

Addressing Scalability Issues of Blockchains with Hypergraph Payment Networks

Arad Kotzer, Bence Ladóczy, János Tapolcai and Ori Rottenstreich

Abstract—Payment channels are auspicious candidates in layer-2 solutions to reduce the number of on-chain transactions on traditional blockchains and increase transaction throughput. To construct payment channels, peers lock funds on 2-of-2 multisig addresses and opening channels between one another to transact via instant peer-to-peer transactions. Transactions between peers without a direct channel are made possible by routing the payment over a series of adjacent channels. In certain cases, this can lead to relatively low transaction success rates and high transaction fees. In this work, we introduce pliability to the construction of payment channels and graft edges with more than two endpoints into the payment graph. We took the liberty to give hyperedges as appellation to these constructions. We present hyperedge-based topologies to form hypergraphs and compare them to Bitcoin’s Lightning network and other state-of-the-art solutions. The results demonstrate that hyperedge-based implementations can both increase transaction success rate, in addition to decreasing the network cost by more than 50% compared to that of the Lightning Network.

Index Terms—Blockchain, Payment Channels, Network Algorithms.

I. INTRODUCTION

Transaction throughput in classical blockchain networks is severely limited by the fact that every participating node has to learn about every single transaction that is being broadcast in the network. On top of that, constant block time thwarts robust scaling in distributed ledger systems. With an average time of 10 minutes between blocks in Bitcoin for example, it takes roughly 1 hour for a transaction to be fully confirmed because as a rule of thumb, 6 later blocks should be built upon a block that contains the transaction for the recipient to be sure that the transaction was not under double spend attacks [1]. *Layer-2 (L2)* solutions have been proposed to improve upon the *transaction per second* (tps) metric of traditional blockchains (Table I). The general idea behind these proposals is to take peer-to-peer payments off-chain. Upon initiation, each participant locks some amount of cryptocurrency on the chain to enter a channel. Then, by utilizing private ledger technology, users can record state changes locally. Eventually, the final state gets published back to the main chain at a certain point in time. Scalability issues are present only in chains with large transaction amounts. Representative examples are the two most popular chains (Bitcoin and Ethereum). It is no surprise that L2 solutions are widespread for these networks. On the other hand, there is a significant difference between the two chains in

Table I: Average and maximum number of transactions per second in popular blockchains in May 2023

Blockchain	Transactions per Second	
	Average	Maximum
Bitcoin	4.7	7
Ethereum	12.2	15
Binance Smart Chain (BSC)	48	100
Cardano	2.9	1000
Ripple	17.3	1500
Solana	4.2	65000

that while Ethereum implements a fully programmable Turing machine, Bitcoin has limited scripting capabilities.

For Bitcoin, payment channels are the main L2 solution. More specifically, participants are organized into a so-called *L2 Payment Channel Network (PCN)*. In PCNs, two users not directly connected can still send money through a series of channels. In June 2024, Lightning Network [2] (LN) is the de facto PCN of Bitcoin with over 13000 nodes, 51000 channels and 5000 BTC of TLV (total locked value). Raiden [3] has come into the fold as a similar concept in Ethereum to facilitate micropayments, instant token swaps and peer-to-peer transactions. The advent of smart contracts on Ethereum paved the way for more complex solutions, including Plasma Chains [4] and Rollups, operating on the Ethereum blockchain. Rollups are designed to help alleviate network congestion and high transaction fees while maintaining security guarantees. Rollups create and process transaction bundles off-chain and then they write a summary of these on the chain using cryptographic primitives. Peers that are in the same group can transact with each other without referencing the L1 layer and it is the operators’ responsibility to collect these transactions into a batch, calculate a Merkle tree with its root and publish the state change represented by the Merkle root along with a proof to a smart contract on the L1 chain. Rollups necessitate smart contract functionality written in a Turing complete programming language (not yet supported in Bitcoin).

The building blocks of traditional PCNs are two-party payment channels. Channels can facilitate significantly more of transactions between the parties than the L1 layer. As long as the parties cooperate and do not exhibit malicious behavior there is no data written back to the chain. Users can simply update their respective balances based on the messages they exchange with one another. The chain is referenced only in case of a dispute or when the parties wish to close the channel. The two endpoint-like nature of the channels guarantees that misbehavior can be unambiguously detected, for example, when a dishonest party is trying to finalize the channel state

Arad Kotzer and Ori Rottenstreich are with the Technion, Haifa, Israel. Bence Ladóczy (HUN-REN-BME) and János Tapolcai are with Budapest University of Technology and Economics, Hungary.

with an expired state that they have both agreed on canceling. Should a malicious party broadcast such a state, the other party has a predefined time window under which an appeal can be made and the channel gets closed by the chain’s consensus rules. This requires both parties to continuously monitor the chain and detect potential misbehavior, otherwise, they are exposed to the risk of coin theft. Such a solution is completely decentralized, implying that there is no central entity that could potentially censor transactions or promote dishonest behavior in the network, thus the original properties of decentralization are preserved. On the other hand, the payment network is purely built up on two-party channels, which leads to a topology graph and to facilitate payments between parties not having a direct connection a path must be calculated through a series of payment channels. This results in the problem of routing, which is challenging and it is an actively researched topic [5], [6], [7], [8], [9].

Nevertheless, research is still far from being over and the theory of optimal PCN structures is neither so ripe nor so clean. Numerous studies have explored the concept of Bitcoin multi-party payment channels, commonly referred to as payment hubs or channel factories. These approaches often demand excessive communication between end nodes, particularly when implemented in a fully decentralized fashion. Given this landscape, the question arises whether two-ended channels are really the ideal structure in the Lightning Network. Our work finds answers to the following questions - *What advantage can one gain by using multi-endpoint channels in the Lightning Network?* Do 3-ended or 4-ended channels bring about any improvement in the transaction metrics of the payment network?

With this work we introduce and evaluate the concept of a hypergraph payment network (HPN) topology. We review existing multi-party implementations and use them as building blocks for HPNs. We additionally present a cost model of an HPN. We describe how to create two HPN topologies, NCH (node cover hypergraph) and FHS (fixed hyperedge size), and compare their performance with the original Lightning Network and other state-of-the-art solutions. Finally, we conclude the paper and suggest future research directions. To the best of our knowledge, no heuristics has been given in scientific literature to transform blockchain-based L2 payment networks into hypergraphs. We supplement our theoretical algorithms with actual implementations and evaluate the performance of the proposed methodology. A summary of the contribution of this work is given as follows:

- **Hypergraph Payment Network.** We provide an overview of existing Layer-2 proposals and introduce a generic framework to model them using the concepts of hyperedges and hypergraphs.
- **Clustering algorithms** By analyzing the real-world LN topology, algorithms to create hypergraphs are presented. Edges in the LN topology are transformed into edges in a hypergraph.
- **Simulations using empirical data** Several metrics of performance are evaluated. We open source the implementation along with the used inputs and the synthetic traffic data.

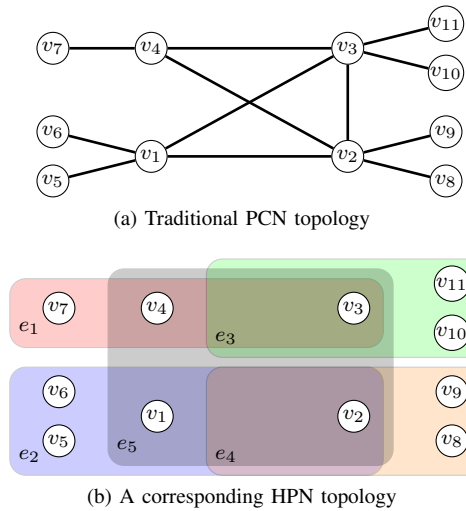


Figure 1: A traditional payment channel network (PCN) and a hypergraph payment network (HPN) construction.

We structure the paper as follows. In Section II we expatiate on L2 solutions, two-party and multi-party payment channel implementations, and introduce related works. In Section III-A we analyze the cost generated by multi-party channels. We present the main contribution of this work in Section IV and design multi-party topologies. In Section V the performance of HPNs is evaluated using real data of from LN. Finally, we conclude and present future directions in Section VII.

II. BACKGROUND

A. L2 Two-Party channels

To establish a payment channel, users utilize a message-passing protocol to open and update channels. The funding transaction of a channel is the initial transaction (performed on-chain) generated between two users and it determines the initial channel capacities. This transaction comes with a timeout so that one of the users can publish the transaction on the blockchain to recover funds. This way, users can perform peer-to-peer transactions rapidly without updating the blockchain for every transaction. In a refund transaction, both users agree upon a new channel balance. A refund transaction either directly invalidates previous refund transactions (Duplex Micropayment Channels [10]) or has a smaller timelock value than the previous refund transaction (Eltoo [11] and Lightning [12] channels), allowing users to publish it on the blockchain earlier, making the previous refund transaction unspendable. Refund transactions move the balance from the channel capacity of one user towards that of the other. Transfers are available as long as both capacities are greater than zero. Creating a payment channel can be expensive as it requires an on-chain transaction as well as locking money in the channel. A user can establish a transaction with another user even if they are not connected by a direct channel but through a path of multiple channels involving other intermediary users. In these cases *fees have to be paid* by the user establishing the transaction. Eventually – when users want to

close the channel – they publish the latest refund transaction along with the capacity balance of each user.

1) *Success Rate Issues:* Although payment channels can possibly increase *tps* and lead to reduced transaction fees there are caveats to pay attention to. As shown in [13], even if a payment network operates perfectly – no hardware ever fails, every node has full information, transactions resolve instantaneously – no payment network can process every single transaction. The authors of [13] argue that the transaction success rate of a realistic PCN is much lower than 100%. The main reason for an L2 transaction to fail can be the lack of a path with sufficient capacity between transaction endpoints. Even though in a large network such as LN there might be paths between users, in many cases, there is no available path due to channel unbalancing. When one side of a bidirectional channel is utilized more than the other, the channel becomes unbalanced. Should a transaction arrive in the opposite direction, this unbalanced state gets slightly mitigated [8]. Channel unbalancing can be an issue in Layer-2 payment networks. Ways to alleviate this problem are being investigated [14], [15], [16]. It has been demonstrated that channel unbalancing can lead to increased fees [17], [18]. Although a higher capacity does not directly affect channel unbalancing, a higher capacity locked on a channel increases the chances of a transaction to succeed. According to a previous work [15] on the LN network nearly 63% of the channels have depleted 80% of their capacity in one direction. The simulation results in the same work have demonstrated that the success rate in LN is approximately 65%. In Section V-B, our simulation results on LN yields a similar success rate (68%).

2) *Extension of Two-Party Channels: Virtual Channels* can be used to extend traditional two-party channels. These are channels built upon existing payment channels and allow users without a direct payment channel to perform payments. In Perun [19], when two users (A and B) want to create a virtual channel using a common intermediary (C) they both have a channel with, A and B can lock funds to form channels with C, and this creates a virtual channel. Once a virtual channel is set up, A and B can perform transactions directly, without any assistance from a third party.

B. Rollups

Rollups have shown explosive growth since their inception. Various rollup solutions have been developed and deployed on the Ethereum network. On certain days in 2022, the TVL in optimistic and zk-rollups exceeded \$4.5B (~ 200 times the TVL of LN). Rollups, take on a completely different approach when compared with payment channels by introducing the concept of rollup operators. They create a single giant channel that all peers can join. To join this giant channel, peers are requested to lock funds. This serves as an incentive to follow the protocol. While zk-SNARK (zero-knowledge Succinct Non-interactive Argument of Knowledge) [20], [21], [22] based zk-rollups are computationally demanding, as they publish a validity proof along with the state change optimistic rollups skip this step and the state is assumed to be valid by default and for a predefined amount of time the state change

can be challenged. zk-SNARKs are found to be computable in a distributed manner [23] and further adoption is anticipated.

C. Multi-Party Channels

To improve upon the performance of two-party payment channels, various studies have explored the potential of multi-party payment channels. Here we list some of these solutions.

1) *Channel Factories:* In this framework, n participants collaboratively lock a specific amount of digital asset in a funding transaction. This collaboration allows them to execute transactions off-chain peer-to-peer. To validate a transaction, signatures from $\lfloor \frac{n}{2} \rfloor + 1$ nodes in the channel are required. Channel factories were initially proposed by Burchert et al. [24] and were further improved to give rise to Duplex Micropayment Channels [10]. Representative variations include Eltoo [11], which facilitates state updates by invalidating prior transactions and Lightning Channels [2], which incorporates additional mechanisms to deter fraudulent activities. Pedrosa et al. [25] adopt these Lightning Channels to design channel factories.

2) *Sharding:* Sharding as a scalability solution has recently gained attention. Within this concept, transactions are split up into distinct *shards* each of which is managed by appointed group of nodes. Elastico [29] is a pioneering protocol that addressed the problem of off-chain transaction processing and the concept of *intra-shard consensus* was introduced. Initially, the allocation of nodes to shards was achieved via a pseudo-random process¹ [30], [31]. Later, Monoxide [32] introduced *static sharding*, in which the the allocation of nodes to shards is fixed. A central challenge in sharding [32], [33], [34], [29], [31] is effectively handling transactions that involve nodes from different shards. These are commonly referred to as *cross-shard transactions*. A promising approach is to encourage nodes to participate in more than one shard, acting as relay nodes or *broker accounts* for these cross-shard transactions. Using our terminology, a cross-shard transaction can be considered a two-hop path in a hypergraph, where each shard constitutes a hyperedge.

3) *Payment Hubs:* These are also multiplexed payment channels, using a centralized entity to enable transactions. In NOCUST [26], an off-chain server serves as an n -party hub through which users execute their transactions. This architecture, however, presents a single point of failure due to reliance on a third-party server. To mitigate issue, alternative solutions like Gnocchi [27] employ a rotating supervisor S , elected through consensus among hub members. This supervisor validates and then broadcasts each transaction to all hub participants. Similarly, Garou [28] designates a supervisor S for each round to sequence transactions. Unlike NOCUST and Garou (intra-hub transactions), Gnocchi enables inter-hub transactions by managing active participants in multiple hubs. A comparison of various payment hub architectures is presented in Table II.

¹Nodes are required to solve a puzzle, and the resulting last few bits dictate the shard assignment.

Table II: A comparison between different payment hub designs.

	NOCUST [26]	Gnocchi [27]	Garou [28]
Topology	Star	Dynamic Star	Dynamic Star
Third-Party	Constant Operator Server	Rotating Supervisor	Rotating Supervisor
Who broadcasts the message	Operator Server	Supervisor	Users

D. Multi-Party Channel Security

Given the critical importance of security in blockchain systems, papers that present a multi-party solution address the implementation’s security. Below, we briefly review the security aspects of several of these solutions.

Gnocchi. In Gnocchi, all transactions are performed via a leader. Assume Alice, Bob and Carol are part of a multi-party channel, where Alice wants to move funds to Bob and Carol is the leader. First, Alice and Bob sign the transaction details and send it to Carol for approval. Once Carol approves the transaction, she creates a new state representing the new amount of funds Alice and Bob have, and both need to sign on the new state. Only then, Carol can distribute the new state to all channel participants. In Gnocchi, in case any of the channel participants (including the leader) misbehave, any user can publish a Fraud-Proof and a penalty is enforced on the malicious user [27].

This implementation guarantees several attributes. First, as Alice and Bob sign the transaction and the new state with their private keys, neither the leader nor any other user can publish a state instead of them. Moreover, when a non-leader user is disconnected (or refuses to cooperate), the only transactions that may be affected are transactions performed directly with this user (similar to two-party-channels). In case the leader disconnects, transactions cannot be performed. Yet, as the leader is not a permanent user rather it is a rotating role, leader disconnecting affects users only for a small period of time.

Garou. In Garou, a rotating leader is used too, where each leader remains the leader for a round (some time epoch). In this implementation, transactions performed during some round can be used only in a later round. Now, when Alice and Bob want to transfer funds, they turn to Carol (the leader) to get a unique transaction ID. This allows dealing with double spending attempts. Using this ID, Alice and Bob create the transaction and each user signs it with their private key. Carol validates the transaction is valid, and signs the new transaction. Now, the transaction which is signed by the private keys of Alice, Bob and Carol can be distributed to users [28].

Similarly to Gnocchi, due to the use of private keys, no user can send transactions instead of another user. Additionally, when non-users disconnect, they affect only transactions they are directly involved in, and if the leader disconnects it affects only the current round. Finally, funds modified in a transaction of some round cannot be used later in that round, assuring all users have enough funds for the operations they perform. Gnocchi additionally makes use of HTLCs for multi-hop transactions, providing security in multi-hop transactions similar to Lightning.

In this paper, we use multi-party channels to create an efficient layer-2 network. Our algorithms and network designs do not change the implementation of the channels, rather they deal

with the network structure and how to optimize it. Therefore, we assume multi-party channel implementations provide the needed security, and focus on the network topology, improving transaction success rate and reducing fees.

E. Hypergraphs in Graph Theory

Traditionally, in a graph an edge connects two vertices. Such graphs are also known as ordinary graphs. A hypergraph is a generalization of a graph in which an edge can join any number of vertices. Formally, a hypergraph is represented as a pair (V, E) of vertices and hyperedges such that an edge refers to a set of vertices of arbitrary size such that $\forall e \in E$ it holds that $e \subseteq V$. We can refer to an edge as an ordered set of vertices so that the hypergraph is directed. On the other hand, in an undirected graph, we view an edge as an (unordered) set of vertices. In d -regular hypergraphs, every vertex has degree d and is included in d hyperedges.

Hypergraphs are often used to model relationships between multiple entities. In network systems, hyperedges are useful for modeling multicast or broadcast communication. In such a system, a message is sent simultaneously from one entity to multiple entities. A hyperedge can represent a single communication that involves multiple recipients, making it easier to handle group communications [35]. Hypergraphs are widely used in blockchain systems as well. For instance, [36] proposes a hypergraph-based adaptive consortium blockchain (HARB) framework to coordinate Distributed Energy Resources (DERs) in peer-to-peer decentralized energy trading. Specifically, they use hypergraphs to represent complex relationships between resources and end users. In [37], hypergraphs are used to organize storage nodes in a hypergraph-based blockchain model for Internet of Things (IoT)-enabled smart homes. In this paper, we focus on undirected hypergraphs and often consider d -regular for various d values such as 3, 20 and 50. We use hypergraphs to present a payment channel between multiple blockchain users, improving the network performance and allowing the decrease of network fees.

F. Related Work

There have been several proposals in the scientific literature to improve upon the scalability properties of blockchain networks. Flow in [38] for instance is trying to decouple the consensus algorithm from the actual computation happening onchain. PCNs are also promising solutions to the same issue, and among them, LN has gained popularity in recent years and these days even resource-constrained IoT devices [39] can use the PCN. Since the advent of LN, some work has been devoted to improving the protocol. Sprites [40] reduces the cost incurred by escrowing funds in $|E|$ channels for w_d time from $\mathcal{O}(|E| \cdot w_d)$ to $\mathcal{O}(|E| + w_d)$. Teechain [41]

for example proposes to use trusted execution environments (TEE) and committee chains to achieve asynchronous chain access with improved transaction throughput. The issue of channel rebalancing [42], [7] to improve user experience in the network is also actively researched, and similarly, Flash [43] and Spider [44] try to increase the transaction success rate by mitigating the chances of the channels becoming unbalanced. Efficient payment routing strategies in Layer-2 payment networks are also being investigated [8].

Some works were aimed at clarifying whether the L2 payment networks are truly decentralized. In [45] the authors theorise about the Gini coefficient of the centrality measures of the nodes and find that there are heavy centralization trends in the network. To uncover other types of vulnerabilities in the network there are works devoted to privacy-related investigations. One of the main issues from privacy [46] perspectives is the visibility of channel balances [47]. A minute analysis of the problem, for example is given in [48]. In [49] Nisslmueller et al. define a probing attack to expose channel capacities and a timing attack to uncover distances in the network that would enable an attacker to analyze network states. Further enumerating the potential vulnerabilities, the authors of [50] find that a small fraction of the nodes can deny service to a large fraction of the network by hijacking transaction routes. Other works have shown that node isolation attacks and channel exhaustion [51], [52] can also have devastating consequences.

III. L2 CHANNELS AND FINANCES

Here we first review some of the latest scientific works on the financial aspects of operating L2 payment channels to motivate our cost model and to give some background to our mathematical formalism based on previous investigations. After this review, the mathematical model of the cost function in an HPN is given.

A. Financial Incentives

The financial aspects of payment channels have also been extensively researched. In [53], for example the authors propose the necessary algorithms to optimize routing rewards when placing an LN node. Béres et al. in [54] estimate such revenue from transaction fees and they supplement their work with a traffic simulator for LN. Similar work has been documented in [55] to uncover the peculiarities in the demand market once LN grows substantially large and to elucidate the effect of increased adoption in the network on the price of the native token. Methods to optimize graph structures for Payment Service Provider (PSP) for profit maximization are proposed by Avarikioti et al. in [56] and DMC for PSPs are proposed in [10]. Furthermore in [57] the authors give a detailed overview of how Bitcoin transaction networks including LN have evolved since the genesis block of the Bitcoin blockchain intending to find the correlation between the price of the chain's native token and the network parameters. A thorough discussion with both analytic formulas and simulation results in a similar topic is given in [12] to elucidate the emergence of a connected component in the network.

B. HPN Cost Model

We now confine our attention to multiplexed payment channel solutions and evaluate the concomitant operating cost. Taking already existing implementation as a basis we create hyperedge topologies (IV). To estimate the cost of an HPN a generic cost function is defined as an admixture of 3 terms:

- w_c The fixed cost of a hyperedge (L1 transaction)
- w_d The interest paid to nodes on the escrowed funds
- w_s The signaling overhead for each transaction in the hyperedges

The exact values of w_c , w_d and w_s can be approximated by analyzing the underlying blockchain technology, energy prices and other economic factors. The total cost of running an HPN for a given period of time in our model is:

$$w_c \cdot s_c + w_d \cdot s_d + w_s \cdot s_s , \quad (1)$$

where s_c is the total cost of setting up the channels, s_d is the cumulative amount of deposits and s_s is the total cost of signaling and cryptographically signing messages of transactions.

The total cost of setting up the channels is

$$s_c = \sum_{\forall v \in V} |N(v)| , \quad (2)$$

where $|N(v)|$ is the number of hyperedges (clusters) node v is adjacent in the HPN topology. In our model, the cost of opening a channel is simply a transaction fee on the L1 chain. Signaling costs (s_s) can be calculated from the set of transactions T initiated in a certain time interval (1 year in our case). Additionally, $H(t)$ represents the number of channels (path length) used when executing transaction t . For each transaction $t \in T$ the set of hyperedges traversed by the transaction is denoted by E_t . The signaling cost then becomes:

$$s_s = \sum_{\forall t \in T} H(t) \cdot \sum_{\forall e \in E_t} |e| , \quad (3)$$

where $|e|$ denotes the size (cardinality) of the hyperedge e . Note that in the original LN implementation $|e| = 2$ (channels with 2 endpoints). For more users in the hyperedge a distributed protocol is needed to track state changes. In this case, hyperedge e with $|e|$ parties incurs a signaling overhead of $w_s \cdot |e|$ for each transaction that has taken place over the edge. Users operating nodes in the HPN should be compensated for the opportunity cost of not earning interest on their deposits. The interest rate (w_d) multiplied by s_d is the amount hypothetically paid by the system to node operators. Let $c_{v,e}$ denote the amount of cryptocurrency deposited by node v in hyperedge e . Then

$$s_d = \sum_{v \in V} \sum_{e \in N(v)} d_{v,e} , \quad (4)$$

where $e \in N(v)$ denotes the set of hyperedges containing node v .

C. Evaluating coefficient costs

This section analyzes Bitcoin data to understand the costs of creating and operating a multi-party-channel network. We evaluated the construction costs which depend on the price of executing layer-1 transactions, and the transaction costs which depend on the fees used by the channels. Since Bitcoin has an existing large PCN network, Lightning, while Raiden (Ethereum’s PCN) is rarely used, we perform our cost analysis on Bitcoin data. The evaluated values will be used later in Section V.

1) *Construction Fee*: In Bitcoin, the transaction fee comprises the transaction size and the fee rate, such that $Fee = transaction_size \cdot fee_rate$. A Bitcoin transaction comprises the following fields: version, inputs, outputs and locktime. While the version and locktime are usually 4 bytes long, each input and output is usually of size 100-150 bytes for SegWit transactions (a storage-efficient transaction) and 150-200 bytes for non-SegWit transactions. This means that the more inputs (and outputs) a transaction consists of, the larger it is, and accordingly the fee charged for publishing it increases. Ethereum has a similar fee model, using Gas (unit of measurement for the computational effort required to execute a transaction). The transaction fee is evaluated based on the Gas used and the Gas price (fee): $Fee = Gas_used \cdot Gas_price$.

Based on data gathered from Mempool², a blockchain explorer tool, the average and median transaction rate fees were around 10 Satoshi (SAT) per byte. 1 SAT is one hundred millionths of a bitcoin. For instance, as the average transaction size was around 300 bytes, the fee for performing a Bitcoin transaction is around 3000 SAT per transaction. Additionally, adding an input or output to a transaction usually increases the fee by 1000-2000 SAT.

2) *Transaction Fee*: When executing a PCN transaction, intermediate nodes participating in the transaction transmission are rewarded with a fee. In Lightning, intermediate nodes charge a fee of the form $Fee = rate_fee \cdot tx_size + base_fee$ [58]. In Raiden, the fee function additionally includes an imbalance fee that increases when the transaction exhausts channel liquidity, trying to keep the channel balanced [59]. We perform an analysis of Lightning’s state during December 2022.

Fig. 2 presents the histogram of the base fees used in the network. The majority of channels did not use any base fee (46%), or used a base fee of 100 SAT (40%). Fig. 3 presents the histogram of the rate fees used in the network. Most of the channels used rate fees in the range of 1-100 SAT, with 34% of the channels using a rate fee of 1 SAT, 23% with 10 SAT and 29% using 100 SAT. Fig. 4 presents the CDF (cumulative distribution function) of the total transaction fee for one channel, for transactions of size 10^3 , 10^4 , 10^5 and 10^6 SAT. For all four transaction sizes, 20% of the channels charge a fee of 1 SAT or less. Additionally, no more than 1% of the channels charge a fee higher than 6000 SAT. The average and median fee charged by a channel depends on the transaction

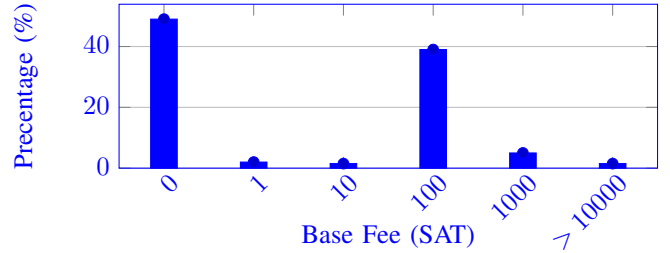


Figure 2: Lightning base fee histogram. The base fee of a transaction is fixed regardless of the size of the payment.

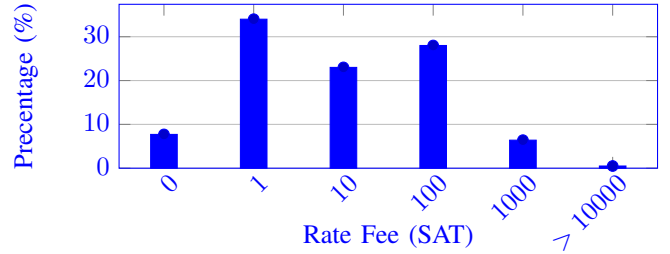


Figure 3: Lightning rate fee histogram. The rate fee is multiplied by the transaction size when evaluating the total fee.

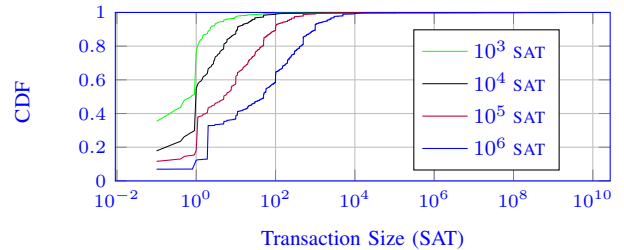


Figure 4: Transaction fee CDF, for transactions of size 10^3 , 10^4 , 10^5 and 10^6 SAT.

size, though the majority of channels charge a fee between 1 and 100 SAT.

Finally, we evaluate the annual number of transactions. A research report performed by River, a regulated financial institution, estimated the number of lightning transactions performed during August 2023 was around 6 million [60]. Hence, in the evaluation section, we simulate 10^8 transactions per year.

IV. CREATING HYPEREDGES IN PCNS

In traditional PCNs peers open channels between one another by escrowing digital assets into multisig addresses. This structure can be greatly improved upon by not limiting the constructions to 1-to-1 channels. One can design a structure with more degrees of freedom by enabling n -to- n channels in payment networks. A handful of proposals to implement n -to- n channels in payment networks were enumerated in Section II-C.

A hyperedge e comprises a set of nodes V_e and each node $v \in V_e$ has a capacity $c_{e,v}$ that is the amount of cryptocurrency locked in the blockchain by node v when entering the channel

²<https://mempool.space/>

e . The capacity of hyperedge e is denoted by c_e and we express it as $c_e = \sum_{v \in V_e} c_{e,v}$. Note that the capacity of the hyperedge remains the same as long as the channel is open and not updated. However, the values of $c_{e,v}$ (node capacities) change after each transaction. Each node v involved in a hyperedge e can initiate transactions as long as its capacity remains non-negative, i.e. $c_{e,v} \geq 0$. The two extreme solutions are (1) if every edge has two end nodes as in LN or (2) when the network is composed of a single giant hyperedge with all the nodes involved. We are also interested in the in-between solutions for a given network with hyperedges of certain sizes. We estimate the hypergraph topology by analyzing the LN topology and replacing a set of edges with a hyperedge. The edges of the LN topology are treated as relationships between the parties which should then be inherited in the hypergraph topology. The following rules of thumb are used to construct a hypergraph from an LN topology. Our principal goal is to have hyperedge e with a set of nodes V_e

- spanning a connected subgraph in the LN topology,
- with cardinality at most m_{max} ,
- spanning a subgraph in the LN topology with a small diameter,
- ideally, span a subgraph in the LN topology with high connectivity (e.g., a clique).

We refer to this process as clustering the edges into hyperedges. Fig. 1 shows an example of this procedure, in which we take the original LN topology $G = (V, E)$ as a reference (Fig. 1 top), and replace the edges with hyperedges (Fig. 1 bottom) to get 5 different, albeit overlapping hyperedges (e_1, \dots, e_5) in hypergraph $H = (V, \mathcal{E})$. For example, hyperedge e_1 is understood as covers by nodes $\{v_7, v_4, v_3\}$ composing LN edges (v_7, v_4) and (v_4, v_3) . Assuming the edge capacities were 2 BTC in LN and they were split equally between its end nodes, the capacity of e_1 is 4 BTC, where node v_1 locked $c_{e_1, v_1} = 2$ BTC and v_7 and v_3 locked 1 BTC when establishing the channel.

A. HPN Topologies

Commencing from the original LN topology we develop methods to restructure the network to improve performance. In Section V we evaluate these topologies. Below we introduce these topologies in great detail.

1) *LN Original Topology*: The original topology is taken as a snapshot of LN. In this topology the majority of nodes were connected in one major connected component. Only channels with active status were used. While channel capacities are known, directed capacities are unknown. In our calculations we use equal capacities in both directions.

2) *LNrollup*: This is taken as a hypergraph composed of a single hyperedge covering every node of the network.

3) *SuperNode Topology (SN)*: Wu and Jiang in [61] pursued a similar aim and proposed a PCN based on a handful of main nodes (supernodes) holding most of the network capacity. Each user is connected to one supernode and supernodes with regular-node neighbors, form clusters. Regular nodes are connected only to the supernode, and the supernodes are connected to each other. This model is motivated by the fact

that users might lock large amounts of capacity in the network, yet, since it is split between multiple channels, the capacity in each channel is smaller and the channel becomes unbalanced and unusable more frequently.

Here we redesign the LN topology based on the algorithms of Wu and Jiang to create an SN network from the LN network. Algorithm 1 is used for choosing supernodes. With the list of supernodes we create clusters with 1 supernode per cluster. In SN, each cluster represents a multi-party channel. Edges of non-supernode nodes are removed, and edges between supernodes are treated as bi-directional channels between clusters.

Algorithm 1 works as follows. Each node u with a unique id $ID(u)$ is initialized as a supernode. The algorithm iterates over all nodes to remove them from the set of supernodes. If node u has two neighbors s, t such that there is an edge $s-t$ connecting them, we remove u from the list of supernodes. u also holds a local 2-hop view (nodes of a distance of at most 2 edges from u). For some nodes u some neighbors, s and t might be connected using a path from the local 2-hop view of u with x on the path $ID(x) > ID(u)$. If this be the case, we remove u from the list of supernodes. Upon termination, every node is treated either as a supernode or as a neighbor of a supernode.

Since the original LN topology can be computationally challenging to handle we restricted paths to be of at most 4 edges when checking if node u should be removed when running the algorithm. At the end of the algorithm, every node u , that was not marked as a supernode is assigned to a supernode (randomly chosen between u 's supernode neighbors if there are more than one cluster for u). From this point, the total capacity on u 's edges in the original graph is locked in the multi-party channel of this cluster. For a supernode v , v is part of the cluster that was constructed by adding its neighbors. The capacity of v in edges connecting other clusters is the same as in the original network. The capacity of v in a multi-party channel represented by the cluster is calculated as the sum of capacities locked on the corresponding edges in the original network between v and the nodes in the respective cluster.

Lemma 1. *The complexity of Algorithm. 1 is $\mathcal{O}(|V|^3 \cdot 2^{|V|})$.*

Proof. The loop in line 2 is performed $|V|$ times. Since each node has at most $|V|$ neighbors, the loop in line 4 is executed at most $|V|$ times. The loop in line 7 is iterated at most $|V|^2$ times. The if statement in line 11 requires going over all possible paths between u and v , the complexity of this operation is $\mathcal{O}(2^{|V|})$. Hence, the total complexity of Algorithm. 1 is $\mathcal{O}(|V|^3 \cdot 2^{|V|})$. \square

4) *Node Cover Hypergraph (NCH)*: A hyperedge graph is composed of hyperedges instead of channels. Each hyperedge represents a multi-party channel, where users have capacities in the hyperedges they belong to. Direct payments are performed via the multi-party channel (i.e. inside the hyperedge) and multi-hop payments (payments between users located in different hyperedge) are performed through users participating in the two respective hyperedges. Given a PCN network topology graph (e.g. LN), an NCH graph can be created as follows.

Algorithm 1 Select supernodes

Input: network topology undirected graph $G = (V, E)$ **Output:** list of supernodes S **Complexity:** $\mathcal{O}(|V|^3 \cdot 2^{|V|})$

```

1:  $S = V$  ▷ initial list of supernodes
2: for  $n$  in  $V$  do
3:    $V_n = N_G(n)$  ▷ list of nodes within 2 hops
4:   for  $u$  in  $N_G(n)$  do
5:      $V_n := V_n \cup N_G(u)$ 
6:    $E_n = \{(u, v) | u, v \in V_n \text{ and } (u, v) \in E\}$ 
7:   for  $u, v$  in  $N_G(n)$  do
8:     if  $(u, v) \in E$  then
9:        $S := S \setminus \{n\}$ 
10:    else ▷  $ID()$  is the unique id of the node
11:      if  $\exists P = u - u_2 - \dots - u_{l-1} - v,$ 
12:        s.t  $ID(u_2) > ID(u_3) > \dots > ID(u_{l-1})$ 
13:        and  $\forall u_i \in P : (u_i, u_{i+1}) \in E_n$  then
14:           $S := S \setminus \{n\}$ 
15: return  $S$ 

```

First, an approximated minimal vertex cover is computed. Then, for each node in the vertex cover, all neighbors are added to a group and a hyperedge is created. The total capacity of each node in the original network is distributed evenly between all hyperedges associated with the node. We note that the intuition behind our approach is to set up hyperedges between nodes that were adjacent in the original LN topology. We assume that these edges were originally created due to their advantages, meaning many payments passed through them. When selecting the end nodes of the hyperedges, we aim to preserve this property so that as many payments as possible are direct transactions between users. NCH graphs were created using Algorithm 2. Given a regular PCN network G (e.g. the original LN topology), this algorithm computes an approximated minimal vertex cover. using NetworkX's `min_weighted_vertex_cover` function [62]. Note that every step of Algorithm 2 is deterministic, and assuming that every participant is aware of the same topology, it can be run in a decentralized network with no trust assumption whatsoever.

Lemma 2. *The complexity of Algorithm 2 is $\mathcal{O}(|E| \log |V|)$.*

Proof. It may on the face of it seem impossible to compute an ideal vertex cover (being NP-hard), but approximations can be achieved at the cost of $\mathcal{O}(|E| + |V|)$ [63] or $\mathcal{O}(|E| \log |V|)$ [64]. Therefore line 2 can be run for either $\mathcal{O}(|E| + |V|)$ or $\mathcal{O}(|E| \log |V|)$. As $V_C \subseteq V$, the loop in line 3 is performed at most $|V|$ times, and as each node has at most $|V|$ neighbors, the complexity of lines 3-5 is $\mathcal{O}(|V| \log |V|)$. Therefore, assuming $|E| \geq |V|$, the complexity of Algorithm 2 is $\mathcal{O}(|E| \log |V|)$. \square

5) *Fixed Hyperedge Size (FHS):* As in an NCH topology, the hyperedges are not necessarily of the same size and the algorithm imposes no restrictions on the allowed hyperedge size. Given a PCN network G and a parameter indicating the maximal hyperedge size m_{max} , Algorithm 3 presents an algorithm transforming a regular PCN network to a hyperedge

Algorithm 2 Create NCH

Input: network topology graph $G = (V, E)$ **Output:** hypergraph $H = (V, \mathcal{E})$ **Complexity:** $\mathcal{O}(|E| \log |V|)$

```

1: Initialize a hypergraph  $H$  with node set  $V$ , and  $\mathcal{E} = \emptyset$ 
2: Compute  $G$ 's vertex cover:  $V_C$ 
3: for  $c$  in  $V_C$  do
4:   The neighbors of node  $c$  in  $G$  is  $V_e$ 
5:   Add a new hyperedge to nodes  $V_e$  to  $H$ 
6: return  $H$ 

```

network with a restriction on the maximal hyperedge size. At each iteration, the algorithm finds the node with the highest degree in G and its m_{max} closest neighbors. It then creates a hyperedge from these nodes and removes inner-hyperedge edges from G . It also removes nodes with degree 0. The algorithm returns a hyperedge network with a constraint on the maximal hyperedge size m_{max} . This allows one to control signaling costs.

Lemma 3. *The complexity of Algorithm 3 is $\mathcal{O}(|E| \cdot (|E| + |V|))$.*

Proof. At each iteration, at least one edge is removed and there are at most $|E|$ iterations. Finding nodes with the highest degree is performed in $\mathcal{O}(|V|)$ steps. Running the breadth first search (BFS) algorithm at line 3 is performed in $\mathcal{O}(|E| + m_{max})$ steps. Lines 5-7 take (for all iterations) $\mathcal{O}(|E| + |V|)$ operations in total. m_{max} is the maximum number of nodes in each hyperedge. As such, it must be smaller than $|V|$, therefore the complexity of Algorithm 3 is $\mathcal{O}(|E| \cdot (|E| + |V|))$. \square

Algorithm 3 Generate FHS

Input: network topology G , max hyperedge size m_{max} **Output:** hypergraph $H = (V, \mathcal{E})$ **Complexity:** $\mathcal{O}(|E| \cdot (|E| + |V|))$

```

1: Initialize a hypergraph  $H$  with node set  $V$ , and  $\mathcal{E} = \emptyset$ 
2: while  $G$  is not empty do
3:   Find the node  $v$  with the highest degree
4:   Run BFS visiting  $m_{max}$  nodes in  $G$ , denoted by  $V_e$ 
5:   Remove the edges between  $V_e \in G$ .
6:   Remove the nodes which became isolated in  $G$ .
7:   Add a new hyperedge among nodes  $V_e$  to  $H$ 
8: return  $H$ 

```

Table III: The parameters of the the LN transaction generator [54] used in our evaluations. for LN.

Node alias	Capacity rank	Channel rank	Payment target ratio (%)
In.nicehash.com [Nicehash]	8	5	9.64
CoinGate	39	4	6.15
1ML.com	191	3	4.23
Moon (paywithmoon.com)	27	21	3.62
Lightning.Watch	237	27	2.38
The Captain [Coincept.com]	83	71	1.90
Blixt Wallet	87	95	1.84
coinhodler04108	54	93	1.72
Narodowy Bank Polski	140	67	1.54
Isats.com	17	70	1.53

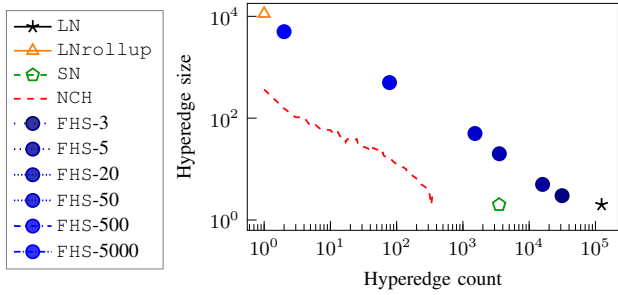


Figure 5: Hyperedge Sizes Distribution.

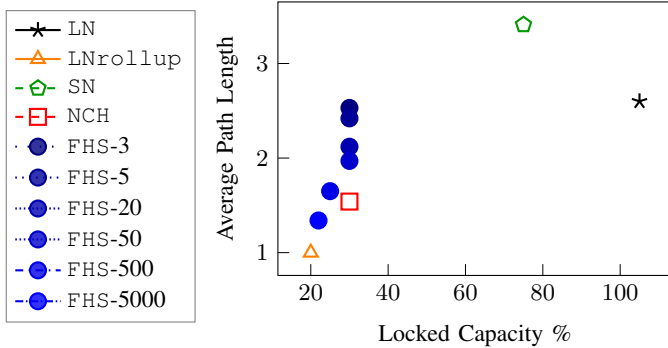


Figure 6: The percentage of the original capacity needed to be locked to achieve a success rate will reach exactly 70%, and the average path length in that case.

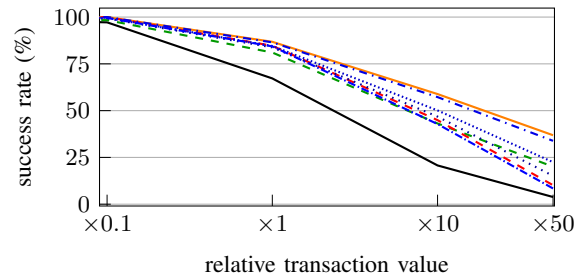
V. EVALUATION

In this section, we evaluate the performance of the topologies from Section IV:

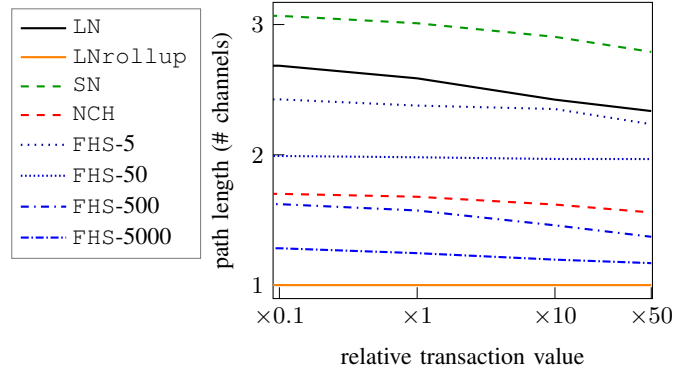
- 1) original LN topology (LN),
- 2) LNrollup topology,
- 3) Supernode Topology (SN),
- 4) Node Cover Hypergraph (NCH),
- 5) Fixed Hyperedge Size (FHS- x), x is the number of nodes of each hyperedge.

Fig. 5 shows a histogram of the edge sizes in these topologies. In the case of LN and SN we use bidirectional channels, which are equivalent to hyperedges of size 2. For LNrollup and FHS we use constant-sized hyperedges too and for the NCH topology we use hyperedges of varying sizes, having the majority of hyperedges containing less than 50 nodes. In the case of the NCH topology we rely on a few large hyperedges and the largest hyperedge contains 350 nodes.

We follow previous works [61], [65], [66] and use the transaction success rate and the average hop size as our evaluation metrics. The network evaluation is done by serving generated transactions (see Section V-A). Accordingly, throughout the paper, we refer to transactions of size 60000 SAT. In Fig. 7a, though, we evaluate the topologies' success rate for various transaction sizes derived by multiplying their original transaction size by factors of $\times 0.01$, $\times 0.1$, $\times 10$, $\times 50$.



(a) Success rate



(b) Average path length

Figure 7: A comparison of the success rate and the average path length compared to the size of the transactions.

A. Simulations with Empirical Data

Simulating real-life transactions in LN is no easy task as currently there is no way to pry into the actual transaction amounts on the edges between two parties. We surmount this difficulty by running simulations and we approximate the cost functions. To this end, we make use of the LN transaction generator presented in [54], using an updated merchant list for 2022 to select transaction endpoints. For reproducibility (and for a more detailed explanation), we publish the traffic traces and LN topology in a GitHub repository³ which will be replaced with our original repository in the final version of the paper. We proceed analogously as described in [54] to utilize the 1ML LN Search and Analysis Engine⁴. By doing so we found 302 merchant nodes also present in our LN data. We sample 80% of payment target nodes using a biased distribution skewed towards merchants with probability proportional to their degree of importance to account for the flow of payments in the network in the direction of service providers. Payment sources and the remaining 20% of target nodes are sampled uniformly at random for the simulation. Table III shows the ten most frequently sampled payment target nodes in our simulations. For faster evaluation, we evaluated traces of 10^4 transactions and multiplied the respective cost by 10^5 to obtain s_d for 10^8 transactions.

Table IV: Performance and cost (in USD) of each topology for running the L2 network of 10^8 transactions with cost parameters $w_s = 10^{-4}$, $w_c = 3$ and $w_d = 0.2$ USD.

topology name	avg node degree	hyper-edge size	avg path length $H(t)$	locked capacity [%]	construction cost ($s_c \cdot 3$)	transact. cost ($s_s \cdot H(t) \cdot 10^{-6}$)	interest paid ($s_d \cdot 0.2$)	total cost [USD]
LN	10.99	2	2.6	105	743 592	159 328	2 456 224	3 359 144
LNrollup	1	11 268	1	21	67 608	151 051 300	491 246	151 610 154
SN	1.25	2	3.41	75	84 528	279 961	1 754 444	2 118 933
NCH	9.61	27	1.54	30	649 836	1 590 604	701 776	2 942 216
FHS-3	8.36	3	2.53	30	565 320	269 343	701 776	1 536 439
FHS-5	7.06	5	2.42	30	477 180	336 283	701 776	1 515 239
FHS-20	6.38	20	2.21	30	429 828	1 314 419	701 776	2 446 023
FHS-50	6.7	50	1.97	30	452 340	2 729 474	701 776	3 883 590
FHS-500	3.64	497	1.65	25	246 000	19 437 198	584 814	20 268 012
FHS-5000	1.34	4 674	1.34	22	90 000	117 461 371	514 638	118 066 009

B. Success Rate

Fig. 7 presents a comparison between the success rate, average hop number and network performance for different transaction sizes. From Fig. 7a, we conclude that LN performs with the worst success rate (only 67.21%). The success rates for transactions of size $\times 10$ and $\times 50$ were 20% and 3%, respectively, less than half of the success rate of NCH. LNrollup (and similarly FHS-5000) yields the best success rate (86%). When scaling the transaction amounts to $\times 10$ the success rate decreases to 58% and after scaling the transaction to $\times 50$ it drops to 36%. The SN topology performs with a slightly lower success rate than the NCH topology, except for the $\times 50$ transactions for which it drops to 20%. Note for comparison that the success rate of NCH drops to 10%. The NCH topology produces similar results to the FHS-50 topology, except for the case of $\times 50$ of the transaction amounts for which the FHS-50 performs with a success rate of 22% while NCH does the same with a success rate of 8.4%. Fig. 7b presents the average path length of transactions for different topologies. With increasing hyperedge sizes, one gets shorter paths. The paths are the longest in LN and SN topologies, on average ≈ 2.4 and 2.95 edges. For the LNrollup topology, the average path length was 1 as each node had a direct path to all other nodes. The average path length of the NCH topology is ≈ 1.55 , a bit shorter than in the FHS-500 topology.

C. Reducing Locked Capacity

Recall that changes in the success rate must be imputed to the capacity locked by participants. More capacity allows the execution of higher-value transactions for longer times, especially when the channel is not balanced. Yet, higher capacities require users to lock more funds implying an opportunity cost as users could have used the funds elsewhere. According to Fig. 7, different topologies perform differently. Here, we were interested in the percentage of the original capacity of each node that has to be locked to achieve a success rate of 70%. Fig. 6 presents the original capacity locked when modifying the original capacity to achieve a success rate of 70% (x-axis) and the proper average path length (y-axis) in that case. We observe that LN requires the most amount of capacity locked,

105% of the original capacity, to reach a success rate of 70%. The average path length in this case is 2.6. Next, in the SN topology 75% of the original topology has to be locked, having an average path length of 3.41. For FHS topologies, the higher the hyperedge size is the less capacity is required. Observe that paths are shorter in this case. While FHS-3 requires 30% of the original capacity and has an average path length of 2.53, FHS-5000 requires only 22% of the original capacity and has an average path length of 1.34. NCH performs similarly to FHS topologies, requiring 30% of the original capacity to be locked while having a path length of 1.54. Finally, LNrollup requires the least capacity (21%) and it operates with the shortest average hop length (1). The above results underline the auspicious performance metrics of the proposed approaches when using both the NCH and the FHS topologies. These results are illuminating in a sense that in both cases we achieved *less amount of locked funds and we managed to decrease average path lengths while maintaining the same success rate as in the original LN topology.*

D. Cost Evaluation of Topologies

After evaluating the performance of the different topologies, we now discuss the cost of each topology, using the cost model presented in Section III-A. We assume the higher the cost of constructing and maintaining the network will be, the higher the fees users will need to pay. Hence, though some topologies (such as LNrollup) seem to appear very good, they might be very costly and thus unrealistic in practice. Recall that the exact w_s, w_c, w_d values can only be obtained by knowing the underlying blockchain technology, the energy prices and other economic factors. As such, we use an approximate cost model in our analysis.

Table IV summarizes our findings for the cost parameters $w_s = 10^{-6}$, $w_c = 3$ and $w_d = 0.2$ USD, when evaluating the costs on the evaluation performed in Section V-C. The cost parameters were driven from the analysis in Section III-C, showing the cost of constructing a payment channel increases by 1000-2000 SAT for each user. Using a BTC/USD exchange rate of 1 BTC = 63000 USD, the cost of a user per channel is around a few dollars per user. Hence, we use $w_c = 3$; Similarly, the transaction cost is around 10 SAT per transaction. Thus, we assume the transaction fee for using a payment channel is on a scale of around $w_s = 10^{-4}$ USD per channel.

³<https://github.com/iAradK/improving-Blockchain-Scalability-with-Hypergraph-Payment-Networks>

⁴<https://1ml.com/directory>

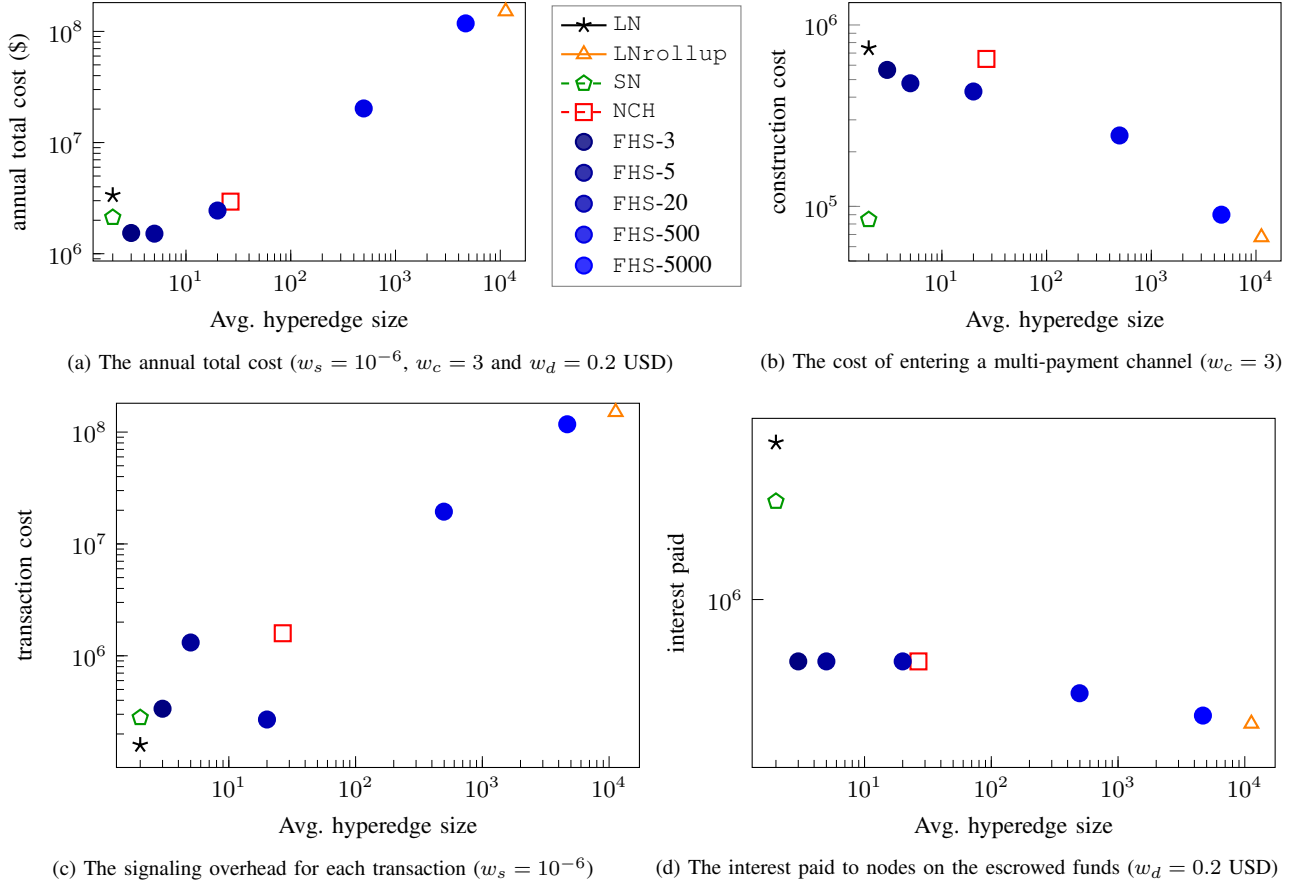


Figure 8: The cost of various payment networks vs. the hyperedge size after 10^8 transactions for a transaction success rate of 70%

All topologies achieve a success rate of 70%, showing multiple trends.

As LNrollup SN and FHS-5000 have a low node degree, each node appears in few hyperedges hence the construction cost is very low. In contrast, all other topologies have a relatively high construction cost. NCH and FHS topologies have low interest-cost since they require low capacity locked, while in the LN and SN topologies the interest-cost is very high. Additionally, as LN, SN, NCH and FHS-3-20 are composed of relatively small hyperedges, the transaction costs of these topologies are relatively small. Note that, LNrollup and FHS-5000 come with enormous transaction fees. According to our results, FHS-3 and FHS-5 achieved the lowest total, having costs 50% lower than LN. Additionally, SN, NCH and FHS-20 had fees lower than LN.

E. Cost Parameter Sensitivity Test

Table IV shows our findings for the different topologies in regards to cost parameters $w_s = 10^{-4}$, $w_c = 3$ and $w_d = 0.2$ USD. Though, when looking at the various cost components each implementation had its advantages and disadvantages. For instance, although the total cost of LN is higher than small-hyperedge-topologies, it has the lowest transaction cost. This might imply that the optimality of an implementation might rely on the cost parameters, as for a network with low

construction and interest fees with high transaction fees, LN might be considered.

To illustrate these trends Fig. 8 plots the cost metrics of the investigated topologies in Table IV. The average size of the hyperedges is measured on the horizontal axis. Three factors contribute to the cost: The construction cost s_c , transaction cost s_s and interest paid s_d . Two of these factors, the construction cost and the interest paid, decrease as the hyperedge size increases, while transaction costs increase with hyperedge size. Fig. 8b displays construction costs (the cost of 2 on-chain transactions) with respect to the hyperedge size. As the construction cost is the sum of the sizes of all hyperedges, it depends on the average node degree. In FHS, the general tendency is for larger hyperedge sizes the average node degree decreases: While for FHS-3 the node degree is 8.36, for FHS-5000 the average node degree decreases to only 1.34 and the construction fees are $5.65 \cdot 10^6$ and $9 \cdot 10^4$, accordingly. Moreover, LN and NCH have an average node degree of ~ 10 , and SN has a very low average node degree, 1.25 (resulting in a low construction cost of $8.4 \cdot 10^4$), since besides the supernodes every node appears only in one cluster.

For small hyperedges, one is faced with a capacity fragmentation problem. More specifically, more funds have to be locked in the channels to keep the same success rate, as seen in Fig. 8d, leading to a higher interest paid. We see LN and

SN need to lock almost 3 times more capacity than the other implementations.

Table V: Performance and the cost of running the L2 network with 10^8 transactions for cost parameters with lower interest paid fees: $w_s = 10^{-4}$, $w_c = 3$ and $w_d = 0.1$ USD.

topology name	construction ($s_c \cdot 3$)	transact. cost ($s_s \cdot 10^{-4}$)	interest paid ($s_d \cdot 0.1$) [%]	total cost [USD]
LN	743 592	159 328	1 228 112	2 131 032
LNrollup	67 608	151 051 300	245 624	151 364 532
SN	84 530	279 961	877 222	1 241 713
NCH	649 836	1 590 604	350 888	2 591 328
FHS-3	565 320	269 343	350 888	1 185 551
FHS-5	477 180	336 283	350 888	1 164 351
FHS-20	429 828	1 314 419	350 888	2 095 135
FHS-50	452 340	2 729 474	350 888	3 532 702
FHS-500	246 000	19 437 198	292 408	19 975 606
FHS-5000	90 000	117 461 371	257 320	117 808 691

Table VI: Performance and the cost of running the L2 network with 10^8 transactions for cost parameters with high transaction cost fees: $w_s = 10^{-4}$, $w_c = 6$ and $w_d = 0.2$ USD.

topology name	construction ($s_c \cdot 6$)	transact. cost ($s_s \cdot 10^{-4}$)	interest paid ($s_d \cdot 0.2$)	total cost [USD]
LN	1 487 184	159 328	2 456 224	4 102 736
LNrollup	135 216	151 051 300	491 246	151 677 762
SN	169 056	279 961	1 754 444	2 203 461
NCH	1 299 672	1 590 604	701 776	3 592 052
FHS-3	1 130 640	269 343	701 776	2 101 759
FHS-5	954 360	336 283	701 776	1 992 419
FHS-20	859 656	1 314 419	701 776	2 875 851
FHS-50	904 680	2 729 474	701 776	4 335 930
FHS-500	492 000	19 437 198	584 814	20 514 012
FHS-5000	180 000	117 461 371	514 638	118 156 009

On the other hand, for larger hyperedges, we have a higher signaling overhead, as more nodes must be informed when channel capacities change, as seen in Fig. 8d. We kept the rest of the factors fixed, however, we are aware of the fact that fluctuations in the exchange rate of the native coin can influence our results.

Since the parameter set affects the total cost of the implementation, Table V presents the costs of each topology when the interest paid is low, and Table VI presents the costs of each topology when the construction cost is high. Table V shows when the interest is low, NCH becomes more expensive than LN. FHS – 3 and FHS – 5 are still the cheapest. Similarly, when the construction cost is as high as in Table VI, FHS – 3 and FHS – 5 still performs the best, and NCH performs better than SN.

To summarize our findings, FHS – 3 and FHS – 5, NCH and SN can yield the same performance as LN for a much lower cost. Although SN and LN deliver low transaction fees and while FHS and NCH exhibit low interest paid as much less capacity is locked by the participants, in non-extreme cases FHS is still the most cost-efficient implementation.

VI. CHALLENGES AND FUTURE WORK

A. How to Add Nodes?

Previously in Section IV we described how NCH and FHS can be created from an existing L2 network. However, PCN networks must be dynamic as users join and leave the network. For this, we introduce an algorithm that can be used when new nodes want to join hypergraph-based payment networks utilizing our constructions. New nodes have to let each other

know of their intents and once the number of new nodes surpasses a certain threshold Algorithm 4 is executed. The

Algorithm 4 Add Nodes

Input: hypergraph H , list of nodes V , sequence number i , hyperedge size m

Output: hypergraph H' , sequence number i' , new nodes L'

Complexity: $\mathcal{O}(|V| \cdot \log |V|)$

- 1: Set $H' = H, V' = V, i' = i + 1$
 - 2: For each node $v \in V$, calculate: $h_v = Hash(v, i)$
 - 3: Sort L' by node hashes
 - 4: **while** $|L'| > m$ **do**
 - 5: Create hyperedge h from nodes L'_1, \dots, L'_m
 - 6: Add h to H'
 - 7: Remove L'_1, \dots, L'_m from L'
 - 8: **return** H', i' and L'
-

algorithm takes the old topology, a new list of nodes, the sequence number i (number of executions of the algorithm up to now) and the hyperedge size n as input. The sequence number i is used to introduce randomness to the algorithm so that the same nodes are not added to the same hyperedge at every execution. First, the algorithm calculates the hash of each node with the sequence number i . Then, the list of nodes is sorted by the hash values and for every group of n nodes the algorithm creates a hyperedge. The hyperedges are then added to the network. Finally, the algorithm returns the new hypergraph with the new hyperedges, the incremented sequential number (for future use when running the algorithm again) and a list of nodes that were not allocated to a hyperedge at this execution. Note that they might be added if the algorithm is executed again.

Lemma 4. *The complexity of Algorithm 4 is $\mathcal{O}(|V| \cdot \log |V|)$.*

Proof. Sorting $|V|$ nodes can be performed in $\mathcal{O}(|V| \cdot \log |V|)$. Line 4 of the algorithm is performed at most $|V|$ times, and every other operation of the algorithm is performed in $\mathcal{O}(1)$. Thus the complexity of Algorithm 4 is $\mathcal{O}(|V| \cdot \log |V|)$. \square

This implementation uses a greedy approach, simply adding new users to a new random hypergraph. Future work will examine this approach and study other joining mechanisms that might create more optimal hypergraph structures.

B. Multi-Party Channel Implementation on Ethereum and Bitcoin

The presented hypergraph network structure can be applied to both Bitcoin and Ethereum. As Ethereum is Turing complete and supports smart contracts, multi-party channels such as Gnocchi and Garou can be easily implemented in Ethereum. Yet, to be suitable for Bitcoin, these protocols require some adjustments. For instance, Gnocchi’s design relies on smart contracts for the fraud-proof mechanism. Moreover, a recent paper proposes a new mechanism to implement smart contracts in Bitcoin [67]. In Garou, another issue arises, as the protocol depends on a smart contract to determine the round leader.

However, this can be resolved by implementing a deterministic offchain protocol run by the channel participants to determine the leader in each round. For example, Algorand [68] uses a verifiable random function to deterministically elect a leader each time a new block is mined. Similarly, [69] uses a leader-election mechanism that can be run on Bitcoin. In future work, we intend to implement these mechanisms to allow HPNs to be implemented on Bitcoin too. Additionally, we intend to evaluate their performance and compare HPN topologies in different networks.

C. Demand-aware HPN Topologies

The current approach to creating HPN networks uses existing layer-2 topologies to create HPN-structured topologies. Yet, the network's structure might not always reflect user demand. Instead, other parameters such as the transaction demand, may assist in creating a more efficient network. Recent work [70] discusses using layer-2 transaction demands to create more efficient 2-party payment channel networks. The paper additionally presets several algorithms to create efficient topologies based on the transaction demand. The transaction demand helps understand transaction trends, which can then be used to design efficient hypergraphs. For instance, two users that communicate frequently with each other should be placed in the the same hyperedge, while nodes that do not communicate can be in distant hyperedges. In the future, we aim to study and evaluate hypergraph networks that are created not only from the topology of a network but based on its transaction patterns. Such networks have the potential to increase the success rate, decrease the signaling costs and reduce the amount of funds that need to be locked in the network.

D. Routing Payments and Capacity Allocation

This work takes a simple approach for routing, selecting for a payment a path as the shortest available path. Past works on payment channel networks studied optimizing the routing of payments, improving the channel capacity balancing [71] [43]. Balancing the routing of payments allows keeping a payment channel available for the routing of transactions in all its directions. Intuitively, balancing a hypergraph channel does not require similar payment values for each pair of nodes, but over time, aggregated change of close to 0 to each of its participating nodes. Similarly, another degree of optimizing refers to the allocation of capacity [9]. A node often has limited capacity that should be divided between the channels it takes part in. Extending the shortest path approach for routing that balances hypergraph payment channels and optimizing the capacity allocation is an interesting task for future work.

VII. CONCLUSION

In this work, we studied the potential of improving L2 solutions using hypergraph payment networks (HPNs). Several HPN topologies were suggested and their performance was evaluated, comparing them to the original Lightning Network and other state-of-the-art topologies, showing that HPN topologies can improve performance and reduce fees. It is

out of all doubt that the use of these topologies can have beneficial effects on L2 blockchains and they can be used to address scaling-related challenges. We see many different lines of potential future research. Among these is the evaluation of more realistic cost functions to find ideal HPN topologies. Additionally, we aim at the evaluation of HPNs built not only based on the original network topology but also based on real-life transaction patterns. Finally, we set forth new directions of research to study the security of the proposed HPN solutions.

REFERENCES

- [1] C. Grunspan and R. Pérez-Marco, "Double spend races," *International Journal of Theoretical and Applied Finance*, 2018.
- [2] J. Poon and T. Dryja, "The Bitcoin Lightning network: Scalable off-chain instant payments," 2016.
- [3] "Raiden network," <http://raiden.network/>, 2017.
- [4] J. Poon, "Plasma : Scalable autonomous smart contracts," in *White paper*, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:13266881>
- [5] C. Grunspan, G. Lehéricy, and R. Pérez-Marco, "Ant routing scalability for the Lightning network," *ArXiv*, vol. 2002.01374, 2020.
- [6] C. Grunspan and R. Perez-Marco, "Ant routing algorithm for the Lightning network," *CoRR*, vol. 1807.00151, 2018.
- [7] R. Khalil and A. Gervais, "Revive: Rebalancing off-blockchain payment networks," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017.
- [8] B. Ladóczy and Z. Luo, "Routing fee adjustment strategies in payment channels," in *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, 10 2023, p. 7.
- [9] Yekaterina Podiatchev and Ariel Orda and Ori Rottenstreich, "Survivable Payment Channel Networks," *IEEE Transactions on Network and Service Management*, to appear, 2025.
- [10] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Stabilization, Safety, and Security of Distributed Systems*, 2015.
- [11] C. Decker, R. Russell, and O. Osuntokun, "Eltoo: A simple layer 2 protocol for bitcoin," *White paper: <https://blockstream.com/eltoo.pdf>*, 2018.
- [12] S. Bartolucci, F. Caccioli, and P. Vivo, "A percolation model for the emergence of the bitcoin lightning network," *Scientific Reports*, vol. 10, p. 4488, 03 2020.
- [13] A. Goel and G. Ramseyer, "Continuous credit networks and layer 2 blockchains: Monotonicity and sampling," in *ACM Conference on Economics and Computation*, 2020.
- [14] S. S. Sahoo, M. M. Hosmane, and V. K. Chaurasiya, "A secure payment channel rebalancing model for layer-2 blockchain," *Internet of Things*, vol. 22, p. 100822, 2023.
- [15] M. Conoscenti, A. Vetro, and J. C. De Martin, "Hubs, rebalancing and service providers in the lightning network," *IEEE Access*, vol. 7, p. 132828, 2019.
- [16] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of layer-two blockchain protocols," *Journal of Network and Computer Applications*, vol. 209, p. 103539, 2023.
- [17] A. Kotzer and O. Rottenstreich, "Braess paradox in layer-2 blockchain payment networks," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023.
- [18] A. Goel and G. Ramseyer, "Continuous credit networks and layer 2 blockchains: Monotonicity and sampling," in *ACM Conference on Economics and Computation*, 2020.
- [19] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *IEEE Symposium on Security and Privacy (SP)*, 2019.
- [20] S. Micali, "Computationally sound proofs," *SIAM Journal on Computing*, vol. 30, no. 4, p. 1253, 2000.
- [21] C. Gentry and D. Wichs, "Separating succinct non-interactive arguments from all falsifiable assumptions," in *ACM Symposium on Theory of Computing (STOC)*, 2011.
- [22] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Innovations in Theoretical Computer Science Conference*, 2012.

- [23] H. Wu, W. Zheng, A. Chiesa, R. A. Popa, and I. Stoica, "DIZK: A distributed zero knowledge proof system," *Cryptology ePrint Archive*, Paper 2018/691, 2018.
- [24] C. Burchert, C. Decker, and R. Wattenhofer, "Scalable funding of bitcoin micropayment channel networks," *Royal Society open science*, vol. 5, no. 8, p. 180089, 2018.
- [25] A. R. Pedrosa, M. Potop-Butucaru, and S. Tucci-Piergiovanni, "Scalable lightning factories for bitcoin," in *ACM/SIGAPP Symposium on Applied Computing*, 2019.
- [26] R. A. Khalil and A. Gervais, "NOCUST - a non-custodial 2nd-layer financial intermediary," *IACR Cryptol. ePrint Arch.*, vol. 642, 2018.
- [27] C. Pan, S. Tang, Z. Ge, Z. Liu, Y. Long, Z. Liu, and D. Gu, "Gnocchi: Multiplexed payment channels for cryptocurrencies," in *Network and System Security*, 2019.
- [28] Y. Ye, Z. Ren, X. Luo, J. Zhang, and W. Wu, "Garou: An efficient and secure off-blockchain multi-party payment hub," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, p. 4450, 2021.
- [29] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *ACM conference on computer and communications security (SIGSAC)*, 2016.
- [30] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *ACM conference on computer and communications security (SIGSAC)*, 2018.
- [31] H. Huang, X. Peng, J. Zhan, S. Zhang, Y. Lin, Z. Zheng, and S. Guo, "Brokerchain: A cross-shard blockchain protocol for account/balance-based state sharding," in *IEEE INFOCOM*, 2022.
- [32] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *USENIX symposium on networked systems design and implementation (NSDI)*, 2019.
- [33] A. Sonnino, "Chainspace: A sharded smart contract platform," in *Network and Distributed System Security Symposium (NDSS)*, 2018.
- [34] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *IEEE symposium on security and privacy (SP)*, 2018.
- [35] Zhang, Hongliang and Song, Lingyang and Han, Zhu and Zhang, Yingjun, *Hypergraph theory in wireless communication networks*. Springer, 2018.
- [36] Karumba, Samuel and Kanhere, Salil S and Jurdak, Raja and Sethuvenkatraman, Subbu, "HARB: A hypergraph-based adaptive consortium blockchain for decentralized energy trading," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14216–14227, 2020.
- [37] Qu, Chao and Tao, Ming and Yuan, Ruifen, "A hypergraph-based blockchain model and application in internet of things-enabled smart homes," *Sensors*, vol. 18, no. 9, p. 2784, 2018.
- [38] A. Hentschel, D. Shirley, and L. Lafrance, "Flow: Separating consensus and compute," *ArXiv*, vol. 1909.05821, 2019.
- [39] C. Hannon and D. Jin, "Bitcoin payment-channels for resource limited IoT devices," in *International Conference on Omni-Layer Intelligent Systems*, 2019.
- [40] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and state channels: Payment networks that go faster than Lightning," in *Financial Cryptography and Data Security (FC)*, 2019.
- [41] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: A secure payment network with asynchronous blockchain access," in *ACM Symposium on Operating Systems Principles (SOSP)*, 2019.
- [42] R. Pickhardt and M. Nowostawski, "Imbalance measure and proactive channel rebalancing algorithm for the Lightning network," *CoRR*, vol. 1912.09555, 2019.
- [43] P. Wang, H. Xu, X. Jin, and T. Wang, "Flash: Efficient dynamic routing for offchain networks," in *ACM CoNEXT*, 2019.
- [44] V. Sivaraman, S. B. Venkatakrisnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020.
- [45] J.-H. Lin, K. Primicerio, T. Squartini, C. Decker, and C. Tessone, "Lightning network: A second path towards centralisation of the bitcoin economy," *New Journal of Physics*, vol. 22, 2020.
- [46] G. Kappos, H. Yousaf, A. M. Piotrowska, S. Kanjalkar, S. Delgado-Segura, A. K. Miller, and S. Meiklejohn, "An empirical analysis of privacy in the Lightning network," in *Financial Cryptography and Data Security (FC)*, 2021.
- [47] S. Tikhomirov, R. Pickhardt, A. Biryukov, and M. Nowostawski, "Probing channel balances in the Lightning network," *arXiv preprint arXiv:2004.00333*, 2020.
- [48] J. Herrera-Joancomartí, G. Navarro-Arribas, A. Ranchal-Pedrosa, C. Pérez-Solà, and J. Garcia-Alfaro, "On the difficulty of hiding the balance of Lightning network channels," in *ACM Asia Conference on Computer and Communications Security*, 2019.
- [49] U. Nisslmueller, K.-T. Foerster, S. Schmid, and C. Decker, "Toward active and passive confidentiality attacks on cryptocurrency off-chain networks," *arXiv preprint arXiv:2003.00003*, 2020.
- [50] S. Tochner, S. Schmid, and A. Zohar, "Hijacking routes in payment channel networks: A predictability tradeoff," *ArXiv*, vol. 1909.06890, 2019.
- [51] E. Rohrer, J. Malliaris, and F. Tschorsch, "Discharged payment channels: Quantifying the Lightning network's resilience to topology-based attacks," in *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2019.
- [52] A. Mizrahi, A. Zohar, and C. Diaz, "Congestion attacks in payment channel networks," in *Financial Cryptography and Data Security (FC)*, 2021.
- [53] O. Ersoy, S. Roos, and Z. Erkin, "How to profit from payments channels," in *Financial Cryptography and Data Security (FC)*, 2020.
- [54] F. Béres, I. A. Seres, and A. A. Benczúr, "A cryptoeconomic traffic analysis of Bitcoin's Lightning network," *Cryptoeconomic Systems*, 2020.
- [55] S. Brânzei, E. Segal-Halevi, and A. Zohar, "How to Charge Lightning: The Economics of Bitcoin Transaction Channels," *arXiv e-prints*, 2017.
- [56] G. Avarikioti, G. Janssen, Y. Wang, and R. Wattenhofer, "Payment network design with fees," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, 2018.
- [57] N. Vallarano, C. J. Tessone, and T. Squartini, "Bitcoin transaction networks: An overview of recent results," *Frontiers in Physics*, 2020.
- [58] "Lightning RFC: Lightning network specifications," <https://github.com/lightningnetwork/lightning-rfc>, 2021. [Online]. Available: <https://github.com/lightningnetwork/lightning-rfc>
- [59] "Raiden network 3.0.1 documentation," {<https://raiden-network.readthedocs.io/en/stable/index.html>}, 2022. [Online]. Available: <https://raiden-network.readthedocs.io/en/stable/index.html>
- [60] River, "The Lightning Network Grew by 1212% in 2 Years," *River research report*, 2023.
- [61] J. Wu and S. Jiang, "Local pooling of connected supernodes in Lightning networks for blockchains," in *IEEE International Conference on Blockchain*, 2020.
- [62] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug 2008, pp. 11–15.
- [63] M. R. Garey and D. S. Johnson, *Computers and intractability*. freeman San Francisco, 1979, vol. 174.
- [64] R. Bar-Yehuda and S. Even, "A local-ratio theorem for approximating the weighted vertex cover problem," in *North-Holland Mathematics Studies*. Elsevier, 1985, vol. 109, pp. 27–45.
- [65] C. N. Cordi, "Simulating high-throughput cryptocurrency payment channel networks," <https://core.ac.uk/download/pdf/158324076.pdf>, 2017.
- [66] N. Papadis and L. Tassiulas, "Blockchain-based payment channel networks: challenges and recent advances," *IEEE Access*, vol. 8, p. 227596, 2020.
- [67] Linus, Robin, "Bitvm: Compute anything on bitcoin," [URL: https://bitvm.org/bitvm.pdf-\(12.12.2023\)](https://bitvm.org/bitvm.pdf-(12.12.2023)), 2023.
- [68] Chen, Jing and Micali, Silvio, "Algorand," *arXiv preprint arXiv:1607.01341*, 2016.
- [69] Amin, Md Faizul Ibne and Watanobe, Yutaka and Rahman, Md Mostafizer and Kabir, Raihan, "Watchtower selection in off-blockchain pcn using peterson leader-election algorithm," in *New Trends in Intelligent Software Methodologies, Tools and Techniques*. IOS Press, 2022, pp. 193–202.
- [70] Khamis, Julia and Kotzer, Arad and Rottenstreich, Ori, "Topologies for Blockchain Payment Channel Networks: Models and Constructions," *IEEE/ACM Transactions on Networking, to appear*, 2025.
- [71] V. Sivaraman, S. B. Venkatakrisnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. C. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020.