduce the number of incoming -channels and -channel volume to mitigate node isolation attacks. In [121], the authors also determined that an attack following the centrality-based approach has a near-optimal effectiveness. It occurs when the attacker uses a node selection strategy based on betweenness centrality. Also, the attack is effective for a denial-of-service (DoS) where compromised intermediary nodes may drop or delay transactions. The attacker uses the node's position in the routing tree based on routing algorithms. However, the effectiveness of the attacks decreases in a less centralized network.

From the analysis performed on [122], the authors determined that LN has strong scale-free network properties, which makes it prone to DoS attacks. In this kind of attack, the target is specific nodes that are highly connected. An attacker targets those nodes with high centrality; as a result, the network connectivity is greatly affected. The research covers some strategic attacks to remove nodes based on randomness, high degree, high centrality, and community. This last strategy consists of extracting the network community structure and eliminating nodes according to their degree. Some defense mechanisms are proposed, such as random defense that arbitrarily chooses which nodes to link to the newly restocked nodes, preferential defense similar to above but considers the degree of the nodes, and balances defense akin to the former but considers the betweenness centrality of the nodes. As a conclusion of this research, the high-degree attack provides the desired effect even if the attacker does not know the entire network topology. In contrast, balance defense provides a better counter-attack effect as new nodes connect to low centrality nodes.

Another type of attack is the ***Flood & Loot attack*** [123] that triggers, in a broad systemic manner, the simultaneous closure of many payment channels. This attack overloads the blockchain with a high volume of transactions in which there is an improper settlement of some debts; thus, the attacker could steal funds. The authors also discovered by examining the fee estimation mechanism that an attacker gradually keeps lowering the transaction fees the victim would subsequently use to recover the funds. The victims can avoid this attack by correctly choosing LN parameters such as the channel's $feerate$ or the most unresolved HTLCs that can be accepted $max\_accepted\_htlcs$. Another countermeasure would be to increase $commitment\_broadcast\_delta$, which indicates the time for a node to unilaterally close a channel with unresolved incoming HTLCs. After a unilateral channel closure, a node has to publish the last committed transaction and the set of successful HTLC transactions to collect the incoming HTLCs on the Bitcoin network. Based on this guideline, some LN implementations release the successful HTLC transaction only after the commitment transaction has been confirmed. A straightforward solution would be to publish instantly each of these transactions to the network.

An attacker that acts as an intermediary in a payment route attempts to steal the fees of other nodes; such an approach is known as ***Wormhole attack***. The attack [124] occurs by excluding intermediate users from being part of the successful completion of a payment. Two adversarial users on a payment path can steal the payment fees for honest path nodes. The attack is as follows 1) *commitment phase:* Each user behaves honestly by locking funds to get a reward and 2) *releasing phase:*

As expected, honest users fulfill their HTLCs and settle their balances and profits in their payment channels. However, one of the adversaries behaves honestly with the next node on the path but cancels the payment with its predecessor, which continues this behavior until reaching the other adversary. This last adversarial obtains the releasing condition of the other adversary so it can deceive the other nodes on the path before the last adversary to fulfill the HTLC. Anonymous multi-hop locks (AMHLs) appear as a mitigation strategy, a cryptographic primitive for atomic swaps that impact privacy, security, scalability, and interoperability.

Other sets of attacks are those related to anonymity; these ***privacy attacks*** attempt to infer sensitive data about user identities. In that sense, [125] presents several attacks to discover private data on the network, such as the funds available in a node or the sender and receiver in a payment. All this, exploiting the publicly available information on the network. The objective of that research is to consider the main privacy properties of LN. One of them is the *private channel* in which the nodes that create a channel and the channel information are hidden, but this is compromised through a heuristic that determines the on-chain funding of this channel. Another property is the *third-party balance secrecy* where the channel balance remains a secret, but an attacker can use a generic method to discover the channel balances. The *on-path relationship anonymity* property refers to intermediate nodes in a payment path that should not know other nodes besides its predecessor and successor. Revealing this data is achieved by evaluating how well an intermediary node can deduce the sender and receiver of a payment it routes. The last property is the *off-path payment privacy* that relates to nodes not participating in payment routing and ought not to deduce any information. Based on the discovery of channel balances, an attacker can use this ability to create snapshots of the network and then determine where and how the balances shifted.

The analysis performed on [126] attempts to validate whether solutions to deanonymization attacks offer reliable guarantees. To do this, the authors modeled several anonymity solutions that, as a result, do not provide acceptable guarantees to their users in this regard. The model obtains the probability distributions connecting transactions to potential originators using Bayesian inference. This approach revealed that an attacker could deanonymize around half of all network transactions by colluding with a few influential nodes. A similar approach is that of [127]. But instead of analyzing anonymity solutions, it proposes a method that thoroughly reviews the code of the LN implementations to predict the sender and receiver in a multi-hop payment. The attack has two phases: finding nodes accessible by a simple loop-less path with matching timelock and creating lists of potential recipients and senders based on these nodes. The analysis determined that payment anonymity cannot be guaranteed substantially by the layered encryption used in onion routing when there is a nearly predictable path selection.

[128] is another approach that aims to deanonymize transaction information by analyzing whether or not there are any vulnerabilities in gossip and probing mechanisms that might allow them to infer transactional information and compromise privacy. For this, two threats are related to active and passive adversaries through 1) *Probing attack:* the adversary actively probes the target channel to determine the greatest amount transferable in a certain direction by analyzing the response messages. 2) *Timing attack:* the adversary determines the vicinity of the routed payment destination by passively analyzing the time deltas between delivered messages and responses. Instead of being a practical attack vector, the probing attack is known as a proof of concept. In contrast, in the timing attack, the distance to the original payment source cannot be ascertained because of the nature of LN routing.

Another type of anonymity attack is related to discovering the amount of funds a user has in its payment channel. As ascertained, probing attacks threaten users' privacy by discovering channel balances. However, such attacks do not consider parallel channels between nodes, yielding false results when using naive probing algorithms. In multi-channel hops where previous probing approaches were unable to obtain whole payment balance information, the jamming-enhanced probing model described in [129] takes into account parallel channels. The authors claim that different strategies can counter this attack, such as a new payment forwarding method, unannounced channels, rebalancing, and split intra-hop payments.

In the attack carried out by [130], the goal is to reveal the channel balance by sending multiple payments without any of them finishing and thus reduce the cost of the attack. The success of the attack is feasible since it is difficult to detect the attacker due to the nature of LN onion routing. First, the attacker $Mike$ creates a channel with a node $Alice$, which has a channel with another node Bob whose balance the attacker wants to reveal. Then, $Mike$ sends multiple payments to Bob by increasing the amount of each one until an error in the payment arises. To avoid the completion of the payments, $Mike$ creates fake invoices with a random hash $h(x)$ as mimicking Bob's invoice, which, in the end, denies the last hop payment. Various countermeasures can prevent the success of this attack, such as restricting access to debug messages or rejecting some payment requests selectively or randomly.

4.2 Performance of the LN

Throughout the existence of LN, the proposal of attacks with their countermeasures, several protocols, and network analysis have tried to improve the network performance. For instance, [131] aims to prove that the topology of LN is resilient to both directed and random attacks by measuring and describing it objectively. However, its approach is based on the use of discrete snapshots. As a counterpart, Flash [132] tries to improve routing performance by using payment characteristics. However, it ignores the measurement of the network as a whole and instead concentrates on the local view of each node.

In [60], it addresses the potential for availability attacks to impact the bandwidth of LN payment channels. The adversary exploits misbehaving nodes and disrupts the victim's role as an intermediate node in multi-hop payments. Through this attack and with minimal economic cost, the adversary can establish a lockdown for a reasonable time. The authors use Attack Effort Radio (AER) to measure the profitability of the attack, which is the ratio of the capacity required to carry out the attack to the capacity blocked by it. When the AER value increases, the attack becomes less profitable. Thus, the likelihood of a single payment completing a route with multiple hops lowers the AER value. To prevent the attack from being performed close to the victim, which decreases the AER value, the best is to minimize or forbid loops on payment routes. Also, when the length of a route increases, the AER value decreases. A countermeasure would be to reduce the maximum route length value. However, this value impacts network performance, as lowering it could eliminate routes for legitimate payments.

On the other hand, most solutions attempt to provide more reliability to LN by modifying the applicability of channel policy parameters. The proposal in [133] focuses on a multi-path routing payment scheme and fee functions. The routing payment scheme follows a multi-path atomic payment approach that drastically reduces user fees while keeping the network balanced. For the fee functions, beyond maintaining the balance on the network, they improve performance by specifying fees as linear functions over the continuous piecewise transfer amount. Another angle to take is [134], in which the authors determined which channel policy parameters may affect LN functionality through privacy, anonymity, and wormhole attacks.

As noted in that research, LN does not fully support extremely low-value transactions set by $htlc\_minimum\_msat$, as it discourages micropayments. Furthermore, a channel in LN can only contain at most 483 unsettled HTLCs defined by $max\_accepted\_htlcs$, which, with a small value, attackers can block channels. By setting these parameters, an adversary could launch a DoS attack with a minimal cost when the payment channels are configured with low settings for the number of in-flight transactions and the minimum transaction amount. To avoid such attacks, the authors propose path-selection policies as a countermeasure. However, they also note that a downside could arise, as achieving high payment success rates requires relying on large hubs that expose users to the risk of turning into honeypots or working together to deanonymize other users.

In LN, a payment on a route has as a conditional that it must have sufficient funds. But when payments travel in the same direction through a single channel, it gradually depletes and cannot support additional payments. The network's ability to route payments atomically is another feature that exacerbates the problem. This scenario could be worse by using routing schemes, such as shortest-path, that exhaust crucial payment channels and could eventually paralyze the system. A primary mechanism to make payments is its routing protocol, where, without a payer-payee direct channel, LN nodes process payment by receiving a routing fee.

In that regard, Spider [135] is a routing solution that achieves high-throughput routing using a multi-path transport protocol and packetizing transactions. The multi-path congestion control system handles all flows fairly and guarantees balanced channel use. Packetization allows even large transactions to be completed through low-capacity payment channels. To validate a protocol, transaction throughput per unit collateral is a crucial performance indicator as it measures the number of transactions and the value of transactions per second. In that sense, Spider's performance, compared to different algorithms and transaction arrival rates per sender, achieves an almost flawless average success rate, surpassing all other schemes.

Above all, the authors in [133] point out that the current LN routing algorithm is not ideal for providing optimum network performance. Authors claim that the routing problem depends on how intermediate nodes apply fees to process payments. This idea is consistent with the BOLT specification [136], in which the total fee that an intermediate node charges to forward payments comes from a proportional fee plus a fixed charge. Newly released payment routing algorithms, such as [14, 137, 132, 138, 139, 140], aim to improve LN by addressing routing failures, such as reduced routing fees or illiquidity inbound. Additionally, these new routing algorithms, which we cover in more detail in Section 4.4, aim to increase approval of LN as a viable solution to the Bitcoin trilemma.

Regarding performance analysis, the research in [141] performs a systematic measurement of LN performance using data collected over time. This measurement evaluated payment success rates and network performance during attacks. Moreover, payment channels were investigated in terms of their functionalities to validate their performance. Their LN analysis begins with building an undirected graph G by collecting data from nodes and channels. The authors ignore the direction of the channels because they are generally unknown and change over time. Based on the graph G, it allows studying both the network performance by analyzing the routing efficiency and network resilience and the communication performance by analyzing the characteristics of the pair channel. Routing analysis reveals that the success rate of routing in LN depends on the amount of routing. Resilience analysis reveals a strong yet weak structure. Channel research unveils more effective ways to employ LN. Although these findings highlight the current issues of LN, this work contributes to understanding the mechanics of the network and exploring the future ramifications of LN.

4.3 Node importance metrics for the LN

The node importance metrics in the LN are crucial to understanding the network dynamics and potential vulnerabilities. A research on the centrality of LN [142] highlights the significance of betweenness centrality in the routing of transactions. It also emphasizes the need for decentralization to mitigate on-path attacks and liquidity bottlenecks [142, 143]. The empirical analyses in [144] show that although the network has a notable degree of decentralization. However, a small number of nodes receive a substantial share of transactions, which introduces skew and increases centrality over time. Studies such as [143] also reveal that the network exhibits a scale-free generative model with strategic node interactions, where a centralized network is not optimal, and routing fees exceed marginal costs.

In another aspect, analysis of the network topology [145] shows the presence of key patterns like bouquets, with specific nodes playing critical roles that impact the connectivity and resilience of the network. However, removing these nodes can cause major disconnections within the network. Furthermore, [146] addresses the evolution of the network, demonstrating that LN has a centralized structure with nodes that actively participate and serve as hubs. This configuration results in the network's vulnerability to attacks targeted from within the system, where the removal of these core nodes may result in a significant decrease in efficiency [142].

Other studies have highlighted some metrics such as the Gini coefficient, Nakamoto coefficient, and core-periphery structure to assess node importance [52, 147]. The LN's reliability in routing payments is inversely related to payment volume, with just around one-third of destination nodes successfully reachable, indicating the significance of well-connected nodes [30]. Additionally, the network's structure and distribution, including node types like Eclair, LND, and C-Lightning, play a role in determining node importance and potential performance improvements or security risks [31]. By analyzing these metrics, researchers can enhance the network's efficiency, security, and overall resilience, paving the way for a more robust off-chain payment ecosystem.

On the other hand, analyzing the LN topology is essential to understanding its intrinsic elements, particularly the process that handles the payment routes. LN topology has been one of the most extensively analyzed topics since its launch, mainly because increasing user transactions on the Bitcoin payment network is its principal objective. Most studies base their analysis on snapshots taken over an extended period, such as [148] or [131], with the latter being more compact. Other studies, as in [149, 150, 146, 131] make use of graph theory tools to model LN by collecting basic information about the channels, such as the existence of a channel and its capacity to make payments. Also, the approaches [148, 141, 149, 146, 120, 131] model LN as a network that is both undirected and weighted. These approaches consider that the distribution of funds across channels is unpredictable and that its direction constantly changes over time. Also, as pointed out in [149], as the network size squares, the total amount of transactions increases proportionately. However, the LN distribution is unequal because a small set of nodes holds most bitcoins.

The studies shown in Table 4.1 provide insights into the network topology with its main features and metrics. However, they lack more depth on significant channel properties, such as the balance on each side of a channel, the fees charged to make payments or existing HTLCs. In fact, of all the studies mentioned so far, only [148] models LN as a multigraph. This model considers that a pair of nodes can have various channels. In contrast, there are approaches, such as [120], that model LN in its most basic form through a graph that does not use the channel directions or their weight but only its nodes and channels. Nevertheless, most studies present features of LN centrality depending on the metrics used, such as [149, 150, 120] for degree, [141, 149, 120, 131] for betweenness, or [141, 149, 131] for closeness.

Likewise, authors in [148, 149, 120] consider that a centralized network is the most effective way to characterize the structure of LN, which is consistent with the structure of a core-periphery network as defined in [149]. On the other hand, the objective of some studies is to establish which centrality metric, beyond the usual ones, best defines this network. For instance, through a core-periphery network as in [149], the awaited payment success ratio and the average maximum flow as in [120], estimated revenue based on number of unsuccessful transactions and traffic volume as in [148], or the mean local effectiveness across all nodes as in [146].

Similarly, various metrics have been proposed to evaluate node importance, such as the Generalized Economic Complexity Index (GENEPY) [151], the Maximum Betweenness Improvement Problem algorithm using Advantage Actor-Critic models [152], and the analysis of node centrality and network structure through the Undirected Binary Configuration Model (UBCM) [153]. These metrics consider factors such as node connections, routing opportunities, and network resilience. Additionally, LN's high concentration of bitcoins among a small percentage of nodes raises concerns about network centralization and potential vulnerabilities to split attacks [149]. Understanding and utilizing these node importance metrics are essential for optimizing the LN's performance, security, and scalability [134].

Proposal	Period	Nodes/Channels	Metrics	LN Modeled features
[131]	Jan19	2344/16617	• Centrality: Betweenness and Closeness • Percolation threshold • Local clustering coefficient	• Channel capacity, the sum of each balance separately, as weight in an undirected network.
[146]	Jan18/Jan19 12 snapshots	4189/67917	• The mean local efficiency across all nodes	• Channel capacity as weight in an undirected network
[150]	Jan18/Jan19 12 snapshots mainnet	3613/23860	• Lower strength • Median degree	• Channel capacity as weight in an undirected network
[141]	Apr18/Apr19	7796/41705	• Effective eccentricity • Assortativity coefficient • Centrality: Betweenness and Closeness • PageRank	• The sum of channels' capacities between a pair of nodes as weight in an undirected graph • In the event that two nodes are not connected by an edge, the channel capacity is used as weight.
[148]	Dec17/May19 40 snapshots	4787/N-A	• Revenue estimate based on traffic volume • Node's unavailable payment failure count	• Undirected weighted multigraph
[149]	Jan18/Jul19 18 snapshots	8216/122517	• Gini • Centrality: Degree, Eigenvector, Betweenness, and Closeness • Core-periphery structure	• Channel capacity as weight in an undirected network • Number of user nodes • Symmetric and Adjacency matrix • Channel capacity • Number of open channels
[120]	Oct-Nov18 and Jan-Feb19	±2500/N-A	• Diameter and Distance • Average path length • Centrality: Degree and Betweenness • Scale-free networks • Small-world networks • Network's node cardinality • Number of reachable nodes • Expected payment success ratio • Average maximum flow. • Average fee gain	• Number of user nodes and channels • Connected components

Tab. 4.1: Measurements on LN graphs

Additionally, one can follow LN mainnet evolution through independent projects like 1ml[2] that provide statistics about relevant data such as the number of nodes and channels, capacity, average node, and channel capacity, to name a few. Based on the available data, the majority of research analyses regard betweenness, closeness, and degree as attributes that more effectively portray the properties of nodes. Besides the purpose of finding out the evolutionary topology of this network, there is a greater interest in proving whether it is resilient or not against attacks (split and topology-based) and random failures [150, 120]. A main concern is the users' privacy issue because of network centralization. It shows up due to the nature of sending payments through nodes using a multi-hop algorithm. Specifically, a hub node may be able to collect data or deny the forwarding of a transaction by either censorship or fee increase since it can be a main hop in the payment route [154, 5].

On the other hand, the authors on [41] demonstrate that the LN network suffers from a longer convergence delay than expected following the protocol specification. As a result, significant failures in payment attempts occur due to delays. Node delays arise due to obsolete routing data when calculating a route, which requires reducing these delays to improve the throughput of routing protocols. To mitigate

[2]`https://1ml.com/statistics?json=true`

such drawbacks, the authors focus on two paths: using different gossip protocols or modifying the parameters of the current gossip protocol that uses a staggered transmission mechanism. For the former, several approaches were used for gossip algorithms based on flooding, global spanning trees for a broadcasting structure, Minisketch-based set reconciliation, and inventory-based gossip.

The measurement of these approaches was not only in terms of convergence delays but also in terms of the impact on payment attempts and bandwidth usage. As a result, the algorithm with the highest bandwidth usage is flooding, although its convergence delay is low. On the contrary, the spanning tree algorithm has the lowest convergence delay and the lowest bandwidth usage. For the latter, although each LN node implementation has its version of information dissemination protocols, the parametrizations of those implementations significantly impact the convergence delay. To avoid a too drastic change in the LN implementations, a modification to the choice of parameter value for the staggered broadcast might keep the staggered broadcast's rate-limiting characteristics. It also addresses the significant convergence delay.

The research in [155] takes a different view by focusing on mass exit attacks in LN. Specifically, their interest lies in understanding how attacks affect the network when adversaries attempt to lock funds for a period that exceeds the LN protocol's limit, such as in zombie attacks. Likewise, the loss of funds when sending transactions that close channels using expired protocol states, such as in a mass double-spend attack. LN is a scale-free network with a power-law degree distribution in the channel distribution to nodes due to its topological characteristics [146]. In that case, a small alliance of hostile nodes can carry out such attacks. This fact was more relevant after the scenarios and setups used to test both attacks. As a countermeasure to both attacks, the authors propose increasing the block size to decrease the effects of the attacks at the cost of a decrease in decentralization.

In the mass double-spend attack, the countermeasures proposed are watchtowers, mempool monitoring, and parameter modification. The watchtowers proposal protects against fraudulent commitments when the user is offline, thus preventing the loss of funds. The mempool monitoring proposal is a distinct watchtower strategy that detects hostile transactions in the mempool whereby endeavors to send a transaction ahead of the adversary to close the channel using a higher fee. The last proposal, parameter modification, is to modify the value of the to_self_delay by increasing its value to lessen the harm done by the attack. In the case of an unfriendly channel closure, this parameter's goal is to determine the delay or amount of time, expressed in blocks, that the opposing side on the channel has to wait to withdraw the funds. As a new perspective, in Chapter 5, we cover the tuning of

LN parameter values to evaluate the security and performance of the network and provide recommendations for those values.

On the other hand, several contributions based their analysis on the network disruption by choosing specific nodes to isolate. The data analysis of snapshots and their corresponding results allowed us to ascertain that LN suffers from unequal wealth distribution and node centrality. From each snapshot, only relevant data, such as the number of nodes, channels, and channel capacity, are essential for analysis. It is worth mentioning that the analysis focused on centrality measures, which, as mentioned above, represent the properties of nodes. Therefore, in Chapter 6, we provide a new proposal to evaluate centrality in LN using classic centrality and alternative metrics, as well as network properties.

4.4 Routing protocols

Routing protocols on the LN serve the crucial purpose of determining the path for off-chain transactions, ensuring they are fast, cost-efficient, and secure while maintaining user anonymity [38, 127]. The protocols, such as those used by LN's predominant routing clients like LND, C-Lightning, and Eclair, play a significant role in selecting the optimal route for multi-hop payments. For that, it has to consider factors like path length, maximal delays, fees, and success rates of transactions [156]. However, the existing routing algorithms' low randomness in path selection poses challenges to user anonymity, allowing potential attackers to compromise the privacy of senders and recipients [157]. Other challenges that face LN are finding paths with sufficient funds, dealing with unidirectional channels, and ensuring atomic delivery of payments, which can hinder successful transactions. To address these issues, new routing algorithms are being proposed to enhance user anonymity by creating less predictable transaction paths, highlighting the trade-off between anonymity and transaction fees in the LN.

Although LN covers all aspects related to instant payments using a network of channels, its structure is formed by distinct protocols. These protocols cover aspects such as peer channel (establishment, usual operation, and closing of channels), communication (P2P node and channel discovery), invoice (destination and purpose of payment), and onion-routed packet (payment routing from an origin node to a destiny node). We summarize in this section the main idea behind each of the most relevant routing protocols. There are five different approaches: distributed hash tables (Flare [14]), landmark routing (SilentWhisphers [138]), network embeddings (SpeedyMurmurs [139]), flow-based (Spider [137] and Flash [132]), and ant behavior (Ant [140]).

4.4.1 Flare

The Flare [14] protocol uses distributed hash tables to find optimal routes where nodes store routing data of neighbors within a certain hop distance. Beacon nodes are crucial, as most nodes connect to them since they are closer to node addresses. Connecting to a beacon allows a node to access the local view of its connected neighbors, expanding its network outreach. In cases where a direct route is not found, nodes collaborate to locate a path to the receiver using routing data from the nodes closest to the receiver's address. Flare addresses specific limitations of the LN, such as allowing the payer to choose the payment route independently, overcoming incomplete network views, operating efficiently in terms of time and memory, and minimizing fees in routing decisions.

Flare solution is similar to a hybrid mobile ad-hoc network approach. It deals with slow (payment channels) and fast-changing (node status, fund distribution) information in LN. It is better to gather static information than dynamic, which can be unpredictable and memory-consuming. Every node has a routing table that contains information. The Flare algorithm interacts with nodes proactively to find neighbors and reactively to rank and select a route, as follows:

- ***Route discovery (proactive part):*** It involves a node's scheduled routing table gathering available channels within the network to find paths to the payee or beacons. This proactive part helps increase network awareness by adding arbitrary nodes outside the payer's immediate vicinity.

- ***Route selection (reactive part):*** This process involves a payee integrating its routing tables with payees to determine the most likely routes. If no route is found, the payer uses the payee's beacons and selected nodes' routing tables. The algorithm ranks the routes using a cost function based on static and dynamic information. Thus, ranking follows two steps:

  - *Static ranking:* ranks by only using information (static) from routing tables that yield a list of potential routes.

  - *Dynamic ranking:* ranks based on most likely static and dynamic information by sending an onion-wrapped polling message to each route among the candidate ones to collect updated information.

Based on the collected data and the ranked channels, the payer selects the best route. The algorithm uses multiple paths on the reactive step to ensure fault tolerance. If a node is unresponsive, an alternate path is chosen. If no viable routes exist, the algorithm consults the beacons routing tables and repeats the process.

Routing Table: A node's routing table comprises static data that includes its overall capacity and information about routes to other nodes. Moreover, routing tables store channels according to the following characteristics:

- ***Neighborhood map:*** Channels connect all nodes within the range of neighboring nodes.

- ***Beacon paths:*** Channels that construct routes to beacon nodes in the address space near the node.

- ***Cached payment routes:*** Channels that establish connections through routes to previously used nodes to make payment transfers.

A node could successfully find paths of shorter length when it possessed a larger routing table and, therefore, a larger number of nodes collected in it. However, a larger table presents constraints regarding network bandwidth and computational power, which must be increased to maintain them.

Neighborhood Discovery: A node updates its neighborhood information by accepting the following messages, but only from nodes that are nearby:

- ***NEIGHBOR_HELLO:*** As a signal of existence, an LN node sends out the whole routing table.

- ***NEIGHBOR_UPD:*** The node's routing table receives incremental changes with each new update message.

- ***NEIGHBOR_RST:*** In case a node determines that update information is supposedly lost or its routing table is corrupted, the node sends this message that indicates its wishes to receive a fully updated version of the routing table belonging to the recipient of the message.

- ***NEIGHBOR_ACK:*** This message is a response to the first two messages by acknowledging that the node processed the message correctly.

A node must maintain a minimum interval between sending presence and update messages to prevent DoS attacks. It updates its routing table in response to a message, considering the channels within its neighborhood. Otherwise, it ignores the channels. An event-driven approach processes communication independently with adjacent nodes; thus, a set of processing rules must be followed:

- To save bandwidth and computational resources, inactive nodes do not receive messages.
- In connection with the synchronized view of the routing table, it transmits an incremental update with each update message.
- There are no pending messages of presence or update since a pending message must finish before the routing table of the node is updated.
- A presence message is sent during the creation of a channel or as a response to a reset message.
- A timeout avoids spam messages of type reset.
- The owner of a node has the option of delivering filtered changes to the routing table of the node.

Beacon Discovery: The purpose of beacon discovery is to expand the node's awareness of the network once it finds its neighborhood. The closest nodes in the address space are the selected beacons for a node. Like neighborhood discovery, this mechanism is based on the following messages:

- ***BEACON_REQ:*** A node receives this message, which indicates its selection as a beacon candidate.
- ***BEACON_ACK:*** This message is a reply to the previous one and indicates that the node agrees to be a candidate.
- ***BEACON_SET:*** A node receives this message, indicating it has been designated as a beacon.

Beacon discovery is a technique that a node can use at any time to update the beacon set after it has identified its neighborhood. As a result, a node searches the address.

4.4.2 SilentWhisphers

SilentWhispers [138] protocol is a novel solution for credit networks that ensures transaction integrity and privacy by combining digital signature chains with secret-sharing-based multiparty computing. It is designed for cross-currency transactions and has an inherent tolerance for inconsistencies, unlike networks like Bitcoin, Stellar, and Ripple, which rely on consensus-based ledgers. However, credit networks cannot guarantee privacy due to linkability and de-anonymization attacks. SilentWhispers does not use max-flow for routing, as it does not scale for large networks or distributed ones. Instead, it uses landmark routing, which calculates a fraction of routes between nodes.

Landmark routing protocol differs from classical technique by being suited for distributed environments, utilizing landmarks as intermediate nodes for route discovery. Once nominated as a landmark, its information is available to the entire network. The landmark node mechanism uses the breadth-first-search (BFS) algorithm to find the shortest distance between all nodes and the landmark, referred to as arborescence, and its inverse, anti-arborescence. The system also uses a secure multi-party calculation protocol to compute available bandwidth for discovering routes, protecting anonymity.

To modify the behavior of a centralized to a distributed credit network, operations such as the following must exist that execute in a distributed manner:

- *chgLink* and *testLink*: executed locally by a pair of users that share a corresponding link.
- *pay*: has three main steps:
  - Sender and receiver reconstruct, from the arborescence and counterpart generated afterward the routing protocol is performed, the transaction paths through the different landmarks.
  - The credit available for each path depends on the least accessible credit among the credits offered by each connection in the path.
  - The sender reduces the available credit in the routes by the whole amount that corresponds to the required transaction value.
- *test*: is comparable to the pay algorithm, without the stage when the sender lowers the amount of credit accessible in the routes by the transaction value.

Trust in the network is achieved through landmarks, which act as network operators. These landmarks calculate the lowest route value throughout transactions and require recalculating routing information due to constantly changing credit networks. Loose synchronization is a unique feature of users, leading to BFS arborescences and anti-arborescences as epochs. Users use their routing information during each epoch, which is created at the start. BFS must be accurate to ensure that every user receives routing information from every neighbor for every landmark.

The protocol ensures privacy on the credit network through the secure multi-party computation (SMPC) protocol used by landmarks. When a link is made between a sender and a receiver, a portion of the credit is given to the landmark, allowing the sender to get their portion. Security is enhanced by combining long-term and fresh keys and including timestamps in signatures to prevent rollback attacks. The protocol for the credit network $\mathcal{F}_{CN}$ maintains static information about nodes and links using a matrix, tracking credit shifts between nodes over time. Consequently, $\mathcal{F}_{CN}$ contains a set of functionalities:

- $\mathcal{F}_{CHGLINK}$: Every pair of neighboring nodes can generate a new credit or update the existing one on a link.
- $\mathcal{F}_{TESTLINK}$: Each node can test the credit available in its adjacent links.
- $\mathcal{F}_{ROUTE}$: $\mathcal{F}_{NET}$ is used as a synchronization mechanism to update the routing information of the nodes. Thus, in order to provide transaction routes between two nodes, it builds BFS trees (arborescence and anti-arborescence trees).
- $\mathcal{F}_{PAY}$: facilitates the pay process initiated by the sender and connects them to the receiver through two paths created by each landmark. This functionality communicates with intermediary nodes, which determine $\mathcal{F}_{PAY}$'s future path. The sender receives the computed total credit amount for each route. Nodes can confirm or cancel transactions if the amount exceeds the link's capacity.
- $\mathcal{F}_{TEST}$: Determines how much credit is accessible on the paths connecting two nodes.
- $\mathcal{F}_{ACC}$: Resolves disagreements on the link value between pairs of nodes.

4.4.3 SpeedyMurmurs

SpeedyMurmurs [139], built upon VOUTE [158], is a protocol that enhances the privacy of greedy embeddings by using anonymous return addresses instead of node coordinates. When managing weighted connections and updates, it employs an embedding-based routing algorithm that is comparable to tree-only routing to preserve privacy. This approach addresses SilentWhispers' drawbacks by assigning unrepeatable coordinates to each node based on its position in a spanning tree, creating a distributed, rooted spanning tree for the network.

The Path-based Transaction (PBT) network involves a greedy process for nodes to give themselves vector coordinates, with the root node being the only one with an empty vector. Parent nodes provide their child nodes their coordinates. When a node that is deciding which way to route finds the path between its neighbors and the destination node. The shortest path length between two points determines the distance on the spanning tree between the sender and the recipient. The model (G, w) describes the PBT network through a directed graph $G = (V, E)$, where w is a weight function that determines the number of funds to send between adjacent nodes over an edge. This is achieved by defining two sets for incoming and outgoing neighbors of a node and a route through a link arrangement. The funds available on a route are the minimum weights on the links, determining the node's net balance.

To perform the routing operations, the protocol uses the following algorithms:

- *setRoutes(L)*: given a set of landmarks, this algorithm sets up the routing information that every node requires.
- *setCre(c,u,v)*: c specifies the value of the weight function of the nodes sender u and receiver v, which may change the routing information produced by $setRoutes$.
- *routePay(c,u,v)*: returns a group of tuples with paths and funds, where each pair is the funds routed through a given path.

The protocol ensures value privacy when the entire transaction value cannot be determined by the adversary. When utilizing transaction privacy in a path-based transaction, sender privacy ensures that the adversary cannot identify the sender. The key idea to outperform SilentWhispers is by modifying VOUTE, which contributes to:

1. Find out the existence of unidirectional links through a two-phase construction algorithm,
2. Decide when to use maintenance that is available on demand and
3. Utilize a path discovery method to adaptively select linkages based on neighbor coordinates and available credit. Therefore, in addition to embedding-based routing, Speedy Murmurs implements path-splitting credit across routes before path discovery.

4.4.4 Spider

Spider [137] is a protocol that uses a flow-based algorithm to find its route. It transmits data across the network by dividing it into blocks and transferring them using network nodes, similar to packet-switched networks. The process aims to balance the network by performing balanced payment flows between incoming and outgoing transactions. The node divides payments into equal transactions sent through routes, maintaining channel balance. Although the protocol does not consider channel unbalancement, it eventually overcomes it through routing.

The protocol prioritizes routes that rebalance channels, replicating on-chain transactions. Spider introduces imbalance-aware routing to maintain balanced channels, considering rate-imbalance constraints through rate control and optimization-based routing. It also maximizes the use of payment channel funds through in-network transaction block scheduling and congestion control. The protocol also manages the speed at which nodes send transactions for payments. The mechanism that Spider considers is through two types of nodes:

- **Hosts:** Hosts use message-oriented transport to send payments, requiring parameters such as the amount, deadline, recipient address, and routing fee. Transport offers two payment interfaces: atomic and non-atomic. Atomic Multi-Path Payments (AMP) divide a payment into multiple paths or use a single "base key" to generate keys. The latter can partially deliver the payment, with the sender receiving notification about the amount sent by the deadline without more transactions. The congestion control algorithm determines the rate at which transaction blocks are sent for different payments.

- **Routers:** Onion routing allows routers to forward transaction blocks to receivers for user payment privacy, similar to LN. However, the router must wait for the key and queue them when funds are insufficient. This approach can cause longer payment delays for certain transactions. To overcome this drawback, services have the ability to rank payments according on size, deadlines, and routing costs, scheduling transaction blocks accordingly.

4.4.5 Flash

Flash [132] addresses the limitations of traditional solutions like Flare, SilentWhispers, and Speedy-Murmurs, which rely on dynamic and static routing. Spider, on the other hand, uses a defined number of paths for payments to achieve higher performance through dynamic routing. It uses a flow-based algorithm to find routes, balancing optimal routes with probing overhead. The Flash algorithm categorizes transactions between small (mice) and large (elephants) values, ensuring a balance between probing overhead and optimal routes.

Different routing strategies are used for payments to achieve a balance. Elephant payments optimize to reduce routing fees by utilizing a modified max-flow method to determine the maximum flow from a collection of k paths. Mice payments send the entire payment through one of m-shortest paths, reducing probing overhead by using precomputed paths from a routing table. Unsuccessful payments are retried with partial payments through paths until the transaction is completed.

Careful consideration is necessary for dynamic routing implementation, especially with optimal paths that incur costly network probing. Off-chain network channel balances change with payments, requiring probing overhead for each payment. Traditional computer networking distinguishes between elephant and mice flows to balance optimal path and probing overhead. In mice flows, it permits static paths for minimal probing and many paths for optimal performance in elephant flows. This protocol finds potential routes with sufficient capacity for payment division and lowers transaction fees for elephants by modifying the max-flow algorithm.

The protocol for mice payments uses computed paths to minimize probing. The Ripple network has small payments, making it unlikely that mice payments overwhelm a channel. However, elephant payments require multiple routes, improving success and ratio. Transaction cost, including liquidity and set fees, is crucial for minimizing fees, especially given the significant volume of elephant payments.

The protocol distinguishes between elephant and mice payments and employs various routing algorithms. For elephants, it uses an Edmonds-Karp-modified max-flow algorithm. To lower transaction fees, it splits payments across paths using an optimization algorithm. A capacity matrix C records polled channel capacity, while a residual capacity matrix C' records the remaining capacity after applying the Edmonds-Karp solution. C' reflects the flow found by the shortest path p.

The algorithm identifies paths P with enough capacity and then seeks the optimal solution to route across P, reducing transaction fees and data collection. A rating function $f_{u,v}$ shows the fee collected by channel (u, v), with r_p representing a partial payment and $f_{u,v}(r_p)$ the fee collected. $a^p_{u,v}$ determines channel usage. An optimization algorithm minimizes fees while meeting channel capacity and payment demand d, with a local ledger keeping track of channel changes.

To decrease probing overhead, the mice approach makes use of a small number of precomputed paths. Every node has a routing table, and it uses Yen's algorithm to determine the top-m shortest paths when a new receiver is not included in the table.

The routing table refreshes when the network topology G updates. When selecting m shortest paths, a trial-and-error loop is used. The chosen path sends the entire payment via p random path, and if successful, the protocol ends. If unsuccessful, the sender tries a different random path to determine its capacity, sending a partial volume payment through p. The random selection of paths helps maintain load balance without knowing the instantaneous capacities. A payment failure occurs when all M paths are used, resulting in unsatisfied demand.

4.4.6 Ant

The protocol [140] is inspired by ants' coordination in obtaining food. It involves initiating gossip messages between neighbors, which reach an intermediary node for a discovery route. To maintain privacy, the origin and destination exchange a secret. This secret is used to create pheromone seeds, with the first bit being the only difference. This protocol ensures that no one is aware of the route and maintains confidentiality.

The algorithm uses pheromone seeds to transmit seeds to neighbors until one or more intermediate nodes receive them and verify if a matching seed exists. If a match does exist, the intermediate node sends the data back to the predecessor node, ensuring the seed reaches its origin and destiny. The algorithm considers a network with bidirectional and unidirectional channels, open communication channels, and no limitations on channel volume or fee collection. Therefore, when sender A requires sending a payment to receiver B, the nodes perform the following operations:

- Nodes exchange in a secure way random 128-bit numbers $R(A)$ and $R(B)$.
- Nodes create pheromone seeds S by combining a bit (0-sender, 1-receiver) with hash $R = h(R(A)^\frown R(B))$ to form $S(A) = 0^\frown R$ and $S(B) = 1^\frown R$, sharing them with neighboring nodes via open payment channels.
- The sender waits for an answer from its neighbors, indicating a path has been found.
- The receiver waits for news from the sender that a path has been found.

Besides S, the algorithm uses a derived seed S', which is the hash R, and the conjugate seed $\overline{S}$. For that, $\overline{S}$ is the inverse bit of S (0-receiver and 1-sender). On the other hand, the nodes perform the following tasks to get a route:

- For the routing tasks, each node reserves a fast-access memory space known as LNmempool.
- LNmempool keeps a numbered list of neighbors together with the relevant information about its payment channel.
- When a S arrives at a node, the node has to check if S' is not the derived seed of a seed already stored in LNmempool.
- If the node has not stored S', it stores S in LNmempool with the information of the neighbor that sent S and broadcasts S to other neighbors.
- If S' is stored, then the node checks if S is stored as well:
  - If S is stored, it adds the data about the new transmitter neighbor.
  - If S is not saved, $\overline{S}$ is, hence a match results.
- When a match results, the node creates a matched seed $S_m = 0^\frown S$ and transmits it to the neighbors from whom it received S.
- The node that receives S_m broadcasts it back to the neighbors that sent him the unmatched seeds.
- The sender receives back several matched seeds, chooses one to construct a confirmed seed $S_c = 0^\frown S_m$ and sends it back to the neighbor that sent it S_m. The nodes broadcast back S_c to the subsequent neighbor from which it received S_m until they reach the node that did the match. That node broadcasts back S_c until it reaches the receiver.
- Once the sender receives confirmation from the receiver of the payment path, it starts the conditional payment chain.

Upon completion of payment, nodes delete data in S_c and erase all data related to matched and unmatched seeds older than threshold time τ. If no path is found after τ, path-finding requests are erased to maintain LNmempool size. To ensure a balance, the sender and receiver add an "amount field" to S so nodes only broadcast to neighbors with sufficient balance in open payment channels.

Obfuscation can help maintain anonymity by using multiple transactions through micropayments. In this process, S includes a "current fee field" set to 0 and a "maximum fee field" set to the sender's maximum payment amount. When broadcasting

S, a node checks that the current amount and fee are less than the maximum fee. If so, it adds the fee to the current fee field. The total fee is the sum of both S's current fee and the matching node's fee.

Part II

Contributions

5 LN Contract Parameters Selection

Simplicity, carried to the extreme, becomes elegance.

— **Jon Franklin**
(Computer scientist)

In previous chapters, we provided some basic definitions of LN, the messaging in the network, its functionality, and the basic setup of the parameters that shape this network. From this chapter onwards, we provide our contributions to this network. Specifically, in this chapter, we give a glimpse of how to enhance the performance and security of the LN network.

In the context of multi-hop payments, the contract parameters not only determine the security level but also the network's effectiveness as a payment mechanism. Therefore, it is crucial to evaluate how tuning contract parameters impacts LN performance and security. This assessment includes recommendations on optimal values for these parameters, considering the trade-off between the utility of this network and its security.

5.1 Multi-hop Route Parameters

Let us consider the next scenario, $A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_i \rightarrow \cdots \rightarrow A_{n-1} \rightarrow A_n$ in which node A_i may not proceed with the payment due to an unknown reason. In that case, locks will arise among the hops performed between nodes A_1 and A_i. The reason comes from the policy taken by A_1 that sets a time that bounds the total locking time, known as the absolute expiration block height θ (total timelock). On the event of an unsuccessful payment, θ sets the time to release the amount involved on a locked payment route.

5.1.1 Time-lock parameters

As nodes route the payment to their subsequent nodes, the value of θ decreases after each hop. The value by which the total timelock is decreased at each hop is advertised by each node of LN, and is known as $cltv\_expiry\_delta$ [45] (δ), i.e. δ defines the tolerated difference in blocks specified by each node along the route. It is worth to mention that the value of δ may differ depending on the direction that a transfer traverses the same channel, because each node sets that value. With regard to the target node, the $min\_final\_cltv\_expiry$ [159] value comes into play instead of δ.

On the other hand, the public data released by each node allows to create a route from a source node to a target node. The source node provides the initial θ value, that is decreased at each hop on the route. For a payment path to be valid, the last hop must still be able to set a timelock he agrees to, that is, at least $(\theta_0 - \sum_{i=2}^{n} \delta_i) > 0$. Although, this mechanism allows to source node to limit the duration of a locktime over a payment, a malicious node could try to lock indefinitely the funds of intermediate nodes by setting an initial large θ. Moreover, among the LN implementations, the δ value can be set by default to either 40 blocks or 144 blocks.

5.1.2 Limit parameters

To avoid long timeouts, nodes set their $locktime\_max$ (T_{max}), the maximum time they allow for HTLC expiration values in outgoing payments on the channel which by default is 2016 bocks (2 weeks) [119]. In consequence, the relation $\theta < T_{max}$ must always hold, to provide a node the option to accept a payment as an intermediate node on that route. Otherwise, the intermediate node must refuse to route the payment and the source node must compute a different path. Besides the maximum hop limit (set to at most 20 hops [119]), form part of the parametric consideration taken by a source node to create a payment route.

Despite the parameters explained above, an intermediate node goal is to make a profit from routing payments. Whence, at the time a source node constructs a route, it must gather specific information about fees or minimum payment amount that each intermediate node will charge or agree to transfer during using its channel. As part of that information, nodes propagate the minimum amount payment, in millisatoshi, that will agree to transfer, known as $htlc\_minimum\_msat$ [160]. When a node sets the $htlc\_minimum\_msat$ parameter, its value gets broadcast to the whole network via the $channel\_update$ message. In the case of an update, it is part of the miscellaneous updates category [41] within the channel update gossip message.

5.1.3 Fees related parameters

A node reveals a couple of values related to fees: fee_base_msat [159] , fee comprised on each HTLC as the constant amount charged by a node that performs a transfer, and $fee\_proportional\_millionths$ [159], fee that increases proportionally per amount transferred. To calculate the fees, the amount transferred in the transaction is in millisatoshi:

$$fee\_base\_msat + (transferred\_amount * fee\_proportional\_millionths/1000000) \tag{5.1}$$

We intend to provide recommendations on the optimum values of δ and T_{max} configuration parameters, to provide the best trade-off between the efficiency of the network and its resilience to attacks.

Detailed explanations of the main concepts about the LN can be found in the literature [133, 161, 5]. The BOLTs [136] also provide a more technical description of LN specifications.

5.2 Metrics

In order to determine the most appropriate values for the parameters that we want to assess, we define two different sets of metrics. Such metrics are somehow opposite since the extreme values for one of the sets produce poor results in the other one, so a trade-off between both values has to be achieved.

5.2.1 Performance

The Lightning Network (LN) is meant to perform payments between users. With this general objective in mind, we could measure the performance of the LN based on the possibility that two different users of the network would be able to perform a payment between them. However, as we point out in Section 2.2.6, not all pairs of nodes in the LN share a channel so the majority of payments between nodes are performed through multihop routes. A may perform a payment to B if there is a path between both users with enough funds, and the configuration parameters of the implementations allow to do so.

We measure the performance by repeatedly picking two random nodes in the lightning network and trying to perform a payment between them. For each chosen pair, payments of different amounts are attempted, from 1 to 4,294,967 satoshis (the maximum payment commonly allowed in the LN implementations), tacking as intermediate amounts all base 2 possible values between those limits. Performance is measured, for each amount, as the percentage of successful payments from the total number of attempts.

Notice that path availability is a feature that depends on different parameters. First of all, it depends on the topology of the LN, defined by each of the channels created in the network. A path must exist between A and B to allow a payment between them. However, such basic requirement is not the only one needed. Capacity of each channel that the payment traverses must be bigger that the payment amount itself. In fact, even this condition is not enough since the channel capacity must be properly distributed and the balance of each member of each channel has to be greater than the payment (in the right direction of the payment).

Nevertheless, none of the above requirements to perform a payment between two users depends on the parameters that we want to evaluate. An intermediate node in a path may refuse to route a payment if the proposed HTLC expiration time does to meet his requirements, either because it is to high (higher than his T_{max}) or because it is too low (and thus can not ensure a difference of δ with the next hop's HTLC expiration time). For this reason, lowering T_{max} or increasing δ configuration values in the LN clients may reduce the probability that a random payment can be successfully performed in the network.

5.2.2 Security

Recently, security of the LN has been analyzed in different research papers and some attacks have been presented. One of those attacks [60] may lock the funds of a victim by performing payments through that node that take longer to finish than needed. The attack takes advantage of the multihop ability and the possibility to loop a single route multiple times through a single user. The severity of the attack can be measured with two different parameters (as defined in [60]): the Attack Effort Ratio (AER) and $\Delta(b)$ function.

Definition 5.2.1 (Attack Effort Ratio (AER)). This expression is the ratio between the capacity needed to perform the attack and the capacity that the attack blocks, i.e.,

$$AER = \frac{C_{attack}}{C_{blocked}} \tag{5.2}$$

AER measures the profitability of the attack. The lower the AER, the more efficient the attack is in economic terms, and thus the higher the incentive for the adversary to perform such attack. To measure the time during which the balance is locked, the Δ function can be defined.

Definition 5.2.2 ($\boldsymbol{\Delta(b)}$ function). This function is a time based decreasing function that measures the total capacity blocked w.r.t. the time during which the attack has been conducted. The block generation count, b, is used as the time unit for this function.

For instance, $\Delta(0) = C_{blocked}$ since it provides the total capacity blocked at the initial time of the attack, when no new block is yet generated. Eventually, $\Delta(b) = 0$ for a large b, since the blocking effectiveness of the attack decreases when more blocks are generated.

Since the attack is performed through multiple payments, the $\Delta(b)$ function is computed taking into account the expiration values of each payment that forms the attack[1]. If we define $\Delta_i(b)$ as the capacity blocked by payment i during b blocks, then $\Delta(b) = \sum_i \Delta_i(b), \forall_i \in attack$. For comparison purposes, we define two single value metrics that compress the $\Delta(b)$ function: Total Blocked Time and Normalized Total Blocked Time.

Metric 1 (Total Blocked Time (TBT)). This metric represents the sum of the $\Delta(b)$ values that defines the TBT metric of the attack:

$$TBT = \sum_{b=0}^{\infty} \Delta(b) \tag{5.3}$$

Metric 2 (Normalized TBT ($\widetilde{TBT}$)). This metric is defined as:

$$\widetilde{TBT} = \frac{TBT}{C_{blocked} \cdot max\{T_{max}\}} \tag{5.4}$$

where $max\{T_{max}\}$ is the maximum default value of T_{max} used in all the experiments. Therefore, $0 < \widetilde{TBT} \leqslant 1$, and the ideal attack with $\widetilde{TBT} = 1$ would be blocking $C_{blocked}$ capacity during 5,000 blocks, that is, more than 34 days.

5.3 Experiment Setup

The goal of the experiments is to evaluate the impact the values of δ and T_{max} have on both the security of the network and its performance, using the metrics

[1]For more details, see [60]

defined in Section 5.2. The experiments consist on a set of simulations where the parameters δ and T_{max} are adjusted for all the nodes in the network. Then, on the one hand, performance is evaluated over the resulting graph (as described in Section 5.2.1) and, on the other hand, a lockdown attack [60] is simulated over the network and the effectiveness and cost of the attack are also evaluated (using the metrics described in Section 5.2.2).

Specifically, each of the experiments is performed in the following way:

1. A LN mainnet graph describing nodes and the existing channels is obtained from a lightning client.

2. Balances are assigned randomly using different probability distributions, and taking into account the capacity of each channel as described in the LN graph.

3. All nodes of the network are configured to simulate their behaviour assuming a certain pair of (δ, T_{max}) values.

4. Evaluation of the performance of the payment network (enforcing the restrictions given by δ and T_{max} values, and taking into account existing channels and their balances).

5. A lockdown attack is simulated over the network.

6. The cost and effectiveness of the attack is evaluated using the AER and normalized TBT metrics.

The following sections describe the LN graph, the balances assignation procedure, and the tested values of the (δ, T_{max}) pair.

5.3.1 LN Payment Channel Graph and Balances

Our simulations will assess the effectiveness of the attack given the actual topology of the network. We base our simulations on the attack algorithm described in [60]. The simulations are made on a snapshot of the LN running on top of the bitcoin mainnet and taken on the 12nd of January, 2020. Both to execute an attack on the network and to evaluate its performance, we need to complement the information of the LN graph with additional data, specifically, the balance of each channel in the network.

The LN does not publicly disclose channels' balances: each user only knows the balances of the channels he participates into them. One alternative to retrieve such balances will be to execute an attack on the network (as described in [130]). However, instead of performing such attack, we have assigned the balances of each channel using different statistical distributions, trying to reproduce the different scenarios that could be found in the network. In order to assign balances to channels, we proceed in the following way: for each channel, first the balance of one of the nodes is randomly selected using one of the selected distributions, and taking the capacity of the channel as the maximum possible value to generate. Then, the balance of the other node in the channel is set as the remaining balance (that is, the capacity minus the balance). Five different distributions are used to assign balances to channels: *deterministic, uniform, normal, exponential,* and *beta*. The *deterministic* distribution always assigns half of the capacity of the channel to each of the nodes; the *normal* distribution is used with $\mu = 0.5$ and $\sigma = 0.2$; the *exponential* distribution uses $\lambda = 1$; the *uniform* distribution has any value within the interval $[0, capacity]$ with the same probability; and the *beta* distribution $\alpha = \beta = 0.25$.

5.3.2 δ and T_{max} Values

We have simulated the network with 16 combinations of δ and T_{max} values. In particular, we have tested all combinations of $T_{max} \in \{432, 1008, 2016, 500\}$ and $\delta \in \{14, 40, 144, 288\}$. The tested values include the ones found in the most popular LN client implementations (see Table 5.1), as well as one additional value for each parameter: a value of 432 (three times 144) for T_{max} and a value of 288 for δ (the double of the maximum default value in any implementation) are also tested. This allows us to test scenarios for which the T_{max}/δ ratio is less than 2, and thus restricts multihop payments.

	lnd (old)	lnd (new)	c-lightning	eclair
T_{max}	5000	5000	2016	1008
δ	144	40	14	144

Tab. 5.1: δ and T_{max} values found in the most popular LN clients [162].

5.4 Experiment results

This section summarizes the results of the experiments. For each of the probability distributions, the same experiment is repeated 10 times, and the averages of the results are presented here.

5.4.1 Performance

Figure 5.1 shows the performance results when using a normal distribution to generate channel balances. Each of the individual heatmaps shows the percentage of random payments for which there are valid multihop paths for a specific amount of satoshis.

To understand these results it is important to note that the diameter of the graph is 7. Moreover, regardless of any restrictions imposed by the configuration parameters of the nodes (δ and T_{max}), only 21% of the payments between any two random nodes on the graph can be executed (due to the structure of the graph itself). For the payments that can indeed be done, the median number of hops is between 3 and 4 (depending on the specific configuration of balances), with an average around 3.75.

Configurations for (δ, T_{max}) with values (288, 432), (288, 1008) and (144, 432) have a T_{max}/δ ratio lower than the graph's diameter. Therefore, their results differ significantly from all the other configurations.

- Regardless of the amount, payments with $(\delta, T_{max}) = (288, 432)$ always fail. This is because this configuration of parameters does not allow any multihop route, and thus payments may succeed only if two randomly selected nodes have a direct channel.

- Configurations (288, 1008) and (144, 432) have a T_{max}/δ ratio of 3.5 and 3, respectively. These values are close to the median of the paths found. This is why there is a performance decay when using these two configurations (with respect to those that have a ratio higher than the graph's diameter). As expected, the percentage of successful payments decreases with the increase of the payment amount, since available balances limit payments.

- All other configurations have similar performance, regardless of the specific (δ, T_{max}) values. Again, the percentage of successful payments decreases with the increase of the payment amount, going from 21% for lower amounts, down to 1.9% for payments of the maximum amount.

For space constraints, we have not included the results for the other four distributions. However, the results are very similar, and the same conclusions can be extrapolated to them.

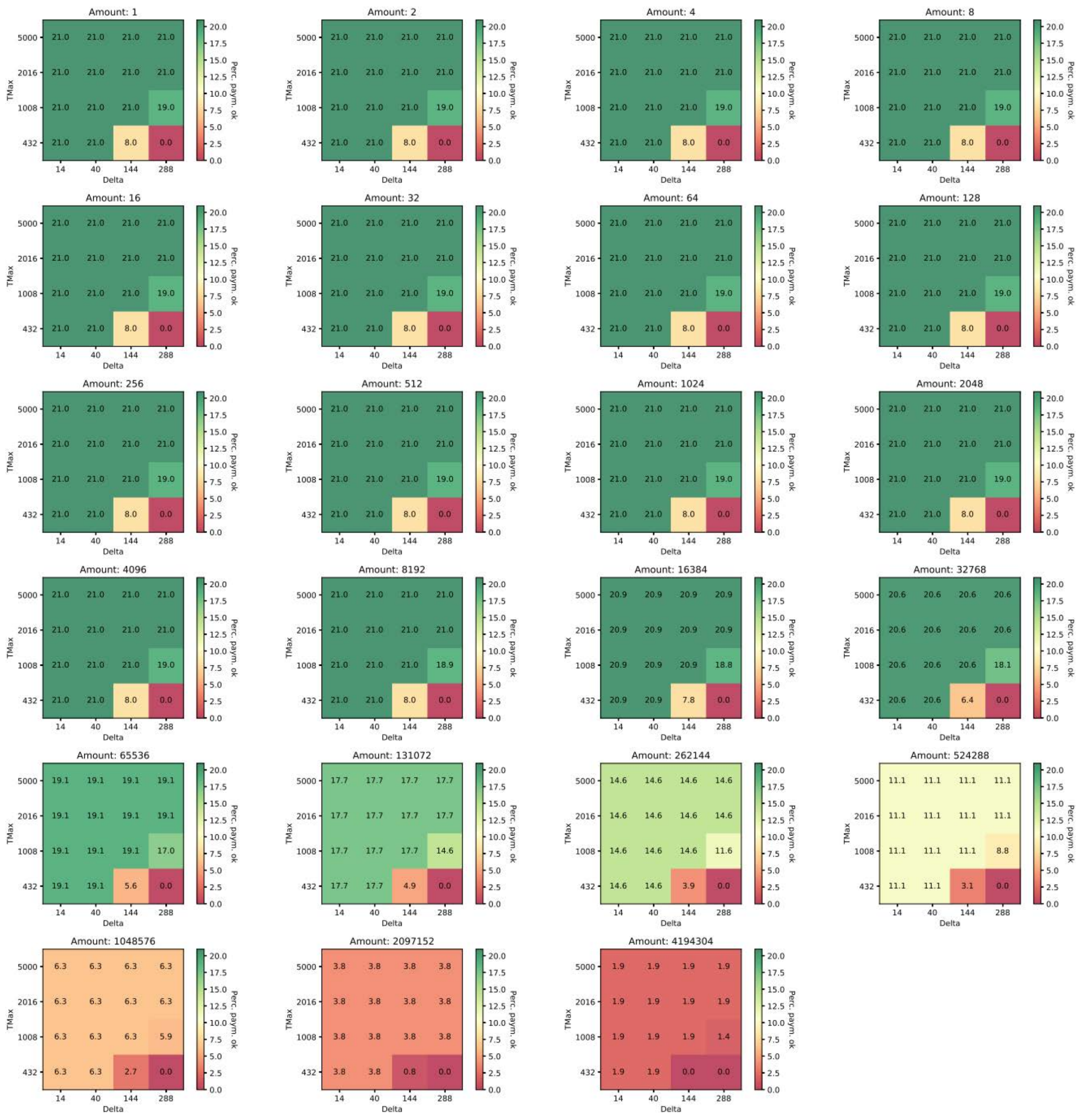


Fig. 5.1: Performance of the payment for different pairs of (δ, T_{max}) [162].

5.4.2 Security

Table 5.2 shows the values of the metrics used to evaluate security (AER and the normalized TBT, $\widetilde{TBT}$) again for instances where balances were assigned using a normal distribution. The general trend that can be observed is that increasing δ and/or decreasing T_{max} makes the attack more difficult for the attacker. Specifically, increasing δ and/or decreasing T_{max} results in:

- higher (or, on some specific configurations, equal) AER values. This implies the attacker needs to be in possession of more bitcoins in order to perform the attack, because more capacity in the channels the attacker creates for the attack is needed. The bitcoins spent in capacity can be recovered once the attack is finished, but the attacker must have them as long as the attack lasts.

- more (or, on some specific configurations, equal) channels needed. This implies a higher economic cost for the attacker which needs to pay more bitcoins in fees to open those channels. The attacker needs to spend these bitcoins (that is, he does not get the bitcoins back once the attack has finished).

- lower (or, on some specific configurations, equal) $\widetilde{TBT}$ values, which means the attacker is able to block the victim during shorter amounts of time.

T_{max}	δ	EAR	*Blocked capacity*	*Channels needed*	$\widetilde{TBT}$
432	14	0.138	95.92%	34.2	0.04
432	40	0.274	95.84%	66.9	0.02
432	144	0.840	75.88%	179.0	0.02
432	288	0.000	0.00%	0.0	0.00
1008	14	0.138	95.92%	34.2	0.15
1008	40	0.138	95.92%	34.2	0.07
1008	144	0.528	95.88%	127.9	0.08
1008	288	0.890	85.34%	201.5	0.07
2016	14	0.138	95.92%	34.2	0.34
2016	40	0.138	95.92%	34.2	0.26
2016	144	0.187	95.98%	46.4	0.06
2016	288	0.528	95.88%	127.9	0.15
5000	14	0.138	95.92%	34.2	0.92
5000	40	0.138	95.92%	34.2	0.83
5000	144	0.138	95.92%	34.2	0.51
5000	288	0.154	95.92%	38.1	0.17

Tab. 5.2: Attack metrics results for different tested parameters with a normal distribution balance [162].

There are a couple of exceptions to the previous tendencies. On the one hand, for the configuration $(\delta, T_{max}) = (288, 432)$ the attacker is not able to block the victim, because no multihop payments can be done with these parameters. Therefore, the attack has no cost for the attacker (since no capacity is blocked). On the other hand, for $(\delta, T_{max}) = (144, 2016)$ the tendency for $\widetilde{TBT}$ deviates from the rest. The reason is that the higher amount of resources spent by the attacker (more channels and capacity) allow for longer blocking times.

5.5 Analysis of LN Contract Parameters Selection

From the perspective of the performance of the payment network, any (δ, T_{max}) configuration except for the three most restrictive ones ((288, 432), (288, 1008) and (144, 432)) offers similar results. Therefore, we should focus on the security properties offered by these similar configurations in order to choose the best pair of values without affecting the performance of the payment network.

Depending on the security metric chosen to evaluate the success of the attack, different pairs of values could be chosen. If the focus is on minimizing the time the attacker is able to lock funds, then (40, 432) and (144, 432) are the best choices, since they both minimize $\widetilde{TBT}$. (144, 432) makes the attack more expensive for the attacker (higher AER and number of channels needed) and less successful (less capacity blocked), however, such configuration results in a poor performance as we show in Figure 5.1. However, attacks with low $\widetilde{TBT}$ are not much of a burdensome for the attacker, since a new attack can be launched again once the funds are released from a previous attack.

Therefore, one may want to hinder the attacker by increasing the economical cost of the attack, both in terms of fees paid to open channels and capacity available in his channels that has to be used to perform the attack. Then, if the focus is on maximizing the cost of the attack, setting $T_{max} = 1008$ and $\delta = 288$ is the best choice: it provides the highest AER and number of channels needed for the attack, while blocking 85% of the victim's capacity and keeping a low $\widetilde{TBT}$ (0.07). Setting $T_{max} = 432$ and $\delta = 144$ is also a good choice, since with a slight decrease in AER, the capacity blocked decreases by 10 percent points and the $\widetilde{TBT}$ decreases to the minimum, 0.02.

5.6 Conclusions

This study aims to offer security while maintaining network usability among LN users by focusing on LN implementations. Specifications guide client configuration by recommending parameter values, which can be customized by the client user. Our study provides an analytical approach to find optimal values for $cltv\_expiry\_delta$ and $locktime\_max$, which enforce the time lock mechanism for commitment transactions and limit the maximum locktime value used in commitment transactions.

The metrics AER and $\widetilde{TBT}$ help us to study network performance and security. The experiments reveal that the parameters currently used in main LN implementations are not optimal regarding performance and security.

Moreover, the results showed that the optimal combination for T_{max} and δ values is 432 and 40, respectively. These values represent a worst-case scenario for attackers but preserve the same payment success rate. Modifying these parameter values impacts network performance and security by reducing HTLC timeouts and preventing lockdown attacks that lock victim funds on the network.

6 Node centrality

BitCoin is actually an exploit against network complexity. Not financial networks, or computer networks, or social networks. Networks themselves.

— **Dan Kaminsky**
(Internet security researcher)

By selecting the most suitable values for LN parameters δ and T_{max}, we evaluate how their tuning as contract parameters affects the performance and security of LN. However, this analysis did not give us a broad view of its topology, which is equally crucial to understanding network performance, privacy, and security. Besides the effect that LN topology has on being able to route payments through nodes successfully, it also affects the privacy of payments and their resilience towards attacks and random failures.

Since LN is a PCN that routes payments made by nodes through Bitcoin, a suitable question is to what extent the untrusted and decentralized blockchain model extends to the LN network. A set of studies [146, 150, 148] attempt to analyze the network through conventional graph theory centrality metrics. However, these studies fail to capture the network's semantics since they neglect to consider LN properties or payment restrictions. In this chapter, we present an extended model for the LN that goes beyond channel capacity to consider a variety of LN properties. We provide a more comprehensive assessment of node centrality in the LN by expanding the analysis to include alternative metrics. We overcome the shortcomings of conventional metrics and improve our knowledge of the relevance of nodes in the network. Our approach goes from the perspective of the theoretical to the practical by considering several centrality metrics whose analysis can be extrapolated to any PCN.

Accordingly, we extend the LN topology analysis of previous studies [131, 141, 149, 120] that consider centrality metrics such as degree, betweenness, and closeness with channel capacity as weights in an undirected network. In our analysis, we include LN properties such as channel parameters (fee, capacity, and balance), contracts (pending HTLCs), and additional metrics (Opsahl, strength, and current flow betweenness). As a result, we can offer an in-depth analysis of which metrics are best suited to define LN network centrality. Therefore, it is the basic properties of LN on which we apply our theoretical approach, which is novel because it delves into the semantics of the specific properties of a PCN. In the case of LN, these properties are the capacity and balance of a channel and the fee related to payment processing.

The following sections present a broad model for LN that includes not only the channel parameters as fee, capacity, and balance but also other policy parameters, all of which represent LN instances. Based on this model, we discuss the semantics of evaluating the LN using traditional centrality metrics and describe how to evaluate LN properties using them. Furthermore, we highlight the limitations of using classical centrality metrics to assess LN and provide insights on how to use them effectively. Thus, we provide alternative metrics that better suit the evaluation of LN centrality. Additionally, we provide empirical analysis of the LN centrality over a two-year period, which can help better understand the network's behavior and performance.

6.1 A model for the Bitcoin LN

Several literature papers (see Section 4.3 for more details) model Bitcoin LN via graph theory tools. However, all the proposals reviewed are based mainly on very general information about the channels, such as their mere existence between two users and their capacity. But, they rule out other more subtle information that greatly affects the flow of money between users, such as: how both parts of the channel divide the capacity of a channel (i.e., their balance in the channel), or how the routing and HTLC apply the fees.

In this section, we propose a finer model to represent a snapshot[1] of the LN. We model the LN as two graphs G_1 and G_2, together with two functions, f_V and f_E, that map elements of one graph with elements of the other graph. G_1 is an undirected graph and G_2 is a directed graph. The rationale behind this decision is that there are properties of the channels that are better modeled with an undirected graph, but other properties are better represented with a directed one. Having thus two graphs, allows us to create a rich representation of the network. Moreover, this double representation allows us to apply graph-theoretic metrics to measure the nodes of the network in a significant way. Next, we describe the details of the two graphs that represent the LN using the proposed model, details summarized in Table 6.1.

[1]We define a snapshot of the LN as the status of the payment channels that conform the network at a given instant of time.

Graph	$\mathbf{G_1}$	$\mathbf{G_2}$
Notation	$G_1 = (V_1, E_1)$ $\overline{e}_{ij} = \overline{e}_{ji} = (\overline{v}_i, \overline{v}_j) = (\overline{v}_j, \overline{v}_i) \in E_1$	$G_2 = (V_2, E_2)$ $e_{ij} = (v_i, v_j) \in E_2$
Type	Undirected multigraph	Directed multigraph
Nodes denote	Public keys	Public keys
Edge identifier	Funding transaction outpoint	Source node public key Funding transaction outpoint
Edge properties	Capacity (C_{ij})	Available (b_{ij}) and Blocked (h_{ij}) balance Minimum HTLC amount (m_{ij}) Fee per byte (f_{ij})
Edges denote	Open channels	Current state of open channels
Data from	Bitcoin blockchain	Off-chain transactions P2P messages

Tab. 6.1: An overview of the model's definition [163].

Let $\mathbf{G_1} = (\mathbf{V_1}, \mathbf{E_1})$ be an **undirected** graph, that contains static channel information that can be extracted from the blockchain.

Nodes $\overline{v}_i \in V_1$ represent users of the LN identified by their public keys. Edges $\overline{e}_{ij} \in E_1$ represent open channels between those users. The outpoint (transaction identifier and output index) that funds the channel uniquely identifies each edge.

The graph G_1 may be a multigraph because many different channels can be opened between a pair of nodes. Moreover, the graph is indeed undirected, because the outpoint that defines is a 2-out-of-2 multisig output, where none of the two public keys has any advantage nor privileged position, and thus both participants have the same role in the relationship.

Edges encode channel information that can be extracted from the funding transaction, e.g., the capacity of the channel. We use the double subindex notation in an edge $\overline{e}_{ij}$ to indicate the index of each incident vertex of the edge, that is, $\overline{e}_{ij} = (\overline{v}_i, \overline{v}_j)$. Furthermore, since G_1 is an undirected graph, $\overline{e}_{ij} = \overline{e}_{ji} = (\overline{v}_i, \overline{v}_j) = (\overline{v}_j, \overline{v}_i)$. Note that, since G_1 is a multigraph, we add an index to identify the multiple edges of the same two nodes, $\overline{e}_{(ij)_k} = \overline{e}_{ijk}$. For simplicity, we omit this third index from the notation whenever it is not specifically needed. We denote by c_{ij} the capacity of edge $\overline{e}_{ij}$, with $c(\overline{e}_{ij}) = c_{ij}$.

Let $\mathbf{G_2} = (\mathbf{V_2}, \mathbf{E_2})$ be a **directed** graph, that contains dynamic channel information that is reflected in off-chain commitment transactions and P2P LN messages exchanged between nodes.

The set of nodes $v_i \in V_2$ represents public keys and is the same set of nodes of G_1, that is $V_2 = V_1$. The set of edges $e_{ij} \in E_2$ represents the current state of the channels between those public keys. Each edge is uniquely identified by the outpoint (transaction identifier and output index) that funds the channel and the source node they refer to. The graph G_2 may be a multidigraph because many different channels can be opened between a pair of nodes[2].

Edges in E_2 encode more detailed information about the channel than data stored in the edges of E_1. The extraction of such information does not come from on-chain transactions but the commitment transactions exchanged from the LN nodes and also from the LN P2P messages that the nodes broadcast. Regarding commitment transactions, we can classify their outputs[3] in two types, depending on which of the two parties is the receiver. Therefore, we model each pair of commitment transactions as two directed edges: the edge $e_{ij} = (v_i, v_j)$ from v_i to v_j will encode v_i's balance in the channel and offered HTLCs and, reciprocally, the edge $e_{ji} = (v_j, v_i)$ from v_j to v_i will encode v_j's balance in the channel and offered HTLCs. We denote by b_{ij} the balance of edge e_{ij}, $b(e_{ij}) = b_{ij}$; and by h_{ij} the balance blocked in HTLC of edge e_{ij}, $h(e_{ij}) = h_{ij}$.

Furthermore, edges in E_2 also encode additional information extracted from channel policies sent within the LN P2P network, such as the fee that is charged to use the channel, f_{ij}, measured in satoshis per byte, and the minimum amount of satoshis that can be routed through that channel, m_{ij}.

Finally, we also define two functions, $\mathbf{f_V}$ and $\mathbf{f_E}$, for mapping nodes and edges between G_1 and G_2. Let $\mathbf{f_V} : \mathbf{V_2} \rightarrow \mathbf{V_1}$ be a bijective function that maps nodes of the graph G_2 with nodes of the graph G_1. Let $\mathbf{f_E} : \mathbf{E_2} \rightarrow \mathbf{E_1}$ be a noninjective surjective function that maps the edges of graph G_2 with the edges of graph G_1.

Note that, regarding the defined model for the LN presented so far, the following restrictions must be preserved:

1. The set of nodes of both graphs is the same, that is, $V_1 = V_2$. So $\mathbf{f}_V$ is the identity function.

2. Each element $e \in E_2$ is mapped to exactly one element in E_1 (derived from the function definition).

[2]Again, since G_2 is a multidigraph, we add an index to identify multiple edges of the same two nodes, $e_{(ij)_k} = e_{ijk}$. For simplicity, again, we omit this third index from the notation whenever it is not specifically needed.

[3]Commitment transactions have four types of outputs which are: local outputs, remote outputs, received HTLC, and offered HTLC.

That is, for each $e_{ij} = (v_i, v_j) \in E_2$ with $f_V(v_i) = \overline{v}_i$ and $f_V(v_j) = \overline{v}_j$, there exists one edge $\overline{e}_{ij} = (\overline{v}_i, \overline{v}_j) \in E_1$.

3. Each element $\overline{e} \in E_1$ is the image of exactly two elements in E_2.

   That is, for each $\overline{e}_{ij} = (\overline{v}_i, \overline{v}_j) \in E_1$ with $f_V(\overline{v}_i) = v_i$ and $f_V(\overline{v}_j) = v_j$, there exists exactly two edges in E_2, e_{ij} and e_{ji}. Therefore, $|E_2| = 2 \cdot |E_1|$.

4. Let $e_{ij} = (v_i, v_j)$ and $e_{ji} = (v_j, v_i)$ be the edges of E_2, the balances and pending HTLC values must be consistent with the total capacity channel, so it must hold that:

$$b_{ij} + h_{ij} + b_{ji} + h_{ji} = c_{ij} = c_{ji}. \tag{6.1}$$

To sum up, Figure 6.1 shows a toy example of an LN snapshot with 3 nodes and 2 channels using the proposed model.

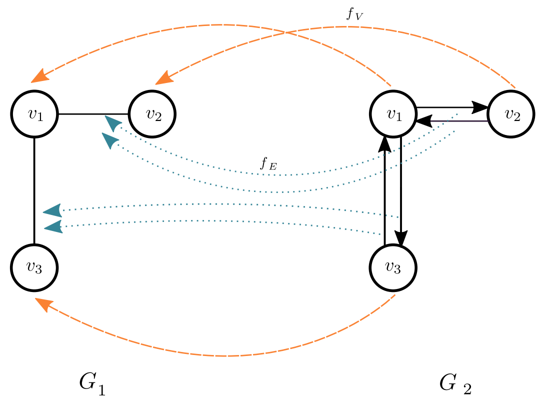


Fig. 6.1: The LN data model

6.2 A discussion on classic centrality metrics applied to LN nodes

In this section, we review different classical centrality measures proposed in the field of graph theory. As well, we analyze to what extent they preserve the centrality meaning when they are computed over a graph that models a payment network, like the LN.

6.2.1 Symmetric graphs

In his seminal paper laying the foundations of centrality metrics in social networks [164], Freeman used the star graph as a starting point to guide his exposition. In a star graph (Figure 6.2), intuition leaves no doubt as to which node is more central. Furthermore, this node is not only the center point of the star graph, but also the most central position imaginable on any graph of a similar size order. But why is this node central? It has three structural properties: it has the highest degree (i.e., the most number of neighbors), it is in the shortest paths between other nodes, and its distance to other nodes is minimal. These three properties are the basis of the three most basic centrality metrics for nodes in networks: degree, betweenness, and closeness centralities.

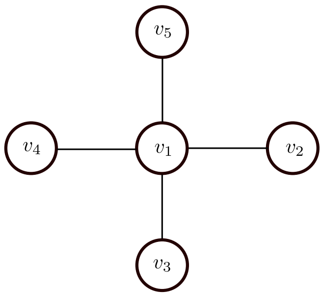


Fig. 6.2: A basic star graph

Given a graph of n nodes with adjacency matrix $A_{n\times n} = [a_{ij}]$, where a_{ij} is a binary value denoting whether there exists an edge between nodes v_i and v_j, degree centrality is defined as the number of neighbors of a node:

$$C_D(v_i) = \deg(v_i) = \sum_{j=1}^{n} a_{ij}. \tag{6.2}$$

The shortest path between two nodes is a path of the shortest length. Let σ_{st} be the number of the shortest paths between s and t; and $\sigma_{st}(v)$ the number of those paths that pass through v. Then, betweenness centrality is defined as the fraction of the shortest paths between all pairs of nodes of the graph that pass through v:

$$C_B(v) = \sum_{s\neq v\neq t\in V} \frac{\sigma_{st}(v)}{\sigma_{st}}. \tag{6.3}$$

The distance d between two nodes in a graph is the length of the shortest path between them. Closeness centrality is defined as the inverse of the sum of distances between one node and all the other nodes of the graph:

$$C_C(v_i) = \frac{1}{\sum_{j\in[1,n],j\neq i} d(v_i, v_j)}. \tag{6.4}$$

But to what extent are these centrality metrics relevant to evaluate nodes in the LN? Indeed, a node with a high **degree** is a node with lots of channels, which provides it with robustness (since it does not rely on a single or a few channels to be able to operate in the network). Moreover, a high degree also implies direct channels with more other nodes in the network and thus independence. On the other hand, a node with high **betweenness** is a node that is in the middle of payments between other nodes, in case the shortest path is used to choose payment routes. This allows it to have some degree of control and information about those payments (e.g., it knows the amount, HTLC values to estimate the overall number of hops, can decline participation, can delay payments), and also to obtain revenue from them in the form of fees. Finally, a node with high **closeness** may benefit from making payments with fewer hops, which may have consequences on both the fees to pay and the privacy of its payments.

6.2.2 Symmetric weighted graphs (capacity)

However, these three basic metrics assume all channels are equally important (have the same contribution) to the importance of the node. Nonetheless, this is hardly the case: lightning channels have a capacity, that limits the amount of bitcoins a payment can move through them. For instance, take as an example the weighted double star graph shown in Figure 6.3. To create it, one could just add v_6 to the simple star graph (Figure 6.2) and connect it to v_2, v_3, v_4 and v_5; and where channels' capacity is represented as edge weights. Now, one could argue that node v_6 is more central than node v_1, since, although they both have exactly four channels and are in the same structural position on the graph, node v_6 can make payments of a higher amount in all of its channels.

Degree, betweenness, and closeness centralities have also been defined to take into consideration edge weights. Newman [165] and Barrat *et al.* [166] extends

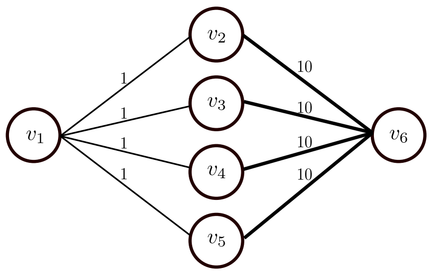


Fig. 6.3: A weighted double star graph

degree centrality to consider weights, where the **strength of a node** is defined as the sum of the weights of its connections (its incident edges):

$$C_D^w(v_i) = s(v_i) = \sum_{j=1}^{n} a_{ij} w_{ij}. \tag{6.5}$$

where w_{ij} is the weight of the edge between nodes v_i and v_j.

Brandes [167] and Newman [165] generalize the centralities of betweenness and closeness for weighted graphs using the sum of the weights of the edges of a path to define its length. Therefore, the shortest path between two nodes is not the path using the least number of hops, but the one that has the least sum of weights, and distance between nodes is defined in the same terms (i.e., the sum of weights of the edges in the shortest paths between them):

$$C_B^w(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}^w(v)}{\sigma_{st}^w}, \tag{6.6}$$

$$C_C^w(v) = \frac{1}{\sum_{u \neq v \in V} d^w(v,u)} \tag{6.7}$$

where d^w and σ^w are distance and number of the shortest paths taking into account the sum of weights as the length of paths.

Nodes	C_D	C_B	C_C	C_D^w	C_B^w	C_C^w	$C_D^{1/w}$	$C_B^{1/w}$	$C_C^{1/w}$
v_1	4	3	0.83	4	6	0.33	4	0	0.98
$v_{2..5}$	2	0.25	0.625	11	0.25	0.29	1.1	0.25	2.94
v_6	4	3	0.83	40	0	0.09	0.4	6	3.3

Tab. 6.2: Centrality measures for a double-star network with weights [163] (Figure 6.3).

Getting back to our example (Figure 6.3), node v_6 has now a higher weighted degree centrality (C_D^w) than node v_1 (40 and 4, respectively). Yet a problem arises when applying weighted betweenness (C_B^w) and closeness (C_C^w) centralities to evaluate the centrality of LN nodes. In their standard formulation, weight is interpreted as the cost of using that edge, whereas capacity is not the cost but the maximum amount that can be transacted through the channel. As a consequence, both betweenness and closeness centralities are higher for v_1 than for v_6, since all shortest paths between other nodes always pass through v_1, and v_1 has a lower distance to all other nodes in the graph. To circumvent this problem, some authors have used the multiplicative inverse of capacity as edges' weight [146] when computing betweenness and closeness centralities. With this definition, v_6 has now more weighted betweenness centrality than v_1 (6 and 0, respectively); and also higher weighted closeness (3.3 and 0.98, respectively), as shown in Table 6.2.

Again, it is important to understand what these metrics evaluate concerning LN nodes. Nodes with high **weighted degree** centrality are nodes that have a lot of capacity to operate within the network: they can potentially transact a higher amount. However, in contrast with unweighted high degree nodes, they will not always have strong robustness or independence, since C_D^w does not capture how this weight is distributed (i.e., it can be concentrated in a single channel). Section 6.2.2 explains how can we incorporate both the strength and the degree into the evaluation.

Note, also, that a node with high **weighted betweenness** centrality is a node that is in the middle of payments between other nodes that choose the shortest paths with higher capacities as payment routes. We argue that this is a very artificial use of the metric, that does not capture how payment networks operate. On the one hand, the restriction that only the path with the highest capacity (i.e., the lowest cost using the inverse of the capacity as weight) is going to be used does not make much sense in a payment network: any channel that has enough balance is valid, and the best path will be chosen based on other considerations such as fees. Sections 6.2.4 and II will explain how to deal with this. On the other hand, this does not take into account other restrictions in the routes, covered in Section 6.3.

Analogously, **weighted closeness** is again not very useful since it does not make sense to consider capacity as a cost or distance between nodes. Sections 6.2.4 will explain another approach that can better capture nodes' closeness.

Weight and strength - Node strength as defined in the previous section only takes into account the total engagement of the node, yet obliterates how is this involvement distributed across different connections. Therefore, although node strength is presented as a generalization of node degree for weighted networks, it fails to capture the original meaning of degree. Opsahl[168] *et al.* proposed a different formulation to combine both degree and strength:

$$C_D^{w\beta}(v_i) = deg(v_i)^{(1-\beta)} \cdot s(v_i)^{\beta} = C_D(v_i)^{(1-\beta)} \cdot C_D^{w}(v_i)^{\beta}. \tag{6.8}$$

This formulation depends on the parameter β to tune the contribution of the number of connections and the strength of the node into the centrality score: if β is 0, $C_D^{w\beta}$ is equal to the node degree; if β is 1, $C_D^{w\beta}$ is the node strength as defined by Newman; values of β between 0 and 1 provide higher $C_D^{w\beta}$ for nodes with a high degree, whereas values of $\beta > 1$ provide higher $C_D^{w\beta}$ for nodes with a lower degree.

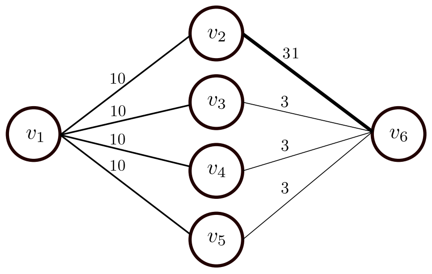


Fig. 6.4: Example of two identically weighted and strength nodes.

However, when evaluating the robustness of a node or ability to make payments in the network, $C_D^{w\beta}$ still falls short. For instance, if we take a graph like the one shown in Figure 6.4, nodes v_1 and v_6 have the same degree and strength. Consequently, regardless of the β value chosen, $C_D^{w\beta}$ is always the same for both nodes. However, intuitively node v_1 is better connected to the network, because of how its strength is distributed across its connections. An attack (or failure) of any of his channels would just affect 1/4 of its capacity. On the contrary, a directed attack over the v_2v_6 channel will strongly affect v_6, making him lose most of its capacity to operate with the network. A variant of the Opsahl metric can take this into account:

$$C_D^{w\alpha}(v_i) = \sum_{j=1}^{n} a_{ij} w_{ij}^{\alpha}. \tag{6.9}$$

This measure is indeed able to capture the differences between v_1 and v_6 (Table 6.3).

Again, if α is 0, $C_D^{w\alpha}$ is equal to the node degree; if α is 1, $C_D^{w\alpha}$ is the node strength as defined by Newman. For nodes with the same strength and degree, values of α between 0 and 1 provide higher $C_D^{w\alpha}$ for nodes with strength equally divided between channels, whereas values of $\alpha > 1$ provide higher $C_D^{w\alpha}$ for nodes

Nodes	C_D	s	$C_D^{w\beta}$						$C_D^{w\alpha}$					
			$\beta = 0$	$\beta = 0.25$	$\beta = 0.9$	$\beta = 0.95$	$\beta = 1$	$\beta = 1.25$	$\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.9$	$\alpha = 0.95$	$\alpha = 1$	$\alpha = 1.25$
v_1	4	40	4	7.11	31.77	35.65	40	71.13	4	7.11	31.77	35.65	40	71.13
v_2	2	41	2	4.26	30.31	35.25	41	87.24	2	4.14	29.93	35.02	41	90.93
$v_{3..5}$	2	13	2	3.19	10.78	11.84	13	20.76	2	3.09	10.63	11.75	13	21.73
v_6	4	40	4	7.11	31.77	35.65	40	71.13	4	6.31	30.05	34.63	40	84.99

Tab. 6.3: Centrality metrics for graph of two identically weighted and strength nodes[163] (Figure 6.4).

with strength concentrated in the same (or a small number of) channels (cf. v_1 and v_6).

6.2.3 Directed weighted graphs (balance)

All the metrics presented so far are computed over the capacity of the nodes' channels, and thus provide information about the possible payments the node may be involved with. Taking into account the model presented in the previous section, all of them can be computed over graph G_1. However, they fail to capture another important detail of payment networks, the current balances of nodes in the channel. That is, at a certain instant of time, these measures do not take into account how is the capacity of the channel distributed between the two ends of the channel to evaluate its centrality. If we consider, for instance, the network from Figure 6.4, the channel between v_1 and v_2 has a capacity of 10. However, the ability of both nodes to operate within the network will not be limited by this capacity, but by the balance that each of them has at that moment. If all the capacity is on v_2's side, v_1 will not be able to make payments (or route outgoing payments) through that channel, and thus its strength will be reduced from 40 to 30 (assuming he has all the possible balance in the other three channels).

Channel balances may be represented with a direct graph, where the weight of the edges represents the balance the source node has in a channel, which corresponds to graph G_2 of our model presented in the previous section. The sum of the two edges (one in each direction) that represent a channel is thus at most the capacity of that channel. Using this representation, we can use the directed versions of the metrics presented above to evaluate the centrality of a node.

Figure 6.5 represents a possible distribution of balances for the network shown in Figure 6.3. The edges with a balance of zero have not been drawn for readability. Most channels are completely unbalanced, with all the capacity available in just one direction. The exception is the channel between nodes v_2 and v_6, whose capacity is split equally between both nodes.

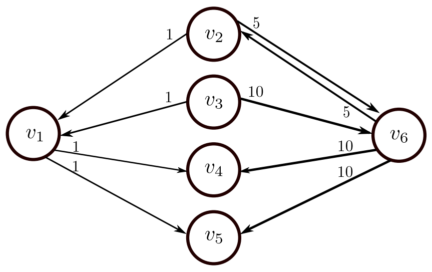


Fig. 6.5: A directed double star graph.

Degree-based centrality metrics over directed graphs distinguish between outgoing and incoming edges. For instance, indegree and outdegree (C_{D^-} and C_{D^+}, respectively) take into account the number of incoming or outgoing edges, respectively; and strength is also computed separately for incoming and outgoing edges. Metrics based on paths consider only those paths that are valid considering the direction of the edges.

Table 6.4 summarizes the centrality metrics for the directed graph example shown in Figure 6.5.

Nodes	C_{D^+}	C_{D^-}	C_B	C_C	$C^w_{D^+}$	$C^w_{D^-}$	C^w_B	C^w_C
v_1	2	2	2	0.75	2	2	6	0.37
v_2	1	2	1	0.67	5	6	3	0.1
v_3	0	2	0	0	0	11	0	0
$v_{4..5}$	2	0	0	0.67	11	0	0	0.33
v_6	2	3	3	1	15	25	1	0.13

Tab. 6.4: Metrics of Centrality for a directed network with weights [163] (Figure 6.5).

6.2.4 Symmetric weighted graphs (fee)

As we have seen in Section 6.2.2, weighted betweenness and closeness centralities using capacity as weight are not able to capture the importance of a node, because these metrics are based on shortest paths and distances taking into account channel capacity as a cost.

Instead, using channel fees as weight is more representative of what rational nodes may implement since fees are indeed a cost of using the channel. Therefore, nodes with a high weighted betweenness centrality with fees as the weight will be in the middle of payments between other nodes that try to optimize the cost of their payments by choosing the cheapest routes. Moreover, nodes with a high weighted closeness centrality with fees as weigh will be nodes that can make payments with the lowest fees.

However, note that this approach, using a simple symmetric graph constructed by channels, has also a problem: channels can only be used if they have enough capacity and balance in the desired direction. Furthermore, dealing with payments in the LN, additional restrictions also apply, as we will explain in Section 6.3.

6.2.5 Flow based centrality metrics

One of the problems of using betweenness centrality as defined in the previous section is that it is based on shortest paths. Even when considering weight, nodes that may offer connectivity to the network, but that are not found in the middle of these shortest paths are not considered to have any influence.

Let's consider again nodes v_1 and v_6 from 6.3 and note that betweenness centrality ($C_B^{1/w}$) is 0 for v_1 and 6 for v_6 (Table 6.2). These values may seem to indicate that node v_1 will never be in the middle of payments between other nodes. However, in a payment channel network, this may not be the case: with the information, we currently have in the graph, payments of less than or equal to 1 would have no reason to prefer to be routed through node v_6 over node v_1. Flow-based centrality metrics allow overcoming this limitation.

I Flow networks

Flow networks are used to model different problems, from pipes moving water to electrical or information networks.

A **flow network** is a directed graph that has a nonnegative capacity in each edge ($c : V \times V \to \mathcal{R}_{\geqslant 0}$). Nonexistent edges are assumed to have a capacity of 0. A flow network has two special nodes: a source s and a sink t.

A **flow** is defined in a flow network as a function that assigns a real number f to each pair of nodes ($f : V \times V \to \mathcal{R}_{\geqslant 0}$), such that:

1. for all $u, v \in V$, $0 \leqslant f(u, v) \leqslant c(u, v)$, and
2. for all $u \in V - \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

That is, at each edge the flow f must be lower or equal than the capacity c (capacity constraint), and the flow must be preserved at each node except for the source and the sink (flow conservation constraint).

The **value** $|f|$ of a flow is:

$$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s). \tag{6.10}$$

Given a flow network, a source, and a sink, the **maximum-flow** problem consists in finding flow f of maximum value $|f|$. Let f'_{st} be a flow of maximum value between nodes s and t. Let $f'_{st}(v)$ be a flow of maximum value between nodes s and t passing through node v.

II Betweenness centrality based on flow

Freeman [169] extended the betweenness centrality metric based on flow, where a node is more central to the extent where more flow between pairs of other nodes in the graph depends on it. That is, to define the flow centrality of a node v is through the amount of flow between any pair of nodes in the graph that needs to pass through v divided by the sum of the maximum flow values of any pair of nodes in the graph:

$$C_F(v) = \frac{\sum_{s \neq v \neq t \in V} |f'|_{st}(v)}{\sum_{s \neq v \neq t \in V} |f'|_{st}}. \tag{6.11}$$

Table 6.5 shows the flow centrality measure for the example graph from Figure 6.3. In contrast with traditional betweenness centrality measures based on weight and its inverse (C_B^w and $C_B^{1/w}$, respectively), flow based betweenness centrality (C_F) is able to capture the ability of nodes to be in the middle of payment paths. Node v_6 is more central than v_1, to the extent that payments between other pairs of nodes will be able to be of a higher amount than payments through v_1. However, node v_1 may still be in the middle of payments between other nodes, given that it has a tenth of the C_F of v_6. In contrast to $C_B^{1/w}$, where the node is not considered to be on the shortest paths and is therefore assigned a centrality of 0).

Nodes	C_B^w	$C_B^{1/w}$	C_F
v_1	6.00	0.00	0.090909
$v_{2..5}$	0.25	0.25	0.048780
v_6	0.00	6.00	0.878049

Tab. 6.5: Weighted double-star graph's centrality based on Flow [163] (Figure 6.3).

Flow based betweenness centrality allows to measure the importance of a node in a payment network to the extent it is in the middle of payments between other pairs of nodes (and therefore collect metadata about those payments and potentially profit from them).

III Betweenness centrality based on current flow

Brandes [170] proposes a centrality metric based on variations of betweenness and closeness, but with a different model in which information spreads efficiently similar to electrical current. The proposed metric overcomes the limitations related to execution times and space requirements that arise with computing large networks. Regardless of the approaches taken in flow betweenness, about including nongeodetic paths in a node's total score and measuring the amount of flow that passes through a node, its paths must be optimized to achieve their maximum value, and thus, solve real situations in which information moves randomly.

A similar approach taken by Newman [171] measures betweenness centrality based on random walks, same as current-flow betweenness (C_{CF}) which measures the portion of current flow that passes through a node v between all possible node pairs in the network. The C_{CF} of a node v can be defined as the amount of current that flows through a node averaged over all node pairs s and t. To be more specific, C_{CF} of a node v is the average of the current flow over all source-target pairs:

$$C_{CF}(v) = \frac{\sum_{s \neq t \in V} {I_V}^{(st)}}{\frac{1}{2}n(n-1)}. \tag{6.12}$$

where ${I_V}^{(st)}$ represents the current flow through a node v between source s and target t and $\frac{1}{2}n(n-1)$ is a normalizing constant.

Table 6.6 shows the current flow centrality measure for the example graph of Figure 6.3. We compare again the traditional betweenness measures based on weight and its inverse, as well as, the flow betweenness. On the contrary, to the previous metrics, current flow based betweenness centrality (C_{CF}) captures the flow that passes through a node in the middle of payment paths. Similar to flow betweenness, node v_6 is more central than v_1, therefore the flow of payments between other pairs of nodes will be greater than payments through v_1. However, nodes v_2 to v_5 are even more central than v_1 because those nodes can process more flow than node v_1.

Flow-based Metric Selection - Luo [172] states that C_F and C_{CF} share a similar behavior for measuring the frequency of a node v among a couple of nodes s and

Nodes	C_B^w	$C_B^{1/w}$	C_F	C_{CF}
v_1	6.00	0.00	0.090909	0.818181
$v_{2..5}$	0.25	0.25	0.048780	1.000000
v_6	0.00	6.00	0.878049	8.181818

Tab. 6.6: Weighted double-star graph's centrality based on Current Flow [163] (Figure 6.3).

t. However, these metrics differ in that C_F bases its calculation by comparing the maximum possible paths containing node v. Furthermore, this metric can be described as the proportion of the volume of flow that passes through v when the flow reaches the maximum value [173]. However, this metric might ignore paths that are central in the network when they are not crossed by any unit of flow for pairs of nodes s and t [174]. On the other hand, C_{CF} measures the frequency of a node v in a random-walk, a name that is also given to this metric [175, 170], between nodes s and t when calculating all paths existing between those nodes. Unlike C_{CF}, C_F is not a realistic metric since it only considers a small subset of possible paths between nodes.

At the moment when we consider the complexity of both metrics, C_F can be calculated in time $\mathcal{O}(m^2n)$, instead, the complexity of C_{CF} is $\mathcal{O}((m+n)n^2)$ using matrix methods. This comparison indicates that the computation demand of C_{CF} is comparable to C_F. Therefore, based on the aforementioned, we select current-flow betweenness as a metric to obtain the frequency of a node that occurs on a path.

6.3 Connectivity in the scope of a payment network

The edges between nodes and the paths that form those edges define connectivity in classical graph theory. A path in a graph is a sequence of incident edges such that neither vertices nor edges are repeated. Although not introduced explicitly, the paths are the basis of some of the centrality measures we have reviewed in the previous section, such as betweenness and closeness centrality.

However, such a basic definition of a path may not be suitable in the scope of a payment network like the LN, since not all paths defined in this simple manner are valid payment routes in the modeled payment network. There exist some additional restrictions for a path to be a valid payment route.

Therefore, to provide more accurate centrality measures for the modeled LN, we redefine the concept of a path. We define a **payment path** for an amount σ as a path with the following restrictions:

1. There is enough balance in all the channels to fulfill the payment.

2. The length of the path is smaller or equal to 20.

3. The number of existing HTLCs in each channel is less than 14.

4. The policies of the nodes in the path are compatible.

   a) There exists a set of timeouts for all HTLCs in the path that fulfill the conditions on the nodes' policies for all nodes in the path.

   b) The amount of payment is higher than the minimum ($\sigma > min\_htlc$).

Note that the first restriction is a general restriction of any payment network while the other ones are more specific to the current LN implementation and are extracted from its specifications. For restriction 2, we refer to BOLT 4 Section Packet-Structure. Instead, restrictions 3 and 4b refer to BOLT 2 Sections Adding an HTLC-Rationale and The *open_channel* Message respectively. Finally, for restriction 4a, it refers to BOLTs 2 and 7 Sections *cltv_expiry_delta* Selection and Recommendations for Routing respectively.

6.4 Proposed centrality measures in the scope of LN

Once reviewed all possible centrality measures that can be directly computed on the graph that model the LN, either the symmetric one G_1 or the directed one G_2[4], we now propose and justify which of them are suitable to measure the centrality of the nodes of the network and which is the property that such centrality measure provides, in terms of robustness/resilience or surveillance/control.

Table 6.7 summarizes all the metrics considered in the measures presented in Section 6.5. The first column of the table identifies the measure and the second one provides the exact formula used to compute such metric. The third column provides information about the graph over which the metric is computed. Note that some metrics may have a different meaning if computed on a symmetric (G_1) or a directed (G_2) graph. The Weight column indicates which parameter of the LN is selected as the weight for the calculation (in the case of a weighted metric).

The Restrictions column indicates which restrictions have been considered when applying some specific measures. Note that two clear sets appear when dealing with restrictions: measures based on direct connectivity (degree) and measures based

[4]See Section 6.1 for the defined model

on indirect connectivity (path). The restriction that we have taken into account for direct connectivity is the existence of the channel (so the corresponding edge in the graph) and whether or not such a channel is enabled in the policy information that the node broadcasts. Regarding measures using path connectivity, we have applied the concept of payment path defined in Section 6.3 with the restrictions indicated. Finally, the last column of the table provides a brief description of the meaning of each measure.

Centrality metric	Formulation	Graph	Weight	Restriction Type	LN property to evaluate
Degree	$C_D(v_i) = \deg(v_i) = \sum_{j=1}^{n} a_{ij}$	G_1	N/A	Channels enabled	Number of channels.
Strength	$C_D^w(v_i) = s(v_i) = \sum_{j=1}^{n} a_{ij} w_{ij}$	G_1	c	Channels enabled	The probable channel capacity available to the node for LN transactions.
Opsahl	$C_D^{w\alpha}(v_i) = \sum_{j=1}^{n} a_{ij} w_{ij}^{\alpha}$	G_1	c	Channels enabled	The node's potential for resilience: The highest amount that a node is permitted to transact, as well as how this amount is divided among other nodes.
Incoming strength	$C_{D^-}^w(v_i) = \sum_{j=1}^{n} a_{ji} w_{ji}$	G_2	b	Channels enabled	The amount of channel balance available for LN payments. A node's maximum amount that it may send.
Outgoing strength	$C_{D^+}^w(v_i) = \sum_{j=1}^{n} a_{ij} w_{ij}$	G_2	b	Channels enabled	The channel balance in the LN that is used to receive payments. A node's maximum amount that it may receive.
Incoming Opsahl	$C_{D^-}^{w\alpha}(v_i) = \sum_{j=1}^{n} a_{ij} w_{ij}^{\alpha}$	G_2	b	Channels enabled	The node's resilience in terms of payment processing (the maximum payment amount that a node can make while considering its neighbor distribution).
Outgoing Opsahl	$C_{D^+}^{w\alpha}(v_i) = \sum_{j=1}^{n} a_{ji} w_{ji}^{\alpha}$	G_2	b	Channels enabled	The node's resilience in receiving payments (the maximum payment amount that a node could receive while accounting for how it is split among its neighbors).
Betweenness	$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$	G_2	N/A	Valid payment path	The frequency with which the node is positioned during payments between nodes that select the shortest paths (fewer hops). A node may be able to make an income through fees and information gathering about other payments.
Weighted betweenness (cap)	$C_B^w(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}^w(v)}{\sigma_{st}^w}$	G_2	f	Valid payment path	The frequency with which the node is positioned during payments between nodes that select the least expensive paths (the smaller amount of fees). By charging fees and collecting data from other payments, a node can generate an income.
	$C_B^w(v) \Rightarrow C_B^w(v)_c$		c	Valid payment path	The frequency with which a node might possibly transact a larger amount. Higher channel capacities allow a node to select the shortest paths when it is in the middle of a payment connecting other nodes.
Flow-based betweenness	$C_F(v) = \frac{\sum_{s \neq v \neq t \in V} \lvert f' \rvert_{st}(v)}{\sum_{s \neq v \neq t \in V} \lvert f' \rvert_{st}}$	G_2	c	Valid payment path	The amount of bitcoins that a node can send and receive by connecting different nodes for payments on the network. Resilience against disconnection (For more details, refer to **Flow-based Metric Selection** in III).
Current-flow betweenness	$C_{CF}(v) = \frac{\sum_{s \neq t \in V} I_V{}^{(st)}}{\frac{1}{2}n(n-1)}$	G_2	c	Valid payment path	The node's handling of bitcoins that flow through the payment route to different nodes in the network.
Closeness	$C_C(v_i) = \frac{1}{\sum_{j \in [1,n], j \neq i} d(v_i, v_j)}$	G_2	N/A	Valid payment path	When a node selects the shortest routes, how close it is to the other nodes in the network (the smaller number of hops). Fewer interactions with third-party nodes (privacy and security).
Weighted closeness	$C_C^w(v) = \frac{1}{\sum_{u \neq v \in V} d^w(v,u)}$	G_2	f	Valid payment path	When a node selects the cheapest routes, how close it is to the other nodes in the network (lower fees). Fewer interactions with third-party nodes (privacy and security).

Tab. 6.7: An overview of the proposed measurements for centrality [163].

6.5 Measuring the LN

The moment a pair of nodes open and, at some point, close a payment channel, these two transactions are the only ones added to the blockchain. In theory, the payer and the payee can send an unlimited number of transactions to each other without committing them to the blockchain. A payer may send such transactions with the aid of the global view of the PCN topology, which is the main input for the routing algorithm that requires one to be aware of the structure of the network. In consequence, each node has to gather routing information through broadcasting messages (`channel_announcement` and `channel_update`) through the peer-to-peer network.

Although the transmitted messages contain information such as channel capacity, fee, and signatures, they lack to disclose the channel's balance due to privacy reasons. This factor could incur in that a payment may fail due to the uncertainty that sufficient funds are available to route a transaction. However, the payer may attempt to send a payment a given number of times, in which one could be successful. To avoid such a failure of insufficient funds, especially on multihop payments, LN uses HTLC to ensure balance security. Similarly, in case of a stuck payment, HTLC allows reverting it, by the expiration of the transaction time locks. However, to process HTLC, the nodes involved in payment must be online. Otherwise, funds locked could take place for some time, or even, in the worst case, the funds could be stolen by an adversary.

Based on the depiction of the LN model described in Section 6.1 and the description of restrictions in Section 6.3, we evaluate the results of the simulations obtained from the implemented metrics explained in Section 6.2. Altogether, we analyze 12 metrics divided into 5 scopes (degree, strength, Opsahl, betweenness, and closeness) as described in Table 6.7. Likewise, based on their scope we apply specific restrictions as are enabled channels (degree, strength, and Opsahl) and valid payment path (betweenness and closeness).

Our goal is to draw conclusions regarding the evolution of the metrics over time, i.e., since its conceptualization, we want to know if LN has been prone to be more or less centralized. As well, since our approach makes use of restrictions on paths, it makes us wonder whether or not their use affects the results of the computations and if this is the case, how much error is injected. On the other hand, from the analysis of metrics as betweenness, we are interested to know the degree of error injected if normal betweenness is used, as well as if there are lots of differences between the results of the different metrics based on the rank correlation coefficient.

6.5.1 Snapshots, dataset and the Network

In order to make multihop payments, LN clients need to know the current state of the network, that is, which other nodes there exist, what channels do they maintain, and what are policies applicable to those channels. A **snapshot** of the LN is a graph representing the current state of the network from the point of view of a node.

Although a snapshot captures the composition of the network, its scope does not cover the totality of channels that might exist. The view of the network depends on the information that a node collects, and probably private channels between other pairs of nodes will not be reflected. Therefore, our analysis is limited to public channels since roughly 13.48% are private channels [176].

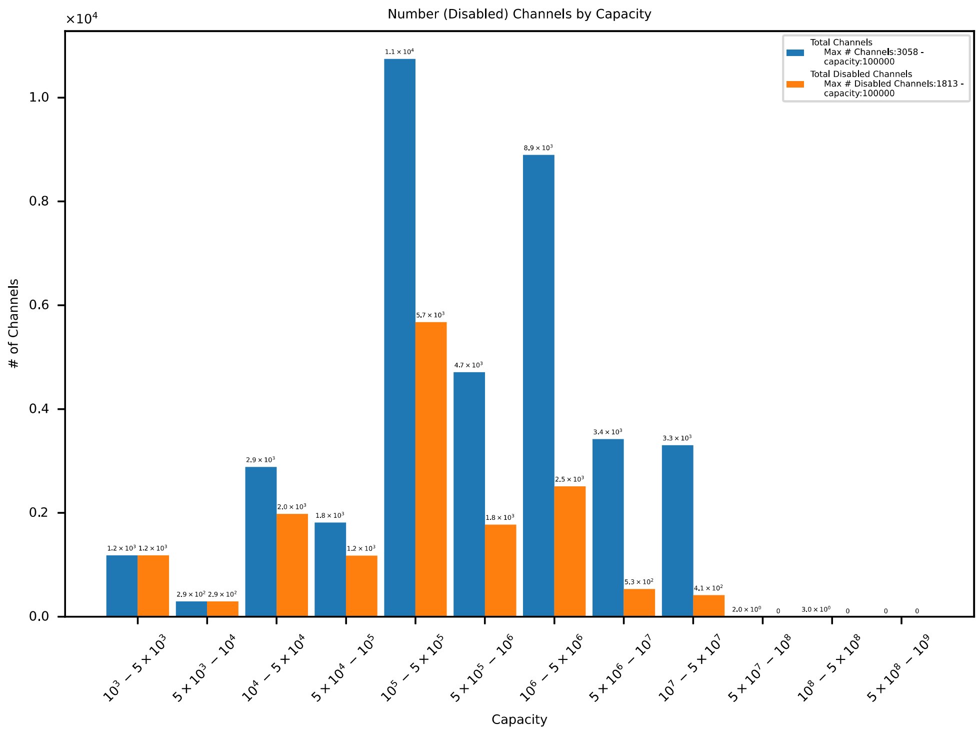
In this work, we use a dataset of LN snapshots captured by Elias Rohrer [120, 177]. The dataset contains snapshots of the network every 6 hours, over a two-year period (from October 2018 to November 2020). For our analysis, we subsampled the dataset and selected one weekly snapshot. We omitted periods where data were corrupted. Moreover, whenever the snapshot represents a disconnected graph, we restrict our analysis to the biggest connected component (that always contains more than 99% of the nodes of the network). Table 6.8 summarizes the main properties of the selected snapshots.

Snapshots	Nodes	Channels	Average degree	Disabled Channels (%)	# of node pairs with multiple channels
2018_10_31__12_00	1,548	7,146	9.2326	2,296 (32.12%)	304
2018_11_07__12_00	1,599	7,425	9.2871	2,319 (31.23%)	311
2019_01_23__12_00	2,348	14,383	12.2513	4,258 (29.60%)	759
2019_02_27__12_00	3,590	30,546	17.0173	9,986 (32.69%)	1,848
2019_03_20__12_00	2,555	15,863	12.4172	5,441 (34.29%)	896
2019_04_27__12_00	2,125	6,435	6.0565	1,708 (26.54%)	459
2019_07_31__12_00	5,696	37,246	13.0779	15,530 (41.69%)	2,608
2019_08_07__12_00	5,824	36,030	12.3729	15,808 (43.87%)	2,615
2019_08_28__12_00	5,912	36,288	12.276	15,283 (42.11%)	2,583
2019_09_25__12_00	5,948	36,077	12.1308	14,346 (39.76%)	2,470
2019_10_30__12_00	5,630	31,252	11.102	12,578 (40.24%)	2,065
2020_02_26__12_00	6,386	36,170	11.3279	15,049 (41.60%)	2,468
2020_03_25__12_00	6,568	35,976	10.9549	14,928 (41.49%)	2,478
2020_04_29__12_00	6,822	36,296	10.6409	15,400 (42.42%)	2,544
2020_05_13__12_00	5,523	20,187	7.3102	9,674 (47.92%)	1,379
2020_10_28__12_00	3,714	7,426	3.9989	7,426 (100%)	140
2020_11_07__12_00	3,693	7,388	4.0011	7,388 (100%)	136

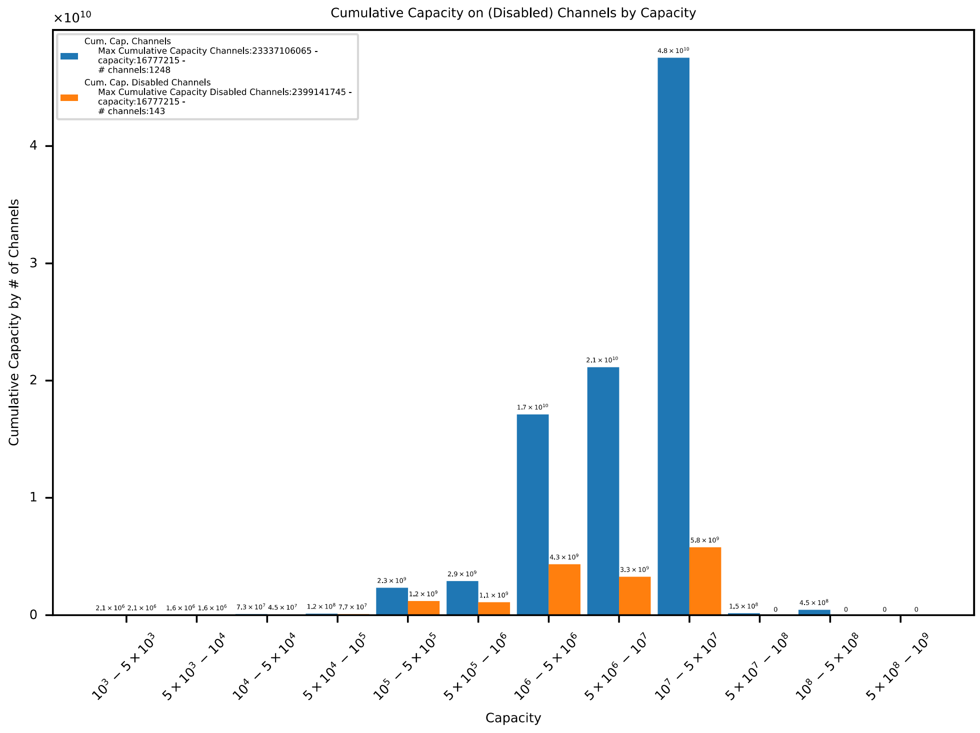
Tab. 6.8: Monthly snapshots between Oct. 2018 and Nov. 2020 [163].

The first consideration to take into account is related to the channels, which can be either enabled or disabled. We consider a channel to be disabled when either of the policies of both nodes in the channel is set to $disabled = True$ or the whole policy is set to $node\_policy = null$.

Figure 6.6 shows the grouped distribution of the channels between pairs of nodes based on their capacities for the snapshot corresponding to Jul. 31, 2019. For instance, Figure 6.6a shows that there are $1.1 \cdot 10^4$ node pairs that created channels with capacities between 10^5 and $5 \cdot 10^5$ satoshis, of which $5.7 \cdot 10^3$ channels are disabled.



(a) Number of (disabled) channels connecting node pairs based on channel capacity.



(b) Total Capacity of (disabled) channels connecting node pairs based on channel capacity.

Fig. 6.6: Connection between nodes pairs, channels, and the capacity of those channels [163].

Even so, this range of channel capacities is not the one with the highest accumulated capacity among the channels. Figure 6.6b shows that between 10^7 and $5 \cdot 10^7$ satoshis in channel capacities, there is a cumulative capacity of $4.8 \cdot 10^{10}$ of satoshis of which $5.8 \cdot 10^9$ satoshis are on disabled channels. Of this grouped distribution, $16.7 \cdot 10^9$ satoshis are the capacity setting that has the highest accumulated capacity with an amount of $23.3 \cdot 10^9$ between 1,248 pairs of nodes. Similarly, for that accumulated capacity, $2.3 \cdot 10^9$ remain among 143 disabled channels. These properties shape the LN topology that makes it peculiar, even more so, if we consider restrictions in the computation of the centrality metrics.

Another interesting property of the network is shown in the last column of Table 6.8, where it shows the number of pairs of nodes with more than one channel (multiple channels [178]). Figure 6.7 shows the results of the aforementioned column, considering both the enabled and disabled channels for the snapshot with the highest number of channels (Jul. 31, 2019). Note that a vast majority of node pairs share at least one channel and a small percentage have two or more that correspond to hub nodes on the LN.

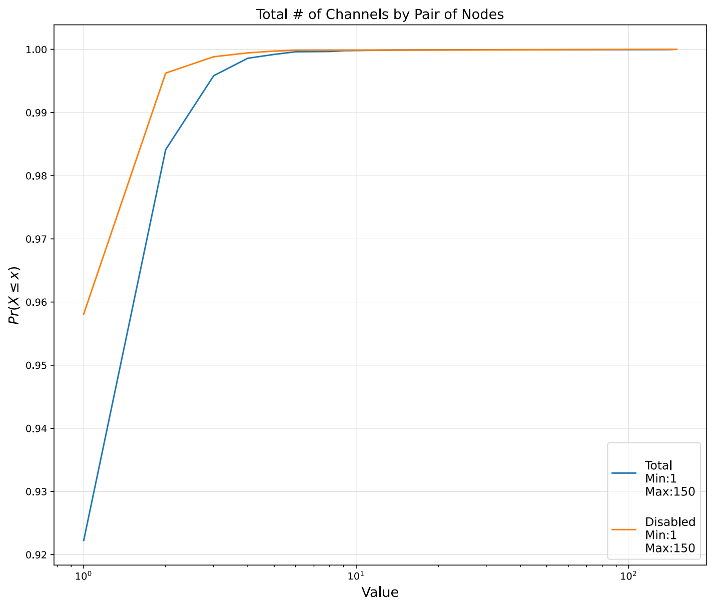


Fig. 6.7: Number of channels connecting each nodes pairs [163].

6.5.2 The effects of restrictions on centrality

As we explained in Section 6.3, not all paths on the LN graph can be used as payment paths. Therefore, additional restrictions must be considered to ensure that a given path can be used to make a payment on the LN. In this section, we provide the results of our experiments by calculating the centrality measures directly on the LN graph. Then, we compare these results with the same metrics calculated taking into account the restrictions on valid payment paths and enabled channels. Due to space constraints, results from one single snapshot (Sept. 4, 2019) are included. For the remaining snapshots, similar results were obtained in the analyzed graphs.

As shown in Table 6.9, we run two different simulations to compute the centrality measures.

Restriction	Parameters	1^{st} simulation	2^{nd} simulation
0	channel flag	enabled	enabled
1	balance	-	10^5
2	max path	20	20
3	max HTLC	-	14
4a	HTLC timeout	-	valid payment
4b	min payment	-	10^5

Tab. 6.9: Parameters for the simulations [163]

In the 1^{st} simulation, two general restrictions are considered that remain throughout the 2^{nd} simulation. On the one hand, to be able to use a channel in a payment path, the policies of both nodes in the channel must be configured as enabled (restriction 0 in Table 6.9). On the other hand, restriction 2 of Section 6.3 indicates that payment paths cannot have more than 20 hops. Therefore, we use the value given in the LN specification for the routing protocol[5] [176]. Note that both restrictions are deterministic and enforced by standard LN payment protocols.

In the 2^{nd} simulation, we apply all restrictions defined in Section 6.3 in which multiple parameters are defined. First of all, we set to $100k$ (10^5) satoshis the amount of the payment in our simulation, so the minimum balance needed in restriction 1 should be that value. Furthermore, as indicated above, we consider the limit of 20 hops for restriction 2. Regarding restriction 3, the LN specification for opening a channel[6] states that 483 is the limit for the number of pending HTLCs. Even though, we set the value of 14 as the limit for the number of HTLCs existing for each channel. Moreover, through a seed, we generate pseudo-random HTLCs, holding the same random generation for both simulations. This process is executed for each channel with its corresponding payment amounts and timeouts. The former reduces the balance in the channel until, at most, $balance = 0$. The latter increases for each channel the total timeout up to the upper limit given by $time\_lock\_delta$, which corresponds to the validation of restriction 4a. Finally, the minimum HTLC needed in restriction 4b is fixed to $\sigma = 10^5$.

[5]See : $HopLimit$ - `https://github.com/lightningnetwork/lnd/blob/40d63d5b4e317a4acca2818f4d5257271d4ac2c7/routing/pathfind.go`

[6]See: $max\_accepted\_htlcs$ - `https://github.com/lightningnetwork/lightning-rfc/blob/master/02-peer-protocol.md`

Note that, in addition to these parameters defined for given restrictions, other values are needed in the payment network for the simulations. In particular, the balance of each node in the payment channel is required to decide whether a payment can be forwarded through that channel. Although for privacy reasons, channel balances are not publicly available, it has been proven that it is possible to learn channel balances executing any of the attacks already described in the literature [179]. However, for ethical reasons, we do not perform these attacks on the live network, so in both simulations, we generate the channel balances through a constant distribution, i.e., half the capacity is assigned to each side of the channel.

Even though we use the parameters of the simulations as a starting point to evaluate the network as a whole, we can indeed tweak such values to create different scenarios and therefore evaluate such a network to improve its payment algorithm. For instance, we can use average payment amounts expressed in satoshis for the restrictions 1 and 4b. We can consider one satoshi as the minimum interval, which is the default minimum HTLC given by the LN policy, and 10^5 satoshis as the maximum interval, since values above this interval have the highest failure rate for the payment path. This scenario can increase the number of nodes participating in payments, but can also be reduced by the fees charged for each hop in a payment path.

Similarly, we can obtain a varied engagement of the nodes in the network using distributions other than the constant for the balance, such as uniform, normal, exponential, or beta distributions. Such diverse assignations of the balance may describe a scenario closer to the network reality. Likewise, for restriction 4a, we can draw a scenario in which the timeouts increase on each pending HTLC. This modification affects the availability of channels for a payment path since it is constrained by $time\_lock\_delta$. These scenarios are an option to have a deeper analysis focused on improving the payments, but we limit the scope of our analysis.

In the next sections, we provide the results for each of the metrics, showing the cumulative distribution function (CDF) of the centrality values of all the nodes in the graph. We split the results into two different measures, degree-based measures and path-based measures, indicating both results for the 1^{st} and the 2^{nd} simulation.

Degree-based

Degree-based centrality measures are degree $C_D(v_i)$, strength $C_D^w(v_i)$, incoming strength $C_{D^-}^w(v_i)$, outgoing strength $C_{D^+}^w(v_i)$, Opsahl $C_D^{w\alpha}(v_i)$, incoming Opsahl $C_{D^-}^{w\alpha}(v_i)$ and outgoing Opsahl $C_{D^+}^{w\alpha}(v_i)$ (see Table 6.7 for details). The first three values can be measured over the modeled graph G_1 since such measures do not involve any edge direction of the graph. However, for incoming and outgoing measures of strength and Opsahl, G_2 needs to be used. Note that even when the pending HTLCs modify the channel balance, we get the same results for degree-based measures in both simulations. Since, although HTLCs reduce the capacity of the channel, this reduction only applies to metrics that consider incoming/outgoing values (balances), but not those that do not consider it (capacity). Consequently, even though the channel balance is essential at the time of payment, it is irrelevant in case the channel is disabled.

Regarding Node Degree, Figure 6.8 contains the values of this metric that are the same for the 1^{st} and 2^{nd} simulations. As well, taking into account restrictions makes the network average degree goes down from 12.25 to 7.18, and the median from 3.0 to 0.0. The average RMSE of node's degree is 18.73. From the plot, we can observe that, when not taking into account restrictions, almost 50% of nodes are well-connected with 3 or more connections. However, such good connectivity is reduced to 30% of nodes when restrictions are considered. These differences are the result of a network with lots of channels disabled with 29,882 out of 72,274 channels which represent 41.34% disabled channels. As we review the remaining metrics, we will further explain this behavior in the next paragraphs.

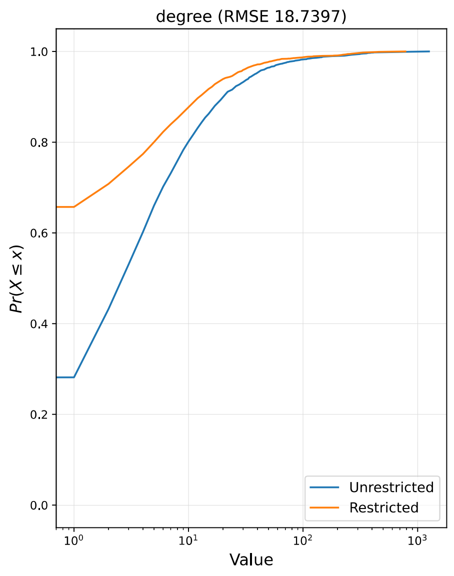


Fig. 6.8: CDF of $C_D(v_i)$ Node degree [163].

Differences between restricted and unrestricted measures are also prominent when analyzing Strength and Opsahl metrics that derive from the capacity[7] of the channels. As shown in Figure 6.9, average node strength has a reduction of 16.64% (from $28.54 \cdot 10^6$ to $23.79 \cdot 10^6$ satoshis), and the median of 100% (from $972 \cdot 10^3$ to 0 satoshis). Also, in Figure 6.10, the reduction on average Opsahl centrality (for $\alpha = 0.5$) is 26.26% ($16.18 \cdot 10^3$ to $11.93 \cdot 10^3$ satoshis) and the median once

[7]Units are expressed in satoshis as such measurements are derived from channel capacity.

again 100% ($1.67 \cdot 10^3$ to 0 satoshis). These differences are the result of 83.34% ($TotalUnrestrictedCapacity = 84,173,823,510$ and $TotalRestrictedCapacity = 70,158,741,106$) of the overall capacity is on enabled channels.

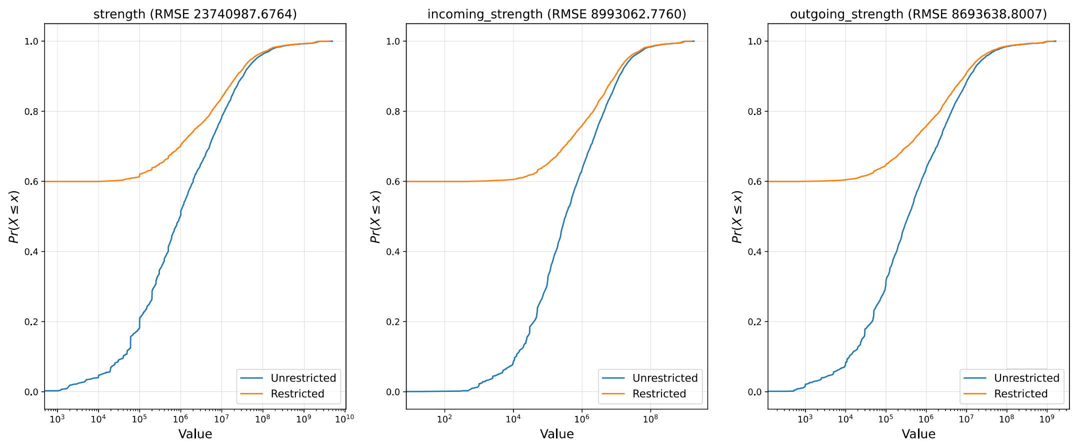


Fig. 6.9: CDF of: $C_D^w(v_i)$ strength, $C_{D-}^w(v_i)$ incoming and $C_{D+}^w(v_i)$ outgoing strength for the 1st and 2nd simulations [163].

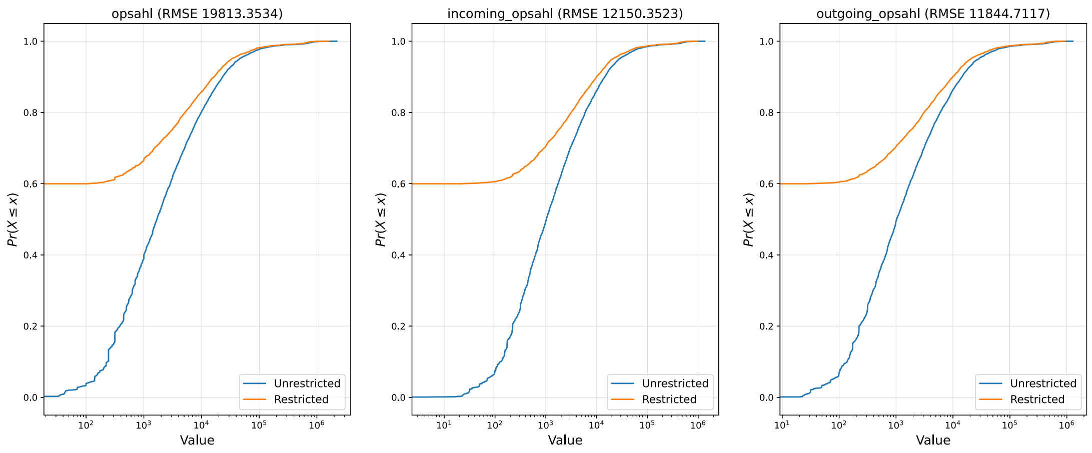


Fig. 6.10: CDF of: $C_D^{w\alpha}(v_i)$ Opsahl, $C_{D-}^{w\alpha}(v_i)$ incoming and $C_{D+}^{w\alpha}(v_i)$ outgoing Opsahl for the 1st and 2nd simulations [163].

II Path-based

Path-based centrality metrics are Betweenness $C_B(v)$, Weighted betweenness $C_B^w(v)$ and $C_B^w(v)_c$, Flow-based betweenness $C_F(v)$, Current-flow betweenness $C_{CF}(v)$, Closeness $C_C(v_i)$ and Weighted closeness $C_C^w(v)$ (see Table 6.7 for details). Note that all those metrics are computed using edge direction, so in our model definition and simulation they are computed over G_2. That means in the directed graph that represents the network with its balances, fees, and the blocked balance h_{ij} by the existing HTLCs. The total value h_{ij} for each channel and its timeout is randomly set through a seed that generates the same values for both simulations, which affects the balance of each side in the payment channel.

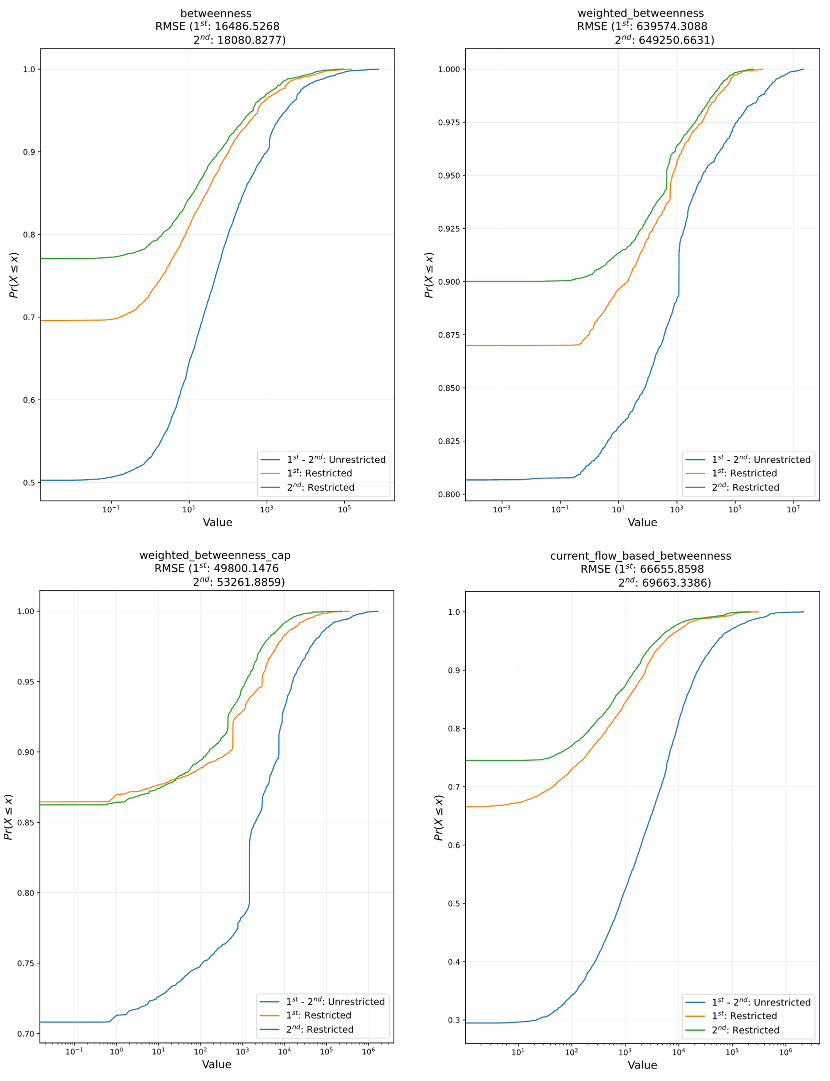


Fig. 6.11: CDF of: $C_B(v)$ betweenness, $C_B^w(v)$ fee- and $C_B^w(v)_c$ capacity-weighted betweenness, and $C_{CF}(v)$ current flow betweenness for the 1^{st} and 2^{nd} simulations [163].

Figure 6.11 and 6.12 show the results of each simulation for the betweenness and closeness centrality respectively with its variations as described previously in Table 6.7. At the moment, we compare the results of the metrics with weights for each simulation. There is a fluctuation in the values according to how the restrictions were used to compute the given metric. For instance, on the average ***betweenness*** for 1^{st} simulation, the reduction is 22.70%, however, this percentage compared with the 2^{st} simulation is 14.52%.

Instead, the average ***weighted_betweenness*** has values of 2.90% and 1.76% respectively, which shows a certain relationship given by the decrease in the balance because of existing HTLCs. On the contrary, for the average ***weighted_betweenness_cap***

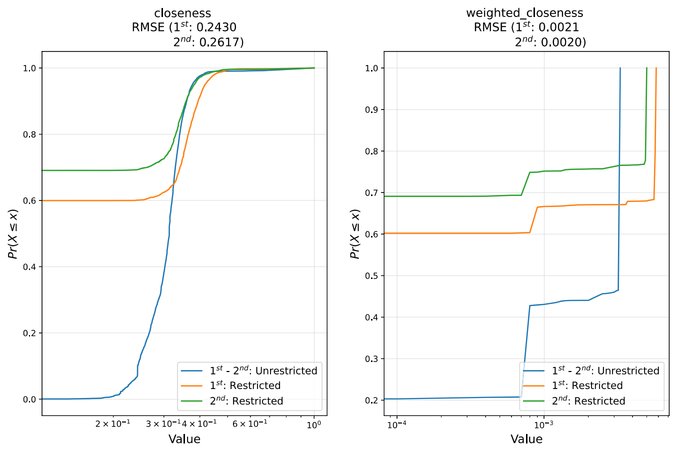


Fig. 6.12: CDF of: $C_C(v_i)$ closeness and $C_C^w(v)$ weighted closeness for the 1^{st} and 2^{nd} simulations [163].

which are 13.74% and 6.20% respectively, the aforementioned relationship does not last since the centrality measure has to consider paths with greater capacity to make a payment. Finally, the ***current_flow_betweenness*** metric has an average of 12.17% and 8.39% respectively, which when comparing its average ratio with ***weigthted_betweenness*** metric, its results (1.45 and 1.57 respectively) are similar. Based on these results and the fact that betweenness indicates how much control has a node over the network, the centrality should be measured by a combination of factors such as: fee, capacity, and balance instead of relying on a unique property of the channel. Table 6.10 summarizes the values of both simulations for the betweenness centrality.

	1^{st} Simulation				2^{nd} Simulation			
	%	RMSE	Average		%	RMSE	Average	
			Unrestricted	Restricted			Unrestricted	Restricted
betweenness	22.70%	16,486.5268	1,827.987	415.075	14.52%	18,080.8277	1,827.987	265.52
weighted_betweenness	2.90%	639,574.3088	53,515.367	1,553.671	1.76%	649,250.6631	53,515.367	944.274
weighted_betweenness_cap	13.74%	49,800.1476	6,797.366	934.270	6.20%	53,261.8859	6,797.366	421.803
current_flow_based_betweenness	12.17%	66,655.8598	15,290.226	1,861.887	8.39%	69,663.3386	15,290.226	1,283.072

Tab. 6.10: Betweenness Centrality for 1^{st} and 2^{nd} Simulations [163].

Subsequently, we took a similar approach with closeness metrics, as shown in Figure 6.12, to determine how reachable is a specific node. Along with whether it would be the most central node in case it is located at a node distance from each other. For the ***closeness***, its average is 47.75% and 34.93% respectively, which follows a similar explanation about its marked difference due to the presence of existing HTLCs in the channels. On the other hand, for the ***weighted_closeness***, the average is around 100% and 50% respectively, in which, the results show almost a flat trend that could be due to the presence of nodes with a large degree, i.e., a given number of nodes are reachable quite easy. In any case and regardless of the specific metric used, restrictions heavily affect the results. This tendency determines that centrality has to be defined by not only one metric but for a set of them considering more than one property of the network. Table 6.11 summarizes the values of both simulations for closeness centrality.

	1st Simulation				2nd Simulation			
	%	RMSE	Average		%	RMSE	Average	
			Unrestricted	Restricted			Unrestricted	Restricted
closeness	47.75%	0.2430	0.312	0.149	34.93%	0.2617	0.312	0.109
weighted_closeness	100%	0.0021	0.002	0.002	50%	0.0020	0.002	0.001

Tab. 6.11: Closeness Centrality for 1^{st} and 2^{nd} Simulations [163].

6.5.3 Relevance of nodes according to Centrality

Once we review the results of the metrics as a whole, it is appropriate to take a closer look at the most relevant nodes from the snapshot of Sept. 4, 2019. In doing so, we intend to find out how important some nodes on the network are to carry out payments either by the number of channels connecting them to the network or by the total capacity among those channels. For our analysis, we select five nodes. Four out of 5,897 nodes show high connectivity and $Bitrefill\ Thor$ and $WalletOfSatoshi.com$ are the pair of nodes with the most channels with 153 between them, none of which is disabled. On the other hand, the node that opened the most channels is $1ML.com\ node\ ALPHA$ with 830 of which 400 are disabled. However, $LightningPowerUsers.com$ is the node that has the most channels with 1,255 of which 473 are disabled. Finally, $ACINQ$ is the node with the highest capital distributed among its channels with $4.89 \cdot 10^9$ satoshis.

I Degree-based

Table 6.12 shows, for the nodes mentioned above, the results and their positions within the degree-based metrics among the most relevant nodes. It is worth mentioning that these results come from G_1 where only the *enabled channel* restriction is considered. $LightningPowerUsers.com$ has the highest degree, so it can be considered as the main ***hub***[8] in the network. Instead, $1ML.com\ node\ ALPHA$ can be seen as a ***beacon***[9] since it has the most open channels with other nodes. Even though, its strength is the lowest compared to the other four nodes. Also, $ACINQ$ has the highest $strength$ among the nodes, which is one of the preferred metrics for analyzing a weighted network. This result could mean that this specific node has a high level of involvement in the network, although its degree is not the highest. Due to the greater number of channels connecting $WalletOfSatoshi.com$ and $Bitrefill\ Thor$, we can assume this pair of nodes is a ***bridge***[10]. Although $WalletOfSatoshi.com$

[8]Defined as a node that connects with many other nodes

[9]Defined as a node that handles information about the awareness of the network topology.

[10]Defined as a pair of nodes that create a tie between nodes that would otherwise be disabled to perform payments without a direct connection.

has a higher degree, it has a low strength compared to $Bitrefill\ Thor$. By analyzing Opsahl, which combines degree and strength with a tune parameter $\alpha = 0.5$, once again $LightningPowerUsers.com$ has a relative importance in the network. Nevertheless, even without the highest degree, $ACINQ$ is the node with both the highest strength and Opsahl. As a consequence, the importance of a node is not only due to the number of channels that connect it to its neighbors, but also due to its participation in the network.

Nodes	*degree*	*strength*	*Opsahl*
$LightningPowerUsers.com$	1255 (1st)	2,408,839,730 (6th)	1,738,704.6504 (2nd)
$ACINQ$	991 (3rd)	4,897,182,784 (1st)	2,202,977.1081 (1st)
$1ML.com\ node\ ALPHA$	884 (4th)	659,107,300 (55th)	763,315.6969 (24th)
$WalletOfSatoshi.com$	390 (17th)	1,302,003,973 (39th)	712,587.924 (34th)
$Bitrefill\ Thor$	248 (49th)	2,151,521,848 (12th)	730,463.8378 (30th)

Tab. 6.12: Comparison of the 1st and 2nd simulations' degrees, Opsahl, and strengths metrics [163].

II Path-based

On the other hand, Table 6.13 compiles the results and their positions within the path-based metrics for the same five nodes. Thus, we analyze them through the betweenness metrics, with restrictions on the *valid payment path* mentioned in Section 6.3. Overall, these metrics give us an idea of the extent to which a node participates in the transactions between other nodes. As well, it indicates that a node could control the network since its income is proportional to how central it is with respect to the payment route. Again, $LightningPowerUsers.com$ is the one with the highest betweenness scores, the same as the degree metric. The importance of this node lies not only in its numerous connections but also in how it stands among its neighbors, which makes it a ***broker***[11]. Although $ACINQ$ generates the highest fee income and has the greatest strength and Opsahl, it handles less capital compared to $LightningPowerUsers.com$. Besides, $1ML.com\ node\ ALPHA$, which has a low strength, its betweenness metrics results are quite higher compared to the bridge nodes. The reason could be because this node creates most of the channels that allow it to connect with the network without making mostly payments. On the contrary, when we compare the bridge nodes, the relevance of $Bitrefill\ Thor$ decreases with respect to $WalletOfSatoshi.com$. This fact is more evident on the metric $weighted\_betweenness\_cap$, which means that $WalletOfSatoshi.com$ has a higher capital distributed among its neighbors. As well, the revenue from fees charged by this node is slightly higher, which could mean that this node could be continuously chosen as a payment intermediary. Finally, the $current\_flow\_betweenness$ metric restates the behavior seen so far. Nodes with a higher degree and betweenness, especially with $weighted\_betweenness\_cap$, have a higher probability to participate

[11]Defined as the node that connects dispersed nodes in order to obtain a competitive advantage based on access to network information.

in payment routes, i.e., these nodes withstand a higher traffic load than most of their neighbors.

Nodes	*betweenness*		*weigthed_betweenness*	
	1^{st} simulation	2^{nd} simulation	1^{st} simulation	2^{nd} simulation
LightningPowerUsers.com	151,899.0754 (1^{st})	92,033.136 (1^{st})	173,505.7543 (8^{th})	122,204.9509 (9^{th})
ACINQ	113,773.3293 (2^{nd})	61,115.5338 (2^{nd})	431,786.3497 (4^{th})	235,449.8886 (4^{th})
1ML.com node ALPHA	81,270.1248 (4^{th})	52,827.8882 (5^{th})	146,710.3616 (12^{th})	91,583.8489 (12^{th})
WalletOfSatoshi.com	14,280.1376 (44^{th})	8,043.7234 (49^{th})	15,986.7698 (93^{rd})	3,958.9318 (150^{th})
Bitrefill Thor	2,084.7293 (153^{rd})	2,164.2028 (114^{th})	13,509.6234 (103^{rd})	10,277.3408 (101^{st})
Nodes	*weighted_bettweenness_cap*		*current_flow_betweenness*	
	1^{st} simulation	2^{nd} simulation	1^{st} simulation	2^{nd} simulation
LightningPowerUsers.com	120,297.0834 (6^{th})	83,946.1151 (3^{rd})	310,126.8883 (1^{st})	216,591.8611 (1^{st})
ACINQ	60,987.8653 (18^{th})	44,310.7451 (8^{th})	274,519.0144 (2^{nd})	189,553.5902 (2^{nd})
1ML.com node ALPHA	189,272.5644 (3^{rd})	196,024.2667 (2^{nd})	128,011.6723 (13^{th})	93,000.8053 (10^{th})
WalletOfSatoshi.com	34,613.1407 (34^{th})	8,349.4167 (64^{th})	67,345.7153 (48^{th})	42,294.1871 (51^{st})
Bitrefill Thor	586.0000 (488^{th})	446.0000 (454^{th})	55,804.9380 (54^{th})	41,117.8770 (52^{nd})

Tab. 6.13: Betweenness Metrics Comparison between the 1^{st} and 2^{nd} simulations [163].

Lastly, Table 6.14 contains very similar results between the five nodes for the closeness metrics. Although, when we analyze *closeness* values, the *LightningPowerUsers.com* node keeps a higher centrality. It indicates that the node is well connected at a short distance from the other nodes and could efficiently distribute payments. Regarding the *weighted_closeness* metric, the results of the five nodes are the same, which means that each node applies the same fee. Therefore, they can be used to route payments without diminishing their centrality.

Nodes	*closeness*		*weighted_closeness*	
	1^{st} simulation	2^{nd} simulation	1^{st} simulation	2^{nd} simulation
LightningPowerUsers.com	0.5418 (20^{th})	0.5080 (25^{th})	0.0058 (10^{th})	0.0050 (10^{th})
ACINQ	0.5286 (23^{rd})	0.5031 (26^{th})	0.0058 (15^{th})	0.0050 (15^{th})
1ML.com node ALPHA	0.4950 (42^{nd})	0.4475 (66^{th})	0.0058 (72^{nd})	0.0050 (78^{th})
Bitrefill Thor	0.4497 (119^{th})	0.4386 (79^{th})	0.0058 (237^{th})	0.0050 (242^{nd})
WalletOfSatoshi.com	0.4879 (52^{nd})	0.4702 (42^{nd})	0.0058 (291^{st})	0.0050 (299^{th})

Tab. 6.14: Closeness Metrics Comparison between the 1^{st} and 2^{nd} simulations [163].

6.6 Analysis of LN Node Centrality

The metrics that we propose are conceptually more suitable for measuring centrality in payment networks. However, these metrics have the drawback that they are computationally more expensive to calculate. In consequence, we are interested in observing the correlation between the metrics that we propose and other simpler ones. Since, if the correlation is high, then we can use the simplest ones as a proxy as long as there are computational restrictions. For that purpose, we use Spearman's rank correlation to determine the degree of association (strength and direction) of a monotonic relationship between two metrics. The value of the coefficient ranges from -1.00 to 1.00, depending on how the two variables are related, for the strongest negative and positive correlation, respectively. The sign of the coefficient corresponds to the direction of the relationship, i.e., if it is positive, one variable increases as the other tend to increase, meanwhile, if it is negative, one variable decreases as the other tends to increase.

For the first part of this analysis, we use the snapshot from Sept. 4, 2019, to compare the results of the metrics from the 1^{st} and 2^{nd} simulations. In addition, we decided to show the correlation coefficients of the metrics using a heat map, since it helps to visualize the variance between multiple metrics, show similarities between them as well as detect if there is any correlation between them. In Figure 6.13, whose results are analogous to both simulations, there is a low-to-medium relationship between the betweenness metrics, which remains when compared to the degree metric. On the other hand, there is a strong relationship between $weighted\_betweenness$ and $weighted\_betweenness\_cap$ metrics. In that case, the coefficient value is the same (0.62) among the results of both simulations, even though, in the 2^{nd} simulation: (1) the balance of the channels decreases because of the existing HTLCs and (2) the restrictions applied in the network.

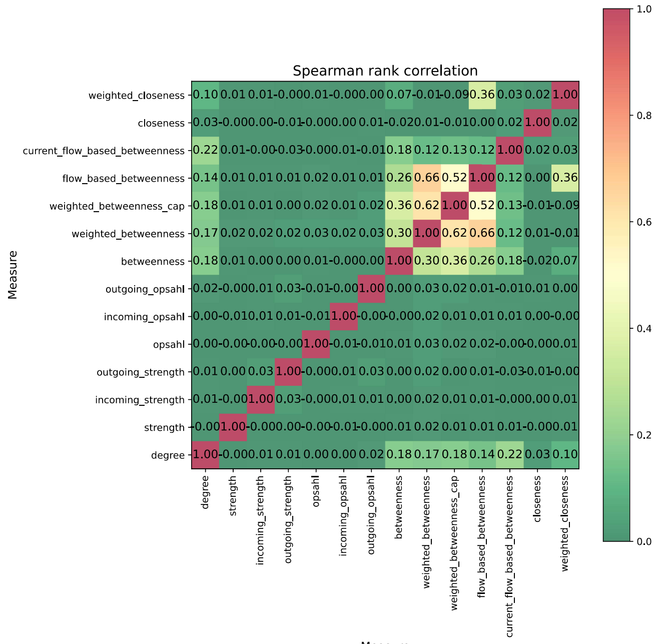
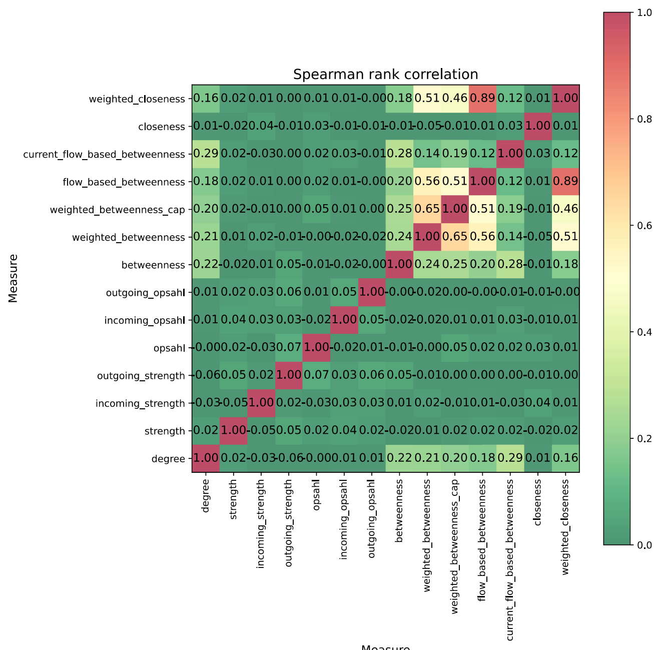
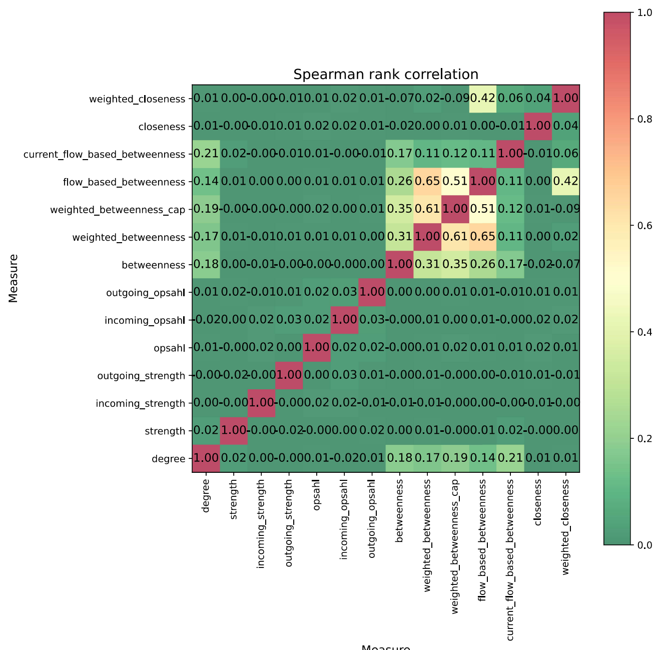


Fig. 6.13: 1^{st} and 2^{nd} simulations' metric correlations similarity [163].

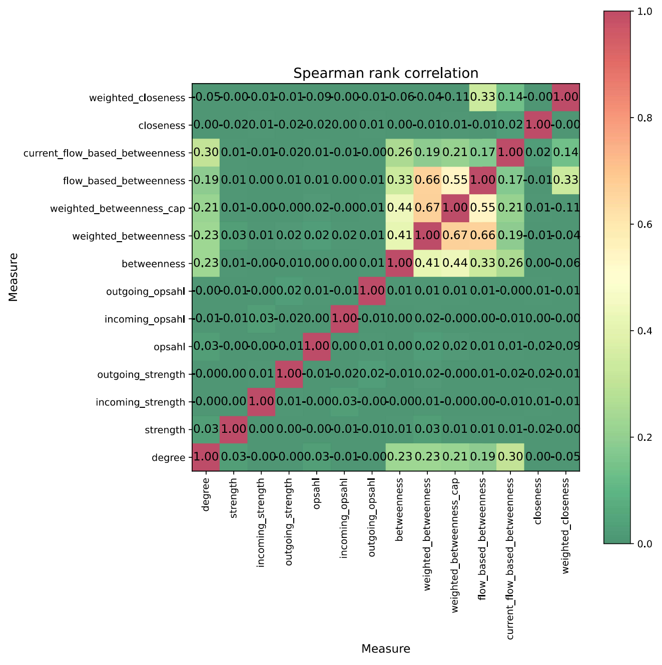
For the last part of this analysis, and as mentioned above, the snapshot set used in our study covers data for the span of a couple of years. These data allow us to obtain a wide range of information to analyze. We compared the results of the 1^{st} simulation between three snapshots since Oct. 2018, with a lapse of one year between the other two snapshots. Thus, we can infer, based on the results shown in Figure 6.14, that the degree metric keeps a constant correlation with the metrics of $betweenness$, $weighted\_betweenness$ (capacity and fee), and *current_flow_betweenness*. For instance, on 2018 the strength of association of degree metric against these four betweenness metrics was 0.22, 0.21, 0.20, and 0.29 respectively, as well in 2020 the results kept slightly similar values 0.23, 0.23, 0.21, and 0.30 respectively.



(a) Correlation October 2018.



(b) Correlation October 2019.



(c) Correlation October 2020.

Fig. 6.14: Metrics' correlation over a two-year period from Oct. 2018 to Nov. 2020 [163].

Similarly, among these same four metrics hold a strong correlation compared with the remaining metrics. In fact, comparing the results between 2019 and 2020 shows that the relationship strengthens over time. For instance, in 2019, the relationship between betweenness and the metrics of weighted betweenness (fee and capacity) and current flow betweenness is 0.31, 0.35, and 0.17, respectively. The results of the same metrics increased for 2020 with values of 0.41, 0.44, and 0.26, respectively. Although, when we compare both weighted betweenness metrics (fee and capacity), these metrics keep an even stronger correlation, the highest value being 0.67 for them in 2020. As the LN structure adjusts over time, the correlation comparison between snapshots gradually decreases for the betweenness and closeness metrics. For instance, in the case of weighted betweenness (fee and capacity) in Oct. 2018 they maintained a degree of relationship between 0.51 and 0.46, respectively. By Oct. 2020 it was reduced to -0.04 and -0.11, respectively.

Based on these results, the main conclusion could emerge to explain the correlation of the centrality metrics. For a given pair of metrics, the correlation values between different snapshots vary more than five percentage points from each other. However, these metrics can provide insight into the evolution and behavior of the network. This conclusion lies in its lack of dependence between one and the other metric, but they are also necessary because they do not have redundant information.

6.7 Conclusions

In this study, we focus on node centrality in LN, in which we integrate properties and restrictions that had previously been overlooked in similar studies. For that, two graphs model LN: G_1 with static channel information under an undirected weighted multigraph and G_2 with dynamic channel information under a directed weighted multigraph. These graphs collect all network information while preserving restrictions required to analyze LN in various contexts. The graphs include channel properties (channel capacity and balance, fees, and channel availability) and their restrictions (minimum balance to forward payments and available HTLCs to make a payment).

The proposed metrics are conceptually more appropriate for measuring node centrality in LN. The results show marked differences regarding centrality, with the path-based metrics showing a significant deviation from the $weighted\_closeness$ metric due to a decreased number of channels. Although the proposed metrics give us an understanding of LN centrality, the computational cost to compute some of these metrics is expensive, reinforcing the importance of studying the correlations between them. The correlation between metrics can serve as a means to determine if it is possible to substitute some metrics for others.

The main conclusion about LN centrality is that in complex networks, a single metric cannot determine its centrality but a combination of several of them. Therefore, the degree metric is recommended as a starting point along with channel properties and restrictions to analyze the network.

Part V

Conclusions and Future Work

7 Conclusions and Future Work

7.1 Conclusions

Nowaday, users demand fast, cheap, and private payment methods, but Bitcoin's design limits such properties, reducing its viability as global payment method. Visa can handle more transactions per second than Bitcoin, highlighting Bitcoin's processing limitations and potential for network congestion. These Bitcoin scalability problems have led to the creation of layer-2 solutions.

These solutions improve Bitcoin's scalability, speed, efficiency, and privacy but introduce new complexities and dependencies, like channel management and protocol interaction. Layer-2 solutions handle high transactionability but are still far from being the definitive solution to Layer-1 scalability problems. Above all Layer-2 solutions, LN is currently the most widely adopted for Bitcoin due to its scalability, low fees, and versatility, to name a few.

Overall, our objective with this research is to enhance LN to increase its performance, security, and robustness as a second-layer payment network. In that sense, as a first approach, we aim to improve the payment process through the LN configuration parameters analysis that affects the HTLC timeouts and consequently impacts network performance and security. Lastly, an approach to measuring LN centrality more appropriately involves considering parameters beyond the usual, such as channel capacity or the shortest payment paths.

The contributions presented in this research cover two aspects of the LN network, detailed in this section. Our initial contribution is to improve the operation and security of the network at a global level. Specifications guide the configuration of a client by recommending parameter values so that the LN client can function correctly. However, some of these parameters can be customized by the client user. Thus, the selection of the exact parameter values is compelling to increase the correct functionality of the network. As an initial step, we provide an analytical approach to select parameter values for $cltv\_expiry\_delta$ (δ) and $locktime\_max$ (T_{max}) as specified by each LN implementation. The δ parameter helps enforce the time lock mechanism for commitment transactions; whereas, the T_{max} parameter limits the maximum locktime value used in commitment transactions.

The first step we took was to define metrics to study the performance and security of the network. Once these metrics were defined, they allowed us to evaluate the parameter values. Various experiments allowed us to conclude that these parameter values used in LN implementations are not optimal regarding payment performance and network security. As a result of the analysis of the experiments, the best combination for the values of the parameters T_{max} and δ is 432 and 40, respectively. Once such values are in use, they represent a worst-case scenario for the attacker when reviewing the attack metrics, even though, for the LN implementations, it preserves the same success rate for payments. Regarding attacks, compared to parameter values used in other LN implementations, there is a reduction of at least a quarter of the overall time the attacker locks a victim's funds ($\widetilde{TBT}$). Even so, the funds that the attacker requires to carry out the attack are of an intermediate level. Above all, modifying these parameter values impacts network performance and security by reducing HTLC timeouts and preventing lockdown attacks that lock victim funds on the network.

Various studies have analyzed LN using graph theory tools to examine channel data such as existence and payment capacity. While some models depict LN as an undirected weighted graph, others focus on different aspects like multiple channels between nodes or a simplified representation without weights or directions. Central to these studies is the exploration of centrality measures like degree, betweenness, and closeness, suggesting a centralized network structure akin to a core-periphery model. Researchers also attempt to identify the most suitable centrality metric for LN, considering factors like income, traffic volume, failed payments, and payment success ratios. However, these studies often overlook key LN properties such as balance distribution, fees, blocked balances, minimum payment amounts, and channel availability. Overall, existing proposals fall short in capturing the intricate details of payment networks and the flow of satoshis between users.

For our last contribution, we proposed a model for the LN based on two graphs: G_1 with static channel information under an undirected weighted multigraph and G_2 with dynamic channel information under a directed weighted multigraph. Through these weighted graphs, we collect all the network information while preserving the restrictions that arise from the fact that the represented network is a PCN and are required to analyze LN in several contexts. The definition of the two graphs that model the network is as follows: channel capacity and node pairs with multiple channels without considering their availability to make payments are part of G_1. On the other hand, G_2 groups the same properties as G_1 but adds channel information, such as channel balances and their policies and blocked balances in HTLC. The LN modeled under these considerations allows us to capture the properties that best describe this network rather than the basic properties like channel capacity that most approaches consider to evaluate the network centrality.

Therefore, to study a network as LN, its model must include channel properties (channel capacity and balance, fees, and channel availability) and their restrictions (minimum balance to forward payments and available HTLCs to make a payment). These channel properties used in the model are relevant because they shape LN more accurately. Then, we suggest a set of metrics to assess the node centrality in LN. Finally, we use these metrics in our proposed model to study the centrality of the network. The results of the experiments using the modeled network show marked differences regarding centrality. For instance, the injected error, given by metrics based on payment path restrictions where their use affects the computation results, goes from one to almost thirteen percentage points for path-based metrics. Instead, the $weighted\_closeness$ metric has a substantial deviation of 50 percentage points due to a decreased number of channels. On the contrary, in a comparison between the normal betweenness and the rest of the betweenness alternatives, the injected error of the first is greater than that of the remaining ones, as is the case of $weighted\_betweenness\_cap$ and $weighted\_betweenness$ with almost one and seven percentage points of difference.

Although the metrics proposed for the experiments are computationally expensive to calculate, the correlation between metrics can serve as a means to determine whether or not it is possible to substitute some metrics for others. We evaluate these metric correlations to replace them with less expensive metrics. The results indicate that the relationship is low-to-medium when comparing the betweenness measurements with the degree metric. However, the correlation between them is constant throughout the years of study. On the contrary, the fee- and capacity-*weighted_betweenness* metrics have the strongest relationship; however, when analyzing the correlation among the betweenness metrics, it is strong, and over time, it becomes stronger. From the results of the experiments, it is feasible to argue that measures that are easy to compute can act as bridges for others that require more complex computations. The main conclusion about LN centrality is that in complex networks, a single metric cannot determine its centrality but a combination of several of them. However, as an initial point, the degree metric can be used in conjunction with channel properties and restrictions to analyze the network.

From the mentioned comparison, one can follow two directions separately to improve routing dependability. On the one hand, select the protocol with the best payment performance to propose a possible improvement over the chosen protocol. On the other hand, in light of the shortcomings and limitations of the earlier protocols, it would be advisable to propose a new one from a different angle.

7.2 Future Work

Although our studies contribute insights into how different configuration parameters in the LN clients affect the security and performance of the network, as well as metrics to evaluate centrality in the LN, several aspects work as follows:

- For our first contribution, our experiments intend to evaluate the network regarding lockdown attacks. Hence, to expand the analysis, the next step is to measure the impact of other attacks and how the LN responds to them. In that sense, we could embrace two directions: one that contemplates the use of distinct metrics to assess network performance, considering not only payments but also the cost that a payer incurs to make such a payment using fees. The other direction to take would be to close multiple channels simultaneously to perform a flooding attack [123], which has more impact on the network due to the severity of the attack.

- For our last contribution, we set specific values in our simulations for the parameters (balance, maximum path, maximum HTLC, HTLC timeout, and minimum payment) of the restrictions in the payment paths. So, we can tweak the values of these parameters to create different scenarios and, thus, evaluate the network to improve its payment method. On the other hand, due to our lack of knowledge of the values of channel balances that are not publicly available due to security reasons, we generate them synthetically through a constant distribution to make simulations of the LN. Since our analysis used a single model for this synthetic generation, it remains to test other models and evaluate the impact this has on the results obtained. For instance, a channel balance generation method may increase the participation of nodes involved in payments. However, it does not imply an increase in successful payments, whether due to high hop fees, longer routes, or other circumstances. Therefore, to achieve an in-depth analysis, it is necessary to define different scenarios under the proposed model for LN.

Bibliography

[1] Fatemeh Rezaeibagha and Yi Mu. „Efficient micropayment of cryptocurrency from blockchains“. In: *The Computer Journal* 62.4 (2019), pp. 507–517 (cit. on p. 3).

[2] Ladislav Kristoufek. „On Bitcoin markets (in) efficiency and its evolution“. In: *Physica A: statistical mechanics and its applications* 503 (2018), pp. 257–262 (cit. on p. 3).

[3] Andrew Urquhart. „The inefficiency of Bitcoin“. In: *Economics Letters* 148 (2016), pp. 80–82 (cit. on p. 3).

[4] Dejan Vujičić, Dijana Jagodić, and Siniša Ranđić. „Blockchain technology, bitcoin, and Ethereum: A brief overview“. In: *2018 17th international symposium infoteh-jahorina (infoteh)*. IEEE. 2018, pp. 1–6 (cit. on p. 3).

[5] Joseph Poon and Thaddeus Dryja. „The bitcoin lightning network: Scalable off-chain instant payments“. In: (2016) (cit. on pp. 3, 12, 34, 37–39, 48, 67).

[6] *A Next-Generation Smart Contract and Decentralized Application Platform*. Ethereum. `https://ethereum.org/en/whitepaper/` [Accessed: 2024-05-30]. 2024 (cit. on p. 3).

[7] Kyle Croman, Christian Decker, Ittay Eyal, et al. „On Scaling Decentralized Blockchains“. In: *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*. Vol. 9604. Springer. 2016, pp. 106–125 (cit. on pp. 3, 11).

[8] Hanna Halaburda and Guillaume Haeringer. „Bitcoin and blockchain: What we know and what questions are still open“. In: *NYU Stern School Business, New York, NY, USA, Tech. Rep* (2019) (cit. on p. 3).

[9] Giulia Iadisernia. „An experimental study of the Bitcoin lightning network properties“. In: (2022) (cit. on p. 4).

[10] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. „Concurrency and privacy with payment-channel networks“. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 455–471 (cit. on pp. 4, 38).

[11] Rami Khalil and Arthur Gervais. „Revive: Rebalancing off-blockchain payment networks“. In: *Proceedings of the 2017 acm sigsac conference on computer and communications security*. 2017, pp. 439–453 (cit. on p. 4).

[12] Changting Lin, Ning Ma, Xun Wang, and Jianhai Chen. „Rapido: Scaling blockchain with multi-path payment channels“. In: *Neurocomputing* 406 (2020), pp. 322–332 (cit. on p. 4).

[13] Satoshi Nakamoto. „Bitcoin: A peer-to-peer electronic cash system“. In: *Decentralized business review* (2008) (cit. on pp. 7, 34, 39).

[14] Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. „Flare: An approach to routing in lightning network“. In: *White Paper* 144 (2016) (cit. on pp. 8, 45, 50, 51).

[15] Andreas M Antonopoulos. *Mastering Bitcoin: Programming the open blockchain*. " O'Reilly Media, Inc.", 2017, pp. 131–138 (cit. on p. 8).

[16] Paul Müller, Sonja Bergsträßer, Amr Rizk, and Ralf Steinmetz. „The bitcoin universe: An architectural overview of the bitcoin blockchain“. In: (2018) (cit. on p. 8).

[17] Pablo Lamela Seijas, Simon Thompson, and Darryl McAdams. „Scripting smart contracts for distributed ledger technology“. In: *Cryptology ePrint Archive* (2016) (cit. on pp. 8, 11).

[18] Stefano Lande et al. „Formal Methods for Secure Bitcoin Smart Contracts“. In: (2021) (cit. on p. 9).

[19] Stefano Bistarelli, Andrea Bracciali, Rick Klomp, and Ivan Mercanti. „Towards automated verification of bitcoin-based decentralised applications“. In: *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. 2023, pp. 262–269 (cit. on pp. 9, 10).

[20] Craig Steven Wright and Stephane Savanah. *Method for compiling from a high-level scripting language to a blockchain native scripting language*. US Patent 11,797,278. Oct. 2023 (cit. on p. 9).

[21] Rick Klomp and Andrea Bracciali. „On symbolic verification of Bitcoin's script language“. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings 13*. Springer. 2018, pp. 38–56 (cit. on p. 9).

[22] Peilin Zheng, Xiapu Luo, and Zibin Zheng. „BSHUNTER: Detecting and Tracing Defects of Bitcoin Scripts“. In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 307–318 (cit. on p. 10).

[23] Craig S Wright. „A proof of turing completeness in bitcoin script“. In: *Proceedings of SAI Intelligent Systems Conference*. Springer. 2019, pp. 299–313 (cit. on p. 10).

[24] *Is there a maximum size of a scriptSig-scriptPubKey?* Bitcoin StackExchange. `http://bitcoin.stackexchange.com/questions/35878/is-there-a-maximum-size-of-a-scriptsig-scriptpubkey` [Accessed: 2024-05-08]. 2015 (cit. on p. 11).

[25] Bitcoin community. *Bitcoin*. WikiPage. `https://en.bitcoin.it`. 2010 (cit. on p. 11).

[26] Bitcoin community. *Block size limit controversy*. WikiPage. `https://en.bitcoin.it/wiki/Block_size_limit_controversy`. 2010 (cit. on p. 11).

[27] Bitcoin community. *Maximum number of opcodes in script*. Bitcoin StackExchange. http://bitcoin.stackexchange.com/questions/38230/maximum-number-of-op-codes-in-script. 2015 (cit. on p. 11).

[28] VISA Inc. *Visa fact sheet and quarter numbers*. https://www.visa.co.uk/dam/VCOM/download/corporate/media/visanet-technology/aboutvisafactsheet.pdf, https://s29.q4cdn.com/385744025/files/doc_downloads/2022/Visa-Inc-Fiscal-2022-Annual-Report.pdf and https://usa.visa.com/run-your-business/small-business-tools/retail.html, 2022. Accessed: 2023-08-05 (cit. on p. 11).

[29] Anantha Divakaruni and Peter Zimmerman. „The lightning network: Turning bitcoin into money“. In: *Finance Research Letters* 52 (2023), p. 103480 (cit. on p. 12).

[30] Ali Abdullah and AM Mutawa. „An Invitation Model Protocol (Imp) for the Bitcoin Asymmetric Lightning Network“. In: *Symmetry* 15.6 (2023), p. 1273 (cit. on pp. 12, 46).

[31] Jian-Hong Lin, Emiliano Marchese, Claudio J Tessone, and Tiziano Squartini. „The weighted bitcoin lightning network“. In: *Chaos, Solitons & Fractals* 164 (2022), p. 112620 (cit. on pp. 12, 46).

[32] Bitcoin Visuals. *Bitcoin and Lightning Network Charts*. Bitcoin Visuals. https://bitcoinvisuals.com/lightning. 2021 (cit. on p. 13).

[33] 1ML. *Lightning Network search and analysis engine*. 1ML. https://1ml.com/statistics?json=true. 2021 (cit. on pp. 13, 18).

[34] LND. *Lightning Network Daemon*. API LND Community. https://github.com/lightningnetwork/lnd. 2023 (cit. on p. 14).

[35] Core Lightning. *Core Lightning*. API Core Lightning Community. https://github.com/ElementsProject/lightning. 2023 (cit. on p. 14).

[36] Eclair. *Eclair*. API Eclair Lightning Community. https://github.com/ACINQ/eclair. 2023 (cit. on p. 14).

[37] Philipp Zabka, Klaus-T Foerster, Stefan Schmid, and Christian Decker. „Empirical evaluation of nodes and channels of the lightning network“. In: *Pervasive and Mobile Computing* 83 (2022), p. 101584 (cit. on p. 14).

[38] Satwik Prabhu Kumble and Stefanie Roos. „Comparative Analysis of Lightning's Routing Clients“. In: *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE. 2021, pp. 79–84 (cit. on pp. 14, 50).

[39] Nida Khan and Radu State. „Lightning network: A comparative review of transaction fees and data analysis“. In: *International congress on blockchain and applications*. Springer. 2019, pp. 11–18 (cit. on p. 14).

[40] Builder's Guide. *The Gossip Network*. https://docs.lightning.engineering/the-lightning-network/the-gossip-network. Accessed: 2024-05-18 (cit. on p. 16).

[41] Niklas Gögge, Elias Rohrer, and Florian Tschorsch. „On the Routing Convergence Delay in the Lightning Network“. In: *International Workshop on Data Privacy Management*. Springer. 2022, pp. 203–218 (cit. on pp. 16, 48, 66).

[42] IETF. „RFC 1035 - Domain Names“. In: *See https://www.ietf.org/rfc/rfc1035.txt* (1987) (cit. on p. 16).

[43] IETF. „RFC 3596 - DNS Extensions to Support IP Version 6“. In: *See https://www.ietf.org/rfc/rfc2782.txt* (2003) (cit. on p. 16).

[44] IETF. „RFC 2782 - A DNS RR for specifying the location of services (DNS SRV)“. In: *See https://www.ietf.org/rfc/rfc2782.txt* (2000) (cit. on p. 16).

[45] Andrew Samokhvalov, Joseph Poon, and Olaoluwa Osuntokun. „BOLT No. 7: P2P Node and Channel Discovery“. In: *See https://github.com/lightning/bolts/blob/master/07-routing-gossip.md* (2020) (cit. on pp. 17, 66).

[46] Andrew Samokhvalov, Joseph Poon, and Olaoluwa Osuntokun. „BOLT No. 10: DNS Bootstrap and Assisted Node Location“. In: *See https://github.com/lightningnetwork/lightning-rfc/blob/master/10-dns-bootstrap.md* (2018) (cit. on p. 17).

[47] Zeta Avarikioti, Tomasz Lizurej, Tomasz Michalak, and Michelle Yeo. „Lightning creation games“. In: *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2023, pp. 1–11 (cit. on p. 17).

[48] Matthias Grundmann and Hannes Hartenstein. „Towards a Formal Verification of the Lightning Network with TLA+“. In: *arXiv preprint arXiv:2307.02342* (2023) (cit. on p. 17).

[49] Chen Chen, Daniele E Asoni, David Barrera, George Danezis, and Adrain Perrig. „HORNET: High-speed onion routing at the network layer“. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1441–1454 (cit. on p. 18).

[50] Sergei Tikhomirov, Rene Pickhardt, Alex Biryukov, and Mariusz Nowostawski. „Probing channel balances in the lightning network“. In: *arXiv preprint arXiv:2004.00333* (2020) (cit. on pp. 18, 30).

[51] Paolo Guasoni, Gur Huberman, and Clara Shikhelman. „Lightning network economics: Channels“. In: *Management Science* (2023) (cit. on p. 18).

[52] Yan Qiao, Kui Wu, and Majid Khabbazian. „Non-intrusive and high-efficient balance tomography in the lightning network“. In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 2021, pp. 832–843 (cit. on pp. 18, 46).

[53] Bitcoin Visuals. *Bitcoin Market Price*. Bitcoin Visuals. https://bitcoinvisuals.com/market-price. 2021 (cit. on p. 18).

[54] Leslie Lamport. „The part-time parliament“. In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 277–317 (cit. on p. 19).

[55] Leslie Lamport. „Time, clocks, and the ordering of events in a distributed system“. In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 179–196 (cit. on p. 19).

[56] Weizhao Tang, Weina Wang, Giulia Fanti, and Sewoong Oh. „Privacy-utility tradeoffs in routing cryptocurrency over payment channel networks“. In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4.2 (2020), pp. 1–39 (cit. on p. 25).

[57] Peter Holzer. „Payment Channel Network Analysis with Focus on Lightning Network“. PhD thesis. Wien, 2020 (cit. on p. 25).

[58] Jan Camenisch and Anna Lysyanskaya. „A formal treatment of onion routing“. In: *Annual International Cryptology Conference*. Springer. 2005, pp. 169–187 (cit. on p. 27).

[59] George Danezis and Ian Goldberg. „Sphinx: A compact and provably secure mix format“. In: *2009 30th IEEE Symposium on Security and Privacy*. IEEE. 2009, pp. 269–282 (cit. on p. 27).

[60] Cristina Pérez-Sola, Alejandro Ranchal-Pedrosa, Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joaquin Garcia-Alfaro. „Lockdown: Balance availability attack against lightning network channels“. In: *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer. 2020, pp. 245–263 (cit. on pp. 29, 40, 44, 68–70).

[61] C-Lightning. *Open Channel C-Lightning*. API Lightning Community. `https://lightning.readthedocs.io/lightning-fundchannel_start.7.html?highlight=open%20channel`. 2019 (cit. on p. 31).

[62] Eclair. *Open Channel Eclair*. API Lightning Community. `https://acinq.github.io/eclair/#open`. 2019 (cit. on p. 31).

[63] Lightning Network Daemon LND. *Open Channel LND*. API Lightning Community. `https://api.lightning.community/?python#openchannel`. 2019 (cit. on p. 31).

[64] C-Lightning. *Send Payment C-Lightning*. API Lightning Community. `https://lightning.readthedocs.io/lightning-keysend.7.html?highlight=send%20payment`. 2019 (cit. on p. 32).

[65] Eclair. *Send Payment Eclair*. API Lightning Community. `https://acinq.github.io/eclair/#sendtonode`. 2019 (cit. on p. 32).

[66] Lightning Network Daemon LND. *Send Payment LND*. API Lightning Community. `https://api.lightning.community/?python#sendpaymentv2`. 2019 (cit. on p. 32).

[67] Mauro Conti, Ankit Gangwal, and Michele Todero. „Blockchain trilemma solver algorand has dilemma over undecidable messages“. In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 2019, pp. 1–8 (cit. on p. 33).

[68] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. „Solutions to scalability of blockchain: A survey“. In: *Ieee Access* 8 (2020), pp. 16440–16455 (cit. on p. 33).

[69] Donghui Ding, Xin Jiang, Jiaping Wang, et al. „Txilm: Lossy block compression with salted short hashing“. In: *arXiv preprint arXiv:1906.06500* (2019) (cit. on p. 33).

[70] Jeremy Levine. „Scalability controversy: understanding past cryptocurrency returns through Segregated Witness“. In: (2019) (cit. on p. 33).

[71] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. „{Bitcoin-NG}: A scalable blockchain protocol“. In: *13th USENIX symposium on networked systems design and implementation (NSDI 16)*. 2016, pp. 45–59 (cit. on p. 33).

[72] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. „Ouroboros: A provably secure proof-of-stake blockchain protocol“. In: *Annual international cryptology conference*. Springer. 2017, pp. 357–388 (cit. on p. 33).

[73] Sunoo Park, Albert Kwon, Georg Fuchsbauer, et al. „Spacemint: A cryptocurrency based on proofs of space“. In: *Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22*. Springer. 2018, pp. 480–499 (cit. on p. 33).

[74] Loi Luu, Viswesh Narayanan, Kunal Baweja, et al. „Scp: A computationally-scalable byzantine consensus protocol for blockchains“. In: *Cryptology ePrint Archive* (2015) (cit. on p. 33).

[75] Laizhong Cui, Shu Yang, Ziteng Chen, et al. „An efficient and compacted DAG-based blockchain protocol for industrial Internet of Things“. In: *IEEE Transactions on Industrial Informatics* 16.6 (2019), pp. 4134–4145 (cit. on p. 33).

[76] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. „Spectre: A fast and scalable cryptocurrency protocol“. In: *Cryptology ePrint Archive* (2016) (cit. on p. 34).

[77] Loi Luu, Viswesh Narayanan, Chaodong Zheng, et al. „A secure sharding protocol for open blockchains“. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 17–30 (cit. on p. 34).

[78] Adem Efe Gencer, Robbert van Renesse, and Emin Gün Sirer. „Service-oriented sharding with aspen“. In: *arXiv preprint arXiv:1611.06816* (2016) (cit. on p. 34).

[79] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, et al. „Omniledger: A secure, scale-out, decentralized ledger via sharding“. In: *2018 IEEE symposium on security and privacy (SP)*. IEEE. 2018, pp. 583–598 (cit. on p. 34).

[80] Yonatan Sompolinsky and Aviv Zohar. „Secure high-rate transaction processing in bitcoin“. In: *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*. Springer. 2015, pp. 507–527 (cit. on p. 34).

[81] Marko Vukolić. „The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication“. In: *Open Problems in Network Security: IFIP WG 11.4 International Workshop, iNetSec 2015, Zurich, Switzerland, October 29, 2015, Revised Selected Papers*. Springer. 2016, pp. 112–125 (cit. on p. 34).

[82] Bitcoin Cash. „Bitcoin Cash“. In: *See `https://www.bitcoincash.org`* (2008) (cit. on p. 34).

[83] Rami Khalil, Arthur Gervais, and Guillaume Felley. „Nocust-a securely scalable commit-chain“. In: *Cryptology ePrint Archive, Report 2018/642* (2018) (cit. on pp. 34, 35).

[84] Christian Decker and Roger Wattenhofer. „A fast and scalable payment network with bitcoin duplex micropayment channels“. In: *Stabilization, Safety, and Security of Distributed Systems: 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings 17*. Springer. 2015, pp. 3–18 (cit. on pp. 34, 37).

[85] bitcoinj. *Working with micropayment channels*. `https://bitcoinj.github.io/working-with-micropayments`, 2015. Accessed: 2023-08-09 (cit. on p. 34).

[86] Gavin Wood et al. „Ethereum: A secure decentralised generalised transaction ledger“. In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32 (cit. on p. 34).

[87] Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. „A survey on blockchain interoperability: Past, present, and future trends“. In: *ACM Computing Surveys (CSUR)* 54.8 (2021), pp. 1–41 (cit. on p. 34).

[88] Alexei Zamyatin, Dominik Harz, Joshua Lind, et al. „Xclaim: Trustless, interoperable, cryptocurrency-backed assets“. In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 193–210 (cit. on p. 34).

[89] Hangyu Tian, Kaiping Xue, Xinyi Luo, et al. „Enabling cross-chain transactions: A decentralized cryptocurrency exchange protocol“. In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 3928–3941 (cit. on p. 34).

[90] Tom M Mayer, Christoph Mai, and N Jesse. *Tokrex: Meta-system for real-time intra- and cross-chain swaps*. Tech. rep. Tech. Rep, 2017 (cit. on p. 35).

[91] Will Warren and Amir Bandeali. „0x: An open protocol for decentralized exchange on the Ethereum blockchain“. In: *URl: https://github. com/0xProject/whitepaper* (2017), pp. 04–18 (cit. on p. 35).

[92] Gavin Wood. „Polkadot: Vision for a heterogeneous multi-chain framework“. In: *White paper* 21.2327 (2016), p. 4662 (cit. on p. 35).

[93] Jae Kwon and Ethan Buchman. „A network of distributed ledgers“. In: *Cosmos, dated* (2018), pp. 1–41 (cit. on p. 35).

[94] Matthew Spoke, NE Team, et al. „Aion: Enabling the decentralized internet“. In: *AION, White Paper* (2017) (cit. on p. 35).

[95] Adam Back, Matt Corallo, Luke Dashjr, et al. „Enabling blockchain innovations with pegged sidechains“. In: *http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* 72 (2014), pp. 201–224 (cit. on p. 35).

[96] Amritraj Singh, Kelly Click, Reza M Parizi, et al. „Sidechain technologies in blockchain networks: An examination and state-of-the-art review“. In: *Journal of Network and Computer Applications* 149 (2020), p. 102471 (cit. on p. 35).

[97] Rami Khalil, Alexei Zamyatin, Guillaume Felley, Pedro Moreno-Sanchez, and Arthur Gervais. „Commit-chains: Secure, scalable off-chain payments“. In: *Cryptology ePrint Archive* (2018) (cit. on p. 35).

[98] Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. „Blockchain scaling using rollups: A comprehensive survey“. In: *IEEE Access* (2022) (cit. on p. 36).

[99] Thomas Lavaur, Jérôme Lacan, and Caroline PC Chanel. „Enabling blockchain services for IoE with Zk-Rollups". In: *Sensors* 22.17 (2022), p. 6493 (cit. on p. 36).

[100] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. „Arbitrum: Scalable, private smart contracts". In: *27th USENIX Security Symposium (USENIX Security 18)*. 2018, pp. 1353–1370 (cit. on p. 36).

[101] Jason Teutsch and Christian Reitwießner. „A scalable verification solution for blockchains". In: *arXiv preprint arXiv:1908.04756* (2019) (cit. on p. 36).

[102] Victor Costan and Srinivas Devadas. „Intel SGX explained". In: *Cryptology ePrint Archive* (2016) (cit. on p. 36).

[103] Sinisa Matetic, Karl Wüst, Moritz Schneider, et al. „{BITE}: Bitcoin lightweight client privacy using trusted execution". In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 783–800 (cit. on p. 36).

[104] Joshua Lind, Ittay Eyal, Peter Pietzuch, and Emin Gün Sirer. „Teechan: Payment channels using trusted execution environments". In: *arXiv preprint arXiv:1612.07766* (2016) (cit. on p. 36).

[105] Karl Wüst, Sinisa Matetic, Moritz Schneider, et al. „Zlite: Lightweight clients for shielded zcash transactions using trusted execution". In: *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer. 2019, pp. 179–198 (cit. on p. 36).

[106] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. „Sprites and state channels: Payment networks that go faster than lightning". In: *International conference on financial cryptography and data security*. Springer. 2019, pp. 508–526 (cit. on p. 37).

[107] Taisei Takahashi and Akira Otsuka. „Short paper: secure offline payments in bitcoin". In: *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer. 2020, pp. 12–20 (cit. on p. 37).

[108] Rafael Pass and Abhi Shelat. „Micropayments for decentralized currencies". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 207–218 (cit. on p. 37).

[109] Mike Hearn. „Micro-payment channels implementation now in bitcoinj". In: *Bitcointalk. org* (2013) (cit. on p. 37).

[110] Alejandro Ranchal Pedrosa, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. „Scalable lightning factories for bitcoin". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pp. 302–309 (cit. on p. 37).

[111] Conrad Burchert, Christian Decker, and Roger Wattenhofer. „Scalable funding of bitcoin micropayment channel networks". In: *Royal Society open science* 5.8 (2018), p. 180089 (cit. on p. 37).

[112] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, Julia Hesse, and Kristina Hostáková. „Multi-party virtual state channels". In: *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*. Springer. 2019, pp. 625–656 (cit. on p. 37).

[113] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. „General state channel networks“. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 949–966 (cit. on p. 37).

[114] Christoph Egger, Pedro Moreno-Sanchez, and Matteo Maffei. „Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks“. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 801–815 (cit. on pp. 38, 39).

[115] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. „PERUN: Virtual payment channels over cryptographic currencies.“ In: *IACR Cryptol. ePrint Arch.* 2017 (2017), p. 635 (cit. on p. 38).

[116] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. „Tumblebit: An untrusted bitcoin-compatible anonymous payment hub“. In: *Network and Distributed System Security Symposium*. 2017 (cit. on p. 38).

[117] D Robinson. *HTLCs considered harmful*. Stanford Blockchain Conference. 2019 (cit. on p. 39).

[118] Interledger. *Connector risk mitigations*. `https://github.com/interledger/rfcs/blob/main/0018-connector-risk-mitigations/0018-connector-risk-mitigations.md`. 2019 (cit. on p. 40).

[119] Ayelet Mizrahi and Aviv Zohar. „Congestion attacks in payment channel networks“. In: *International conference on financial cryptography and data security*. Springer. 2021, pp. 170–188 (cit. on pp. 40, 66).

[120] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. „Discharged payment channels: quantifying the lightning network's resilience to topology-based attacks“. In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2019, pp. 347–356 (cit. on pp. 40, 47, 48, 77, 96).

[121] Ben Weintraub, Cristina Nita-Rotaru, and Stefanie Roos. „Structural attacks on local routing in payment channel networks“. In: *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2021, pp. 367–379 (cit. on p. 40).

[122] Seungjin Lee and Hyoungshick Kim. „On the robustness of lightning network in bitcoin“. In: *Pervasive and Mobile Computing* 61 (2020), p. 101108 (cit. on p. 41).

[123] Jona Harris and Aviv Zohar. „Flood & loot: A systemic attack on the lightning network“. In: *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. 2020, pp. 202–213 (cit. on pp. 41, 118).

[124] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. „Anonymous multi-hop locks for blockchain scalability and interoperability“. In: *Cryptology ePrint Archive* (2018) (cit. on p. 41).

[125] George Kappos, Haaroon Yousaf, Ania Piotrowska, et al. „An empirical analysis of privacy in the lightning network“. In: *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I 25*. Springer. 2021, pp. 167–186 (cit. on p. 42).

[126] Piyush Kumar Sharma, Devashish Gosain, and Claudia Diaz. „On the anonymity of peer-to-peer network anonymity schemes used by cryptocurrencies“. In: *arXiv preprint arXiv:2201.11860* (2022) (cit. on p. 42).

[127] Satwik Prabhu Kumble, Dick Epema, and Stefanie Roos. „How lightning's routing diminishes its anonymity“. In: *Proceedings of the 16th International Conference on Availability, Reliability and Security*. 2021, pp. 1–10 (cit. on pp. 42, 50).

[128] Utz Nisslmueller, Klaus-Tycho Foerster, Stefan Schmid, and Christian Decker. „Toward active and passive confidentiality attacks on cryptocurrency off-chain networks“. In: *arXiv preprint arXiv:2003.00003* (2020) (cit. on p. 43).

[129] Alex Biryukov, Gleb Naumenko, and Sergei Tikhomirov. „Analysis and probing of parallel channels in the lightning network“. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2022, pp. 337–357 (cit. on p. 43).

[130] Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal-Pedrosa, Cristina Pérez-Solà, and Joaquin Garcia-Alfaro. „On the difficulty of hiding the balance of lightning network channels“. In: *Proceedings of the 2019 ACM asia conference on computer and communications security*. 2019, pp. 602–612 (cit. on pp. 43, 71).

[131] István András Seres, László Gulyás, Dániel A Nagy, and Péter Burcsi. „Topological analysis of bitcoin's lightning network“. In: *Mathematical Research for Blockchain Economy: 1st International Conference MARBLE 2019, Santorini, Greece*. Springer. 2020, pp. 1–12 (cit. on pp. 43, 47, 48, 77).

[132] Peng Wang, Hong Xu, Xin Jin, and Tao Wang. „Flash: efficient dynamic routing for offchain networks“. In: *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*. 2019, pp. 370–381 (cit. on pp. 44, 45, 50, 57).

[133] Giovanni Di Stasi, Stefano Avallone, Roberto Canonico, and Giorgio Ventre. „Routing payments on the lightning network“. In: *2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE. 2018, pp. 1161–1170 (cit. on pp. 44, 45, 67).

[134] Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. „A quantitative analysis of security, anonymity and scalability for the lightning network“. In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2020, pp. 387–396 (cit. on pp. 44, 47).

[135] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathleen Ruan, et al. „High throughput cryptocurrency routing in payment channel networks“. In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 2020, pp. 777–796 (cit. on p. 45).

[136] Andrew Samokhvalov, Joseph Poon, and Olaoluwa Osuntokun. „BOLT No. 0: Basis of Lightning Technology (BOLT)“. In: *See `https://github.com/lightning/bolts/blob/master/00-introduction.md`* (2018) (cit. on pp. 45, 67).

[137] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia Fanti, and Pramod Viswanath. „Routing cryptocurrency with the spider network“. In: *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*. 2018, pp. 29–35 (cit. on pp. 45, 50, 57).

[138] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. „SilentWhispers: Enforcing Security and Privacy in Decentralized Credit Networks.“ In: *NDSS*. 2017 (cit. on pp. 45, 50, 53).

[139] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. „Settling payments fast and private: Efficient decentralized routing for path-based transactions“. In: *arXiv preprint arXiv:1709.05748* (2017) (cit. on pp. 45, 50, 55).

[140] Cyril Grunspan and Ricardo Pérez-Marco. „Ant routing algorithm for the lightning network“. In: *arXiv preprint arXiv:1807.00151* (2018) (cit. on pp. 45, 50, 59).

[141] Yuwei Guo, Jinfeng Tong, and Chen Feng. „A measurement study of bitcoin lightning network“. In: *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE. 2019, pp. 202–211 (cit. on pp. 45, 47, 48, 77).

[142] Philipp Zabka, Klaus-T Foerster, Christian Decker, and Stefan Schmid. „Short paper: A centrality analysis of the lightning network“. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2022, pp. 374–385 (cit. on p. 46).

[143] Louis Bertucci. „Incentives on the lightning network: A blockchain-based payment network“. In: *Proceedings of Paris December 2020 Finance Meeting EUROFIDAI-ESSEC*. 2020 (cit. on p. 46).

[144] Philipp Zabka, Klaus-T Förster, Christian Decker, and Stefan Schmid. „A centrality analysis of the Lightning Network“. In: *Telecommunications Policy* 48.2 (2024), p. 102696 (cit. on p. 46).

[145] Andrea Lisi, Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. „Lightnings over rose bouquets: An analysis of the topology of the Bitcoin Lightning Network“. In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE. 2021, pp. 324–331 (cit. on p. 46).

[146] Stefano Martinazzi and Andrea Flori. „The evolving topology of the Lightning Network: Centralization, efficiency, robustness, synchronization, and anonymity“. In: *Plos one* 15.1 (2020), e0225966 (cit. on pp. 46–49, 77, 85).

[147] Finnegan Waugh and Ralph Holz. „An empirical study of availability and reliability properties of the bitcoin lightning network“. In: *arXiv preprint arXiv:2006.14358* (2020) (cit. on p. 46).

[148] Ferenc Béres, Istvan Andras Seres, and András A Benczúr. „A Cryptoeconomic Traffic Analysis of Bitcoin's Lightning Network“. In: *arXiv preprint arXiv:1911.09432* (2019) (cit. on pp. 47, 48, 77).

[149] Jian-Hong Lin, Kevin Primicerio, Tiziano Squartini, Christian Decker, and Claudio J Tessone. „Lightning Network: a second path towards centralisation of the Bitcoin economy“. In: *New Journal of Physics* 22.8 (2020), p. 083022 (cit. on pp. 47, 48, 77).

[150] Stefano Martinazzi. „The evolution of Lightning Network's Topology during its first year and the influence over its core values“. In: *arXiv preprint arXiv:1902.07307* (2019) (cit. on pp. 47, 48, 77).

[151] Xiao ZHANG et al. „Evaluation Formula for Communication Network Node Importance“. In: *Journal of Northeastern University (Natural Science)* 35.5 (2014), p. 663 (cit. on p. 47).

[152] Lorenzo Costantini, Carla Sciarra, Luca Ridolfi, and Francesco Laio. „Measuring node centrality when local and global measures overlap“. In: *Physical Review E* 105.4 (2022), p. 044317 (cit. on p. 47).

[153] Vincent Davis and Brent Harrison. „Learning a Scalable Algorithm for Improving Betweenness in the Lightning Network“. In: *2022 Fourth International Conference on Blockchain Computing and Applications (BCCA)*. IEEE. 2022, pp. 119–126 (cit. on p. 47).

[154] Marco Conoscenti, Antonio Vetrò, Juan Carlos De Martin, and Federico Spini. „The cloth simulator for htlc payment networks with introductory lightning network performance results“. In: *Information* 9.9 (2018), p. 223 (cit. on p. 48).

[155] Cosimo Sguanci and Anastasios Sidiropoulos. „Mass exit attacks on the lightning network“. In: *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2023, pp. 1–3 (cit. on p. 49).

[156] Mihai Plotean. „Improving the Anonymity of the Lightning Network using Sub-Optimal Routes“. In: (2021) (cit. on p. 50).

[157] Cyril Grunspan, Gabriel Lehéricy, and Ricardo Pérez-Marco. „Ant routing scalability for the lightning network“. In: *arXiv preprint arXiv:2002.01374* (2020) (cit. on p. 50).

[158] Stefanie Roos, Martin Beck, and Thorsten Strufe. „Voute-virtual overlays using tree embeddings“. In: *arXiv preprint arXiv:1601.06119* (2016) (cit. on p. 55).

[159] Andrew Samokhvalov, Joseph Poon, and Olaoluwa Osuntokun. „BOLT No. 11: Invoice Protocol for Lightning Payments“. In: *See `https://github.com/lightning/bolts/blob/master/11-payment-encoding.md`* (2020) (cit. on pp. 66, 67).

[160] Andrew Samokhvalov, Joseph Poon, and Olaoluwa Osuntokun. „BOLT No. 4: Onion Routing Protocol“. In: *See `https://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md`* (2020) (cit. on p. 66).

[161] Patrick McCorry, Malte Möser, Siamak F Shahandasti, and Feng Hao. „Towards bitcoin payment networks“. In: *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I 21*. Springer. 2016, pp. 57–76 (cit. on p. 67).

[162] Luis E Oleas-Chávez, Cristina Pérez-Solà, and Jordi Herrera-Joacomartí. „On the Selection of the LN Client Implementation Parameters“. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2020 International Workshops, DPM 2020 and CBT 2020, Guildford, UK, September 17–18, 2020, Revised Selected Papers 15*. Springer. 2020, pp. 305–318 (cit. on pp. 71, 73, 74).

[163] Luis E Oleas-Chávez, Cristina Pérez-Solà, and Jordi Herrera-Joancomartí. „Apples and Oranges: On How to Measure Node Centrality in Payment Channel Networks“. In: *IEEE Access* 10 (2022), pp. 55469–55487 (cit. on pp. 79, 84, 87, 88, 90, 92, 94, 96–99, 101–109).

[164] Linton C Freeman. „Centrality in social networks conceptual clarification“. In: *Social networks* 1.3 (1978), pp. 215–239 (cit. on p. 82).

[165] Mark EJ Newman. „Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality“. In: *Physical review E* 64.1 (2001), p. 016132 (cit. on pp. 83, 84).

[166] Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. „The architecture of complex weighted networks“. In: *Proceedings of the national academy of sciences* 101.11 (2004), pp. 3747–3752 (cit. on p. 83).

[167] Ulrik Brandes. „A faster algorithm for betweenness centrality“. In: *Journal of mathematical sociology* 25.2 (2001), pp. 163–177 (cit. on p. 84).

[168] Tore Opsahl, Filip Agneessens, and John Skvoretz. „Node centrality in weighted networks: Generalizing degree and shortest paths“. In: *Social networks* 32.3 (2010), pp. 245–251 (cit. on p. 85).

[169] Linton C Freeman, Stephen P Borgatti, and Douglas R White. „Centrality in valued graphs: A measure of betweenness based on network flow“. In: *Social networks* 13.2 (1991), pp. 141–154 (cit. on p. 90).

[170] Ulrik Brandes and Daniel Fleischer. „Centrality measures based on current flow“. In: *Annual symposium on theoretical aspects of computer science*. Springer. 2005, pp. 533–544 (cit. on pp. 91, 92).

[171] Mark EJ Newman. „A measure of betweenness centrality based on random walks“. In: *Social networks* 27.1 (2005), pp. 39–54 (cit. on p. 91).

[172] Zhipeng Luo. „Network research: exploration of centrality measures and network flows using simulation studies“. MA thesis. University of Twente, 2018 (cit. on p. 91).

[173] Taras Agryzkov, Leandro Tortosa, and Jose F Vicent. „A variant of the current flow betweenness centrality and its application in urban networks“. In: *Applied Mathematics and Computation* 347 (2019), pp. 600–615 (cit. on p. 92).

[174] Alessandro Lulli, Laura Ricci, Emanuele Carlini, and Patrizio Dazzi. „Distributed current flow betweenness centrality“. In: *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE. 2015, pp. 71–80 (cit. on p. 92).

[175] NetworkX. *Current Flow Betweenness Centrality*. NetworkX Network Analysis in Python. `https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.current_flow_betweenness_centrality.html#networkx.algorithms.centrality.current_flow_betweenness_centrality`. 2020 (cit. on p. 92).

[176] George Kappos, Haaroon Yousaf, Ania Piotrowska, et al. „An empirical analysis of privacy in the lightning network“. In: *arXiv preprint arXiv:2003.12470* (2020) (cit. on pp. 96, 99).

[177] Elias Rohrer. *Discharged-pc-data/Snapshots*. GitLab. `https://git.tu-berlin.de/rohrer/discharged-pc-data/-/tree/master/snapshots`. 2018 (cit. on p. 96).

[178] Alexei Biryukov, Gleb Naumenko, and Sergei Tikhomirov. „Analysis and Probing of Parallel Channels in the Lightning Network“. In: (2021) (cit. on p. 98).

[179] Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal-Pedrosa, Cristina Pérez-Solà, and Joaquin Garcia-Alfaro. „On the Difficulty of Hiding the Balance of Lightning Network Channels“. In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Asia CCS'19. New York, NY, USA: ACM, 2019, pp. 602–612 (cit. on p. 100).