A Survey on Blockchain Scalability: From Hardware to Layer-Two Protocols

Gabriel Antonio F. Rebello[1,2,3], Gustavo F. Camilo[1], Lucas Airam C. de Souza[1],
Maria Potop-Butucaru[2], Marcelo Dias de Amorim[2], Miguel Elias M. Campista[1], and Luís Henrique M. K. Costa[1]

[1]Universidade Federal do Rio de Janeiro (GTA/Poli/COPPE), Brazil
[2]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[3]Instituto de Pesquisas Eldorado, Brazil

Abstract—**Despite the great success of blockchain systems in recent years, blockchains still struggle to provide the same level of latency and throughput as centralized financial systems. The core of this problem lies in the inefficiency of consensus protocols. In this paper, we provide a survey on recent efforts to improve the scalability of blockchains. We focus on layer-two protocols, such as payment channel networks and transaction rollups, which process computations off-chain and only use consensus for dispute resolution. Layer-two protocols are expected to process microtransactions with sub-second latency and reduced fees, allowing blockchains to scale. Much of this work addresses the open challenges of payment channel networks, such as payment routing, channel rebalancing, network design strategies, security and privacy, payment scheduling, congestion control, simulators, and support for light nodes. We also dedicate a section to the existing implementations of smart-contract-based transaction rollups. Our work systematizes the state-of-the-art layer-two protocols, paving the way for future advances.**

Index Terms—**blockchain, payment channel networks, rollups.**

I. Introduction

Cryptocurrencies, the most popular blockchain-based application, allow users worldwide to transfer money securely and distributedly without relying on centralized authorities such as banks, agencies, or governments. In the recent past, this technology quickly became successful, with Bitcoin alone reaching over 100 million users in 2022 [1] and achieving a market value higher than the gross domestic product of more than 150 countries [2] when combined with Ethereum [3], [4]. Now, both academia and industry are investigating the adoption of blockchain in new research areas, including the Internet of Things (IoT) [5], smart cities [6], federated learning [7], [8], and even COVID-19 prevention [9], [10]. The success of blockchains is evident and unlikely to fade in the future, as blockchains provide valuable properties for computing systems: trust, transparency, privacy, automation, security, and decentralization.

Nevertheless, the performance of consensus protocols hinders the adoption of blockchain systems as a standard payment method for small and fast payments that occur in everyday life. Publishing a transaction in Bitcoin takes approximately one hour, can incur over $20 fees, and spends an amount of energy equivalent to the consumption of a US average household over 50 days [1]. Moreover, Bitcoin's and Ethereums' throughputs of approximately 7 tx/s and 15 tx/s, respectively, are still orders of magnitude smaller than the average of over 6,000 tx/s achieved by large credit card companies [11]. Such performance challenges are known in the literature as the *blockchain scalability problem*[1] and hinder the adoption of crypto technology for everyday use, where it is necessary to confirm a transaction within a few seconds with low to zero fees. Hence, scaling blockchain systems poses a significant research challenge that, if solved, has the potential to enable the worldwide adoption of blockchain systems in the life of ordinary citizens.

This paper provides an extensive survey of the recent efforts to improve blockchain scalability. We analyze over 120 papers, from 2013 to 2023, considering a four-layer architecture as reference [13]. Table I summarizes the papers and Figure 1 overviews the scalability solutions addressed in our work, classified per layer[2]. We highlight, however, that the main novelty of our work lies in the profound analysis of *layer-two protocols*, which have not been extensively covered by previous surveys. Layer-two protocols introduce an *off-chain* way of transacting that accelerates payment confirmation and minimizes the amount of transactions that need to appear in the blockchain, effectively offloading consensus protocols.

A large portion of this work focuses on payment channel networks (PCN), a popular layer-two solution to scale blockchain-based transfers. As of the writing of this work, the largest known PCN, the Lightning Network (LN), has approximately 100 million dollars allocated in over 80,000 channels and 17,000 nodes [142]. We cover the main open challenges of PCNs, such as payment routing, channel rebalancing, security, and privacy, PCN attacks, payment concurrency, payment scheduling, congestion control, PCN simulators, and support for light nodes, many of which are not mentioned

[1] Scalability here refers to Buterin's concept of scalability, i.e., the ability to process more transactions per second [12]. This concept contrasts with the notion of scalability in distributed computing, which is the ability to maintain performance when the number of nodes in the system increases. We discuss such notions in Section III-B.

[2] See Section IV for a precise definition of each blockchain layer.

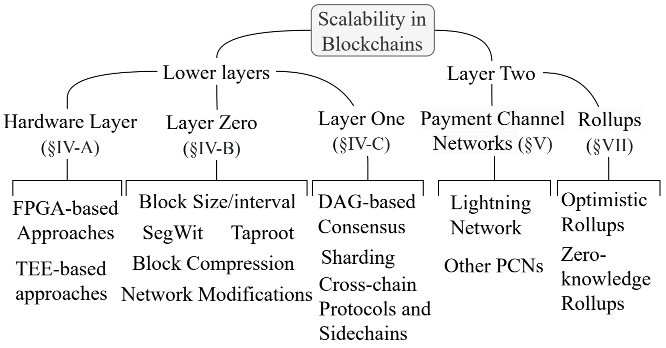

Figure 1. Taxonomy of the scalability solutions discussed in this paper. We adopt a layered architecture to classify protocols and dedicate Section IV to lower-layer solutions. Section V introduces the layer-two solutions, which are discussed in Sections VI and VII.

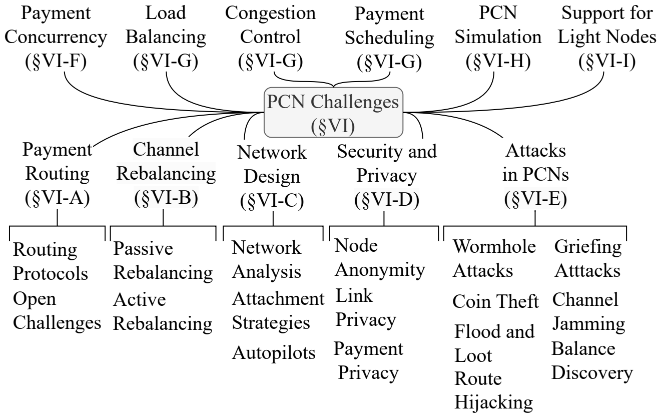

Figure 2. Taxonomy of the PCN challenges addressed in this paper. Section VI contains all the topics depicted in the figure.

Table I
SYSTEMATIC MAPPING OF SCALABILITY SOLUTIONS ACROSS BLOCKCHAIN LAYERS BY YEAR OF PUBLICATION.

Layer	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
HW	-	-	-	[14]	[15]	[16]	[17]	-	[18]	[19]	-
L0	[20]	-	[21]	[22]	-	[23]	[24]–[26]	[27]	-	[28]	-
L1	-	[29]	-	[30], [31]	[32]–[35]	[36]–[38]	[39]–[43]	[44]–[48]	[49]–[51]	-	-
L2	[52]	-	[53]	[54], [55]	[56]–[62]	[63]–[75]	[76]–[86]	[87]–[110]	[111]–[122]	[123]–[139]	[140], [141]
Total	2	1	2	6	12	18	20	30	16	19	2

in other surveys [109], [143]–[149]. We depict the discussed PCN challenges in Figure 2.

The work also dedicates a section to transaction rollups, a new off-chain technology that improves the throughput of blockchain applications by aggregating transactions before publishing them into the blockchain. Transaction rollups have been increasingly adopted to perform generic off-chain computations in blockchains that support smart contracts, such as Ethereum [4] and Cartesi [74]. We present the main existing types of rollups, such as optimistic and zero-knowledge rollups, providing a detailed analysis of their main features and drawbacks.

This article is organized as follows. Section II positions our work concerning existing surveys on blockchain scalability and payment channel networks. Section III explains the scalability problems of current blockchain systems. Section IV presents a layered architecture to classify proposals and describes the leading solutions for the lower layers. Section V introduces the definition of layer-two and announces the known layer-two protocols. Furthermore, it presents the theoretical background of payment channels, detailing the implementation of a payment channel, the opening, and closing of channels, the settling of conflicts, and how to route payments through payment channels. Section VI addresses the main open challenges and opportunities for research in PCNs. Section VII presents transaction rollups, a novel off-chain technology that aggregates payments to improve scalability. Finally, Section VIII concludes the work.

II. Related Work

Although the scalability of blockchain systems is a relatively recent research topic, we find several surveys in the literature that partially cover the solutions proposed in the latest years. We observe that most efforts adopt different layered architectures to categorize proposals and that, because of continuous and rapid developments, the focus of surveys has also evolved over the years. We separate them into four categories: (i) papers that focus on classifying layer-one proposals; (ii) papers that aim to systematize the knowledge of layer-two solutions; (iii) papers that seek to systematize the knowledge of blockchain scalability solutions regardless of the layer; and (iv) papers that focus specifically on the challenges of payment channel networks.

The first category groups consensus-modification proposals. Nguyen and Kim [150] provide the first large-scale survey of alternative blockchain consensus protocols. The work categorizes the existing proposals into proof-based and voting-based protocols, demonstrating the tradeoffs of each group. Xiao et al. [153] expand such classification into more detailed groups and compare the protocols via a five-component framework, which provides an efficient way to compare them. Yu et al. [155] present the primary implementation and challenges of blockchain sharding. Khan et al. [157] explore layer-one solutions in general, including alternative protocols and sharding, while briefly mentioning payment channels. Although some overlap might occur, such works are mostly orthogonal to ours since we focus on layer-

Table II
COMPARISON BETWEEN OUR WORK AND EXISTING SURVEYS ON BLOCKCHAIN SCALABILITY.

Reference	**Hardware Layer**		**Layer 0 (Network)**		**Layer 1 (Blockchain and Consensus)**					**Layer 2 (Offchain Solutions)**		
-	TEE-based aproaches	FPGA-based approaches	Transaction/ block size	Block interval	Alternative consensus	Sharding	DAGs	Cross-chain solutions	Sidechains	PCNs	Optimistic rollups	ZK rollups
Nguyen and Kim (2018) [150]	○	○	●	●	●	○	○	○	○	○	○	○
Kim *et al.* (2018) [151]	○	○	●	○	○	●	○	●	●	●	○	○
Jourenko *et al.* (2019) [143]	●	○	○	○	○	○	○	○	○	●	○	○
Xie *et al.* (2019) [152]	○	○	●	●	●	●	○	○	○	◐	○	○
Gudgeon *et al.* (2020) [13]	●	○	◐	○	◐	◐	○	○	◐	●	○	○
Hafid *et al.* (2020) [145]	○	○	●	●	●	●	●	○	●	●	○	○
Papadis and Tassiulas (2020) [109]	○	○	○	○	○	○	○	●	○	●	○	○
Xiao *et al.* (2020) [153]	●	○	●	●	●	●	●	○	○	○	○	○
Yang *et al.* (2020) [154]	○	○	○	○	○	●	●	●	●	●	○	○
Yu *et al.* (2020) [155]	○	○	◐	○	○	●	◐	○	○	○	○	○
Zhou *et al.* (2020) [156]	○	○	●	●	●	●	●	●	●	●	○	○
Khan *et al.* (2021) [157]	○	○	●	●	●	●	○	○	○	◐	○	○
Khojasteh and Tabatabaei (2021) [146]	◐	○	◐	○	◐	◐	◐	○	○	●	○	○
Zhao *et al.* (2021) [148]	○	○	○	○	○	○	○	○	○	●	○	○
Sanka and Cheung (2021) [158]	○	●	●	●	●	●	●	●	●	●	○	●
Sguanci *et al.* (2021) [147]	○	○	●	●	○	◐	○	○	●	●	●	●
Gangwal *et al.* (2022) [149]	●	○	◐	○	●	●	●	●	●	●	●	●
Nasir *et al.* (2022) [159]	●	○	●	●	●	●	○	○	○	●	○	○
Our work	●	●	●	●	●	●	●	●	●	●	●	●

○not covered; ◐partially covered; ●covered.

Table III
COMPARISON BETWEEN OUR WORK AND EXISTING SURVEYS ON PAYMENT CHANNEL NETWORKS.

Reference	**Payment routing**	**Channel rebalancing**	**Network Design**	**Security and privacy**	**Attacks to PCNs**	**Payment Concurrency**	**Payment Scheduling**	**Congestion Control**	**PCN simulation**	**Support for light devices**
Jourenko et al. (2019) [143]	●	●	◐	◐	◐	○	○	◐	○	○
Gudgeon et al. (2020) [13]	●	◐	○	●	◐	◐	○	○	○	○
Hafid et al. (2020) [145]	○	○	○	◐	○	○	○	○	○	○
Papadis et al. (2020) [109]	◐	●	●	●	○	●	●	●	○	○
Khojasteh et al. (2021) [146]	●	◐	○	●	○	●	○	●	○	○
Sguanci et al. (2021) [147]	◐	○	○	●	○	○	○	○	○	○
Zhao et al. (2021) [148]	○	○	●	●	◐	○	○	○	○	○
Gangwal et al. (2022) [149]	◐	◐	○	◐	●	○	○	○	○	○
Our work	●	●	●	●	●	●	●	●	●	●

○: not covered; ◐: partially covered; ●: covered.

two solutions.

The second category of surveys includes works that classify existing layer-two solutions. Gudgeon et al. [13] provide a systematization of knowledge (SoK) of layer-two proposals that includes payment channels and commit chains. The authors also discuss the four-layer blockchain architecture we adopt in our work. Sguanci et al. [147] compare layer-two proposals. Their comparison is restricted to three popular proposals, namely the Lightning Network [54], Ethereum rollups [160], and Plasma [33]. Gangwal et al. [149] provide a taxonomy of layer-two protocols while focusing on the security of payment channels, rollups, and commit chains. Unlike the above-cited papers, which solely address layer-two solutions, our work discusses scalability proposals across all blockchain layers.

The third category comprises papers that compare solutions across multiple layers. Kim et al. [151], Xie et al. [152], and Yang et al. [154] provide short reviews of scalability solutions on layer one and layer two. The descriptions of the analyzed proposals, however, are high-level and do not cover other layers. Hafid et al. [145] present a comprehensive survey that delves deeper into sharding, alternative consensus, and directed acyclic graph-based (DAG) proposals. The work also analyzes several payment channel network implementations, such as Lightning [54] and Raiden [89]. Zhou et al. [156] expand the range of analyzed solutions by including layer-zero proposals. Nasir et al. [159] extensively study existing efforts on blockchain scalability, categorize them as on-chain or off-chain, and present applications. The above works, however, either fail to cover proposals in the hardware layer or recent layer-two proposals such as rollups. To the best of our knowledge, Sanka and Cheung [158] provide the most complete survey about blockchain scalability solutions across all layers. Compared to theirs, the main improvement of our work is a more significant focus on the main challenges of payment channel networks, which is the most prominent layer-two solution in the present day.

Lastly, we compare our work with surveys focusing on payment channel networks. Jourenko et al. [143] address the main challenges in PCNs, such as payment routing, channel rebalancing, channel settlement, and user privacy and anonymity. Khojasteh and Tabatabaei [146] present a shorter survey on the same topics. Both works mention multiple payment channel networks and recognize Bitcoin's Lightning Network as the most mature implementation to extract knowledge from. Papadis and Tassiulas [109] expand the previous works by discussing network load balancing and congestion control. The work also addresses channel creation and network design strategies, analyzes the topology of the Lightning Network, and discusses the scarcity of available open data to be used in scientific investigations. Zhao et al. [148] provide a systematic overview of the Lightning Network challenges and countermeasures, most of which are general PCN challenges. The authors focus on the security of users and the robustness of the network. Other works mentioned in the previous paragraphs, namely [13], [145], [147], [149], also address PCN challenges to some extent despite being categorized as general scalability surveys. However, no single survey addresses all the known challenges for payment channel networks, nor do they present new technologies such as PCN simulation and support for light devices.

Contribution. To the best of our knowledge, our work is the first to focus on the challenges of payment channel networks while still covering solutions from other blockchain layers. We provide a per-topic comparison between our work and other works on blockchain scalability in Table II. We also summarize the difference between our work and existing surveys that delve specifically into payment channel networks in Table III. We do our best to cover topics that previous works do not address and to update the knowledge on challenges that changed over the years. Besides, we dedicate special attention to optimistic and zero-knowledge rollups since we observe they are important layer-two solutions that only a few surveys address in detail.

III. The Blockchain Scalability Problem

Despite providing disruptive and innovative features, blockchains still present significant latency, power consumption, and transaction throughput issues. The collection of such issues is known as the "blockchain scalability problem" in the literature and is today one of the most important research topics on blockchain technology [149], [150], [158], [159]. In this section, we introduce the proper background needed to understand the problem and formalize it through the enunciation of the blockchain scalability trilemma.

A. Blockchains and Consensus

A blockchain system is a distributed ledger technology (DLT) that leverages independent validator nodes to record, share, and synchronize transactions in a decentralized system over a peer-to-peer network. Each node in the network stores a local copy of the blockchain where it can verify any transaction issued on the network since its creation. The blockchain data structure is a chained list of signed transactions batched into blocks. Each block contains a header with the hash of the predecessor block and a content section that stores transactions, as shown in Figure 3. A transaction represents an atomic action that transfers assets between a sender and a receiver. To transfer assets, a sender must sign a transaction containing assets he/she owns along with the receiver's identifier and send it to a subset of nodes called *validators*. The transaction is confirmed once it appears in a block, meaning that it has been selected by a node and confirmed by the others. The combination of signed transactions, linked blocks, and replication of the complete data structure provides immutability to any data stored in a blockchain, creating an incorruptible log of transactions.

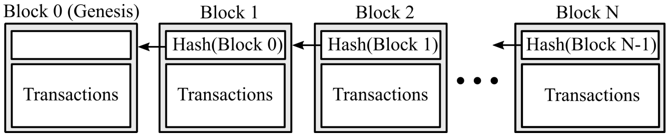

Figure 3. Data structure of a blockchain, in which each block is linked to the previous block via a cryptographic hash function. The replication of such structure in independent nodes provides immutability to transactions.

Consensus in blockchain systems is the process by which the independent validators in the network decide, collectively, whether to accept or refuse the addition of a new block into the blockchain. A blockchain consensus protocol is a distributed algorithm that ensures the system evolves correctly, adding one new block at a time. Figure 4 illustrates how a generic blockchain consensus protocol works. Assume every validator starts at a previously-validated state S. In each round, the consensus leader, i.e., the validator with the right to propose a block, aggregates the received transactions into a block and broadcasts it on the network to be validated locally by the other validators. Upon receiving the proposed block, each validator evaluates it independently and, if approved, adds it to their blockchain, locally reaching the new S' state. When enough validators reach the new state locally, the protocol considers that there has been consensus and that the system as a whole has validated the new block. Hence, S' becomes the current *global* state of the blockchain that all nodes must synchronize with, regardless of their opinion on previous rounds.

Proposing, broadcasting, and verifying the block consume time and energy proportional to the number of validators. A mechanism for defining the consensus leader on each round is also needed. Such procedures and leader-election mechanisms define how fast a block is considered valid, and consequently impact the performance of the blockchain. As many works have shown, the core of the blockchain scalability problem lies in the efficiency of consensus protocols, which must ensure that the system adds blocks to the ledger in a safe manner [151], [152], [156].

B. The Blockchain Scalability Trilemma

The blockchain scalability problem is a generic umbrella term that encompasses the challenges of improving

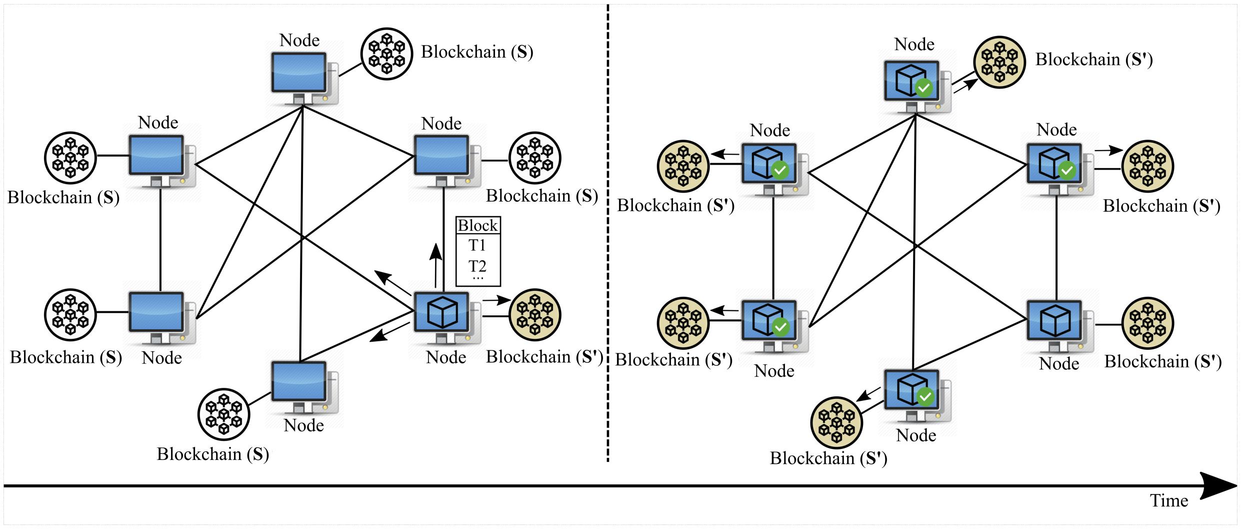

Figure 4. The validation of a block using a generic consensus protocol. On each round, the consensus leader proposes a new block that changes its local state from S to S' and broadcasts it to the network. The other participants independently verify and add the proposed block to the blockchain, replicating the state S' consistently.

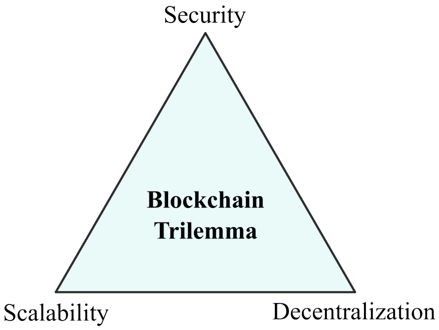

Figure 5. Illustration of the trilemma observed in blockchain-based systems. The trilemma states that no consensus protocol can simultaneously provide security, scalability (measured in transaction throughput), and decentralization (measured in the number of nodes that participate in consensus). Graphically, all blockchains can be represented by a point inside the trilemma triangle.

blockchain performance. For the purposes of this paper, it helps to narrow down such concept to a more specific definition based on the *blockchain scalability trilemma*[3], a term coined by Ethereum's founder Vitalik Buterin [12]:

Concept definition 1. *(The blockchain scalability trilemma). Given the following properties of blockchains:*

- ***Scalability**: the capacity to process transactions at high throughput;*
- ***Decentralization**: the capacity to process transactions without relying on trusted parties or small groups;*
- ***Security**: the capacity to successfully resist collusion attacks;*

the blockchain scalability trilemma is a conjecture that states that no blockchain system can provide all properties simultaneously. Consequently, every blockchain forfeits at least one property at any given time.

We highlight that the trilemma refers to Vitalik Buterin's notion of scalability instead of the classical concept of scalability in distributed computing. The latter concept, which in blockchains corresponds to the ability of the consensus protocol to maintain throughput even when the number of validators significantly increases, is captured by the decentralization property. Henceforth, we adopt the term "scalability" to refer to Buterin's concept and "decentralization" to refer to the concept of distributed computing.

We illustrate the trilemma in Figure 5. The rationale behind it stems from the observation that the validation of transactions in a blockchain system occurs, by definition, through the agreement between the validators of the system. On the one hand, the more nodes participate in consensus decisions, i.e., the more decentralized the system becomes, the more complex and time-consuming the decision-making and broadcasting of messages in the network. On the other hand, reducing the number of validators to improve throughput concentrates the decision power on fewer agents, reducing the level of decentralization and increasing the financial monopoly of the network. Some protocols try to provide high throughput with many validators by allowing multiple blocks to be approved simultaneously, but this also compromises security since conflicting transactions could be considered valid [32], [46]. This event is known as a *fork* in the blockchain. Most protocols eventually solve forks by finalizing a block and discarding the others, but this process also takes time. Besides, transactions in discarded blocks are rolled back and become untraceable, meaning that the

[3] Some authors refer to the blockchain scalability trilemma as simply "the blockchain trilemma" or "the scalability trilemma" [145], [147], [149], [157].

security of a transaction is only guaranteed when its block is finalized. Despite being a conjecture based on informal logic and empirical observations, the trilemma consistently occurs in all major known blockchain systems [13], [153], [161].

C. Modifying Consensus to Improve Scalability

As security is essential in blockchains, in practice, the trilemma mentioned above becomes a dilemma for consensus protocols: the protocol needs to choose between scalability, measured in the number of transactions processed per second, and decentralization, measured in the number of consensus validators. The trade-off between the two properties can be seen in the comparison between the leading blockchain consensus protocols shown in Figure 6.

Proof-based protocols adopt decentralized mechanisms to define who has the right to propose a block, allowing any user to participate in the process. However, these protocols achieve low transaction throughput because they need to introduce spam control tools to mitigate forks in the blockchain and solve forks that occur. Thus, proof-based protocols are challenging to scale but very decentralized, making them more suited to public systems with many users [153], [162], [163]. The main systems that use this type of consensus are cryptocurrencies, such as Bitcoin [3], Ethereum [4], or IOTA [32].

Conversely, committee-based protocols elect a group of special validators that propose blocks through direct communication [19], [164]–[170]. The choice of who participates in the committee can be made in several ways, such as random selection or an election based on the number of coins each user invested. The selection of a committee sacrifices decentralization as only some users participate in the decision but increases the number of transactions processed per second since decisions are independent of time-consuming computational mechanisms and spam control. Committee-based protocols are therefore adapted to systems that already expect some level of centralization, such as consortia of companies, banks, and governments. The prominent representatives of this type of system are Hyperledger Fabric [171], Hyperledger Sawtooth [172], and Ripple (RPCA) [173]. The design of committee-based protocols must include security mechanisms to avoid collusion and denial-of-service attacks.

Several consensus protocols try to solve scalability through hybrid solutions combining the best proof-based and committee-based consensus approaches [174]–[177]. The main objective of such protocols is to provide each property at a specific phase of consensus, leveraging extra-consensus mechanisms to mitigate possible vulnerabilities. However, the hybrid approach suffers from the same trade-off between scalability and decentralization of other approaches, reaching intermediate levels in both aspects for its prominent representatives, EOS.IO [174] and Tendermint [175]. Despite the efforts, no consensus protocol consistently provides a throughput of thousands of transactions per second with high decentralization. This desired "ideal zone" that would allow scaling blockchain systems without compromising their decentralization is shown in Figure 6.

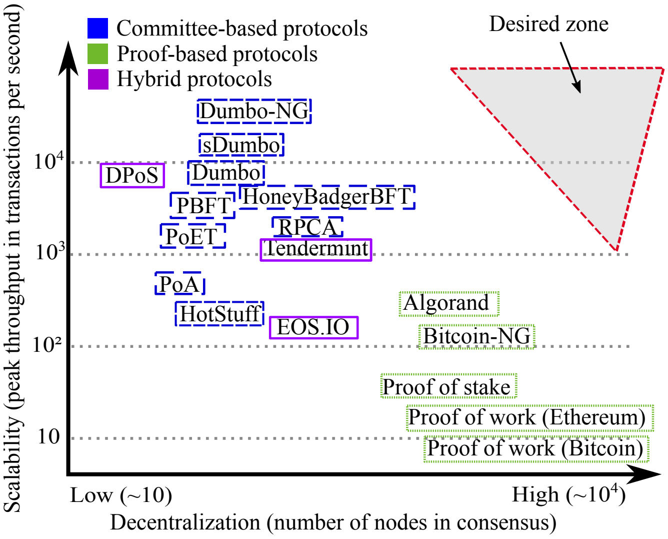

Figure 6. Comparison between the main consensus protocols of blockchain-based systems. The observed trade-off between performance and decentralization makes it difficult to propose a scalable protocol that is tolerant of collusion attacks.

IV. Scalability in Layers HW, L0, and L1

In contrast with approaches that propose new consensus protocols, recent works on blockchain scalability increasingly focus on improving the efficiency of other blockchain components [31], [32], [54], [127], [156], [178]. Like other works, we classify such solutions using a layered architecture as to provide a clear view of the entire blockchain environment [13], [109], [143], [149], [156]–[159]. Moreover, we adopt a popular architecture [13] to easily compare our work with the literature and provide a detailed context of each proposal. The architecture, illustrated in Figure 7, separates blockchain systems into four layers of increasing complexity:

- The **hardware layer (HW)**, which consists of user devices and low-level technologies. Proposals in this layer focus mostly on improving the underlying hardware to accelerate block proposal and retrieval.
- **Layer 0 (L0)**, which comprises the network infrastructure and message exchanges. Proposals in this layer attempt to either adjust block sizes to fit the capabilities of the underlying network or to improve the network itself.
- **Layer 1 (L1)**, in which consensus protocols and blockchains operate. Proposals in this layer modify consensus or the blockchain structure itself to improve transaction throughput.
- **Layer 2 (L2)**, which encompasses technologies that perform *off-chain* processing as much as possible, i.e., proposals that improve throughput by validating transactions outside the blockchain.

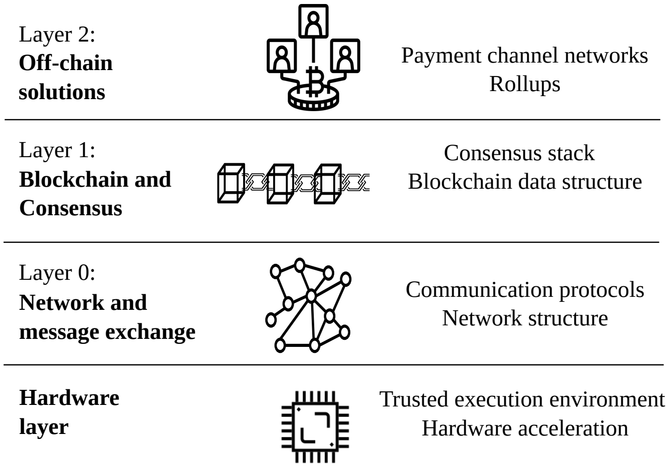

Figure 7. Layers of blockchain-based systems. Most of the proposals to improve the scalability of these systems are layer one or layer two, as they are the simplest to modify on a large-scale decentralized environment.

In the following sections, we present the existing proposals to improve scalability on the hardware layer, layer zero, and layer one while delegating layer-two proposals to their dedicated sections, as they are the focus of our work. We note, however, that the boundary between layer one and layer two is still discussed in the literature. This paper considers all solutions that rely on consensus protocols to validate transactions, including sidechains, block sharding, DAGs, and cross-chain proposals, to be *layer-one proposals*. We discuss this decision and provide a proper definition of layer-two protocols in Section V.

A. Hardware Layer (HW)

At the bottom layer of the architecture, most proposals seek to increase the performance of blockchains through hardware-based solutions, such as hardware acceleration and trusted execution environments.

Among the acceleration proposals, Sakakibara et al. propose using Field Programmable Gate Array (FPGA) to implement a cache memory that stores the most popular blocks and quickly transmits them to users who request them [16]. The technique speeds up access to blocks by allowing devices to obtain information directly from the FPGA, thereby reducing user access time to transactions by up to seven times. Another similar proposal leverages FPGA technology to make verification and dissemination of blocks faster within a consensus protocol round [18]. The results indicate that using FPGA to accelerate consensus increases the total system throughput by up to 12 times.

Besides FPGAs, Trusted Execution Environments (TEE), such as Intel® Software Guard Extensions (SGX), can be used as a hardware-based technology to prevent malicious behavior by consensus participants. TEEs ensure participants cannot perform specific actions, such as lying about a block, which allows protocols in the upper layers to relax or eliminate time-consuming security mechanisms [14], [15], [77]. The TEE technology is the basis of some highly efficient consensus protocols, such as the Hyperledger Sawtooth's Proof of Elapsed Time (PoET) [19]. The protocol employs SGX to replace the decentralized computational challenge with simple tamper-proof timers. Thus, to elect the leaders, all participants generate a random value for a timer, and when it expires, they propose blocks in an orderly manner. This process reduces the time of a consensus round and increases the protocol scalability to up to 2,300 transactions per second [17]. The disadvantage is that all participants must support SGX.

1) Discussion: The main limitation of new hardware solutions is the heterogeneity of user equipment on a highly decentralized network. Since blockchain systems have no central authority, it is difficult to guarantee that all participants support the hardware requirements needed to implement the improvements. Consequently, hardware solutions prioritize security and scalability regarding the blockchain trilemma, but they often come at the cost of increased centralization. Most proposals for highly decentralized environments with many users focus on the upper layers of the architecture. Thus, this approach is most effective in controlled, partially centralized environments, which do not represent the majority of blockchain applications [17], [153], [161].

B. Layer Zero (L0): Network and Message Exchange

Layer zero solutions aim to improve the efficiency of information propagation in the underlying communication network without modifying consensus protocols. The main proposals to increase scalability in this layer try to reduce the redundant data transmitted to the network, for example, by optimizing the size of messages, by modifying block size and interval, by compressing blocks, or by modifying the network.

1) Modifying Block Size and Interval: A naive solution for improving transaction throughput is to produce larger blocks in the same time interval[4]. Large blocks would mean processing more transactions at each consensus round since the extra block space fits transactions that would otherwise stay in the mempool[5].

Nevertheless, the capacity of the links in the underlying communication network limits block sizes. For example, Bitcoin produces a block of approximately *1.1 MB every 10 minutes* [179]. Each block contains around 2,000 transactions, yielding a throughput of about 3.3 transactions per second. Blocks should contain more than 14 million transactions, with a total size of approximately 8 GB, to achieve the VISA's peak throughput of over 24,000 transactions per second with the same block time [180]. Broadcasting and validating such large blocks becomes a new bottleneck for the system since

[4] Some authors consider the modification of block sizes to be layer-one proposals as they often require changes to consensus protocols [109], [145], [156]. We argue that block sizes are mostly linked to the expected transmission delays in the underlying P2P network even when this is true.

[5] The mempool is a familiar name for the set of unconfirmed transactions in blockchains. Validators select transactions from the mempool to create a new block.

only a few users can receive and process 8 GB of data in 10 minutes. This restriction also compromises security, as most users would always see an outdated blockchain version. Hence, blockchain-based systems must consider message transmission delays as their key indicator for defining the optimal block size.

An equivalent approach is to reduce the time needed to validate blocks. For instance, Ethereum reduces the time between blocks from 600 seconds to only 14 seconds when compared with Bitcoin [3], [4]. Although this modification can theoretically improve the latency of high-priority transactions, it implies the need for reduced block sizes to maintain the synchronization between blockchain replicas. Blocks in Ethereum are around 64 kB and contain 350 transactions on average. Reducing block intervals also increases the number of forks in the blockchain since the probability of a fork is a ratio of the time needed to propagate the block over the block interval. The closer the propagation and block interval are, the greater the probability of forks. Also, because forks are more likely to exist, nodes typically wait for 250 blocks to determine that a transaction has finalized, compared to 6 blocks in Bitcoin [181]. Therefore, in practice, the average transaction latency and throughput on the Ethereum blockchain remain in the same order of magnitude as of Bitcoin, i.e., 58 minutes and 15 tx/s, respectively [153].

2) Segregated Witnesses (SegWit): Another approach to increase transaction throughput is to reduce the size of transactions so that more transactions fit into one block. This idea stems from the observation that transaction sizes are dominated by the cryptographic information needed to verify its authenticity. In blockchains, digital signatures typically represent 60 to 70 percent of the total transaction size [152].

One approach that simultaneously increases block size and reduces the space occupied by signatures is Segregated Witness (SegWit) [21]. SegWit is a modification to the Bitcoin protocol proposed to prevent transaction malleability [182]. Malleability is a property of cryptographic algorithms: malleable algorithms allow attackers to modify a cipher while maintaining the same verification result. Before SegWit, transaction malleability could occur when a valid transaction (regarding its signature) was tampered before being confirmed in the blockchain. The attacker would gather information from the transaction in the mempool and slightly change one of its fields so that the signature verification algorithm is still valid. This vulnerability was feasible because the signature algorithm did not cover all transaction data, and transaction signatures were stored in one of the transaction fields. However, the transaction identifier would change since it is a hash of the complete transaction. The proposed solution moves transaction signatures to an external field, thus reducing the transaction size [27].

SegWit blocks include a 3 MB extension over the original 1 MB block size to fit transaction signatures, which prevents the signature from being part of the transaction identifier. Consequently, both SegWit and legacy nodes consider only the transaction fields, allowing them to fit more transactions into a block. SegWit nodes receive larger blocks, while legacy clients visualize smaller transactions. Figure 8 illustrates the difference between legacy and SegWit transactions. SegWit transactions have two parts: the first contains the sender's and receiver's wallet addresses, whereas the second part, called witness data, contains the transaction signatures. Therefore, removing transaction signatures to an external storage in the witness field solves the transaction malleability problem and increases throughput in the system. Nevertheless, the throughput achieved, around 20 tx/s, is still far from the desired values to serve customers on a global scale [158].

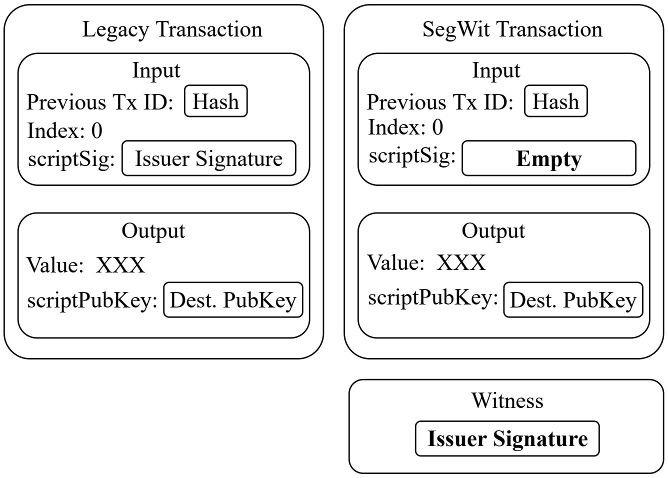

Figure 8. A legacy transaction versus a SegWit transaction. Legacy transactions include the digital signature in the transaction itself, generating an overhead on the final transaction size. SegWit transactions decouple the sender's signature from the transaction by creating an external witness field.

3) Taproot: Taproot, a modification to the Bitcoin blockchain built on top of SegWit, was proposed in the Bitcoin Improvement Proposal (BIP) 341 and 342 to increase user privacy and network scalability [183], [184]. The proposal combines two technologies: Schnorr signatures [185] and Merklized Alternative Script Trees (MAST) [186].

Schnorr signatures introduce a new form of creating multi-signature schemes that consume less data than the previously-used multi-signature protocol. For example, a pre-Schnorr multi-signature address is implemented by creating a script that hardcodes all the possible combinations of signatures of users that can unlock funds. One can easily see that this exponentially increases transaction sizes. Instead of this cumbersome script, Schnorr signatures allow public keys to be combined into a single key, K, which combines all the public keys of parties that can spend the funds. Once the authorized users want to spend the assets, they compute partial signatures to form a single transaction signature. This reduces the number of conditions in the script, thus reducing the size of transactions published in the blockchain.

In addition, MAST reduces the information sent to the blockchain and increases user privacy. Before MAST, users who wanted to define a script to spend their coins needed to publish all conditions inside the script, even though parties will only trigger one condition. Thus, transactions that adopt the pre-MAST form of writing scripts incur unnecessary overhead in transaction size since a contract can be composed of numerous conditions. MAST separates scripts into Merkle trees which have script conditions as leaves. This new transaction format only needs to include the condition used to spend the funds and the hashes of the unused parts of the tree to prove that the user knows how to build the root of the Merkle tree. The new transaction structure increases user privacy since it only reveals the necessary information to spend the funds and significantly reduces the overhead caused by scripts with many conditions.

4) Block Compression: Other proposals prefer to compress blocks to minimize message sizes in the network [22], [24]. These algorithms accomplish compression by discarding information that is already stored in the mempool, reducing redundant-data transmissions.

Matt Corallo proposed the compact block relay algorithm for Bitcoin, an upgrade similar to SegWit, which reduces the size of blocks while they are transmitted [22]. Compact blocks contain only the block header and 32-byte transaction identifiers. The author argues that this is enough to reconstruct blocks since transaction data has been previously received and stored in the receiver's mempool. Consequently, nodes that receive a compact block must use the transaction identifiers to collect transaction data from the mempool. If some transaction data cannot be recovered, the receivers can request the complete transaction from the sender. This eliminates the need to send complete transactions in a block over the network and dramatically reduces message size. Compact block relays allow nodes to transmit 1 MB blocks with 15 kB messages, reducing almost 10 times the amount of transmitted data [158].

Txilm is another proposal that leverages compact block relays to reduce the transmitted data [24]. Txilm replaces the transaction identifiers in compact blocks with a short hash of the transaction IDs to save space. Like compact block relays, nodes that receive Txilm blocks can reconstruct the original block by adding transaction data from their mempool. In Txlim, however, the receivers must compute the hashes of all mempool transaction IDs to match them with the hashes in the compact block. This approach has been shown to reduce the bandwidth by up to 80 times compared to the default block transmission protocol [24], [156], [158].

5) Network Modifications: Relay networks are composed of nodes that effectively relay and broadcast network messages. The approach increases the propagation speed of updates, such as blocks and transactions, and its broadcasting is faster than the main blockchain. Corallo [20] proposed a centralized Bitcoin relay network to relay blocks globally. The centralization, however, faces security challenges despite its scalability benefits. FIBER (Fast Internet Bitcoin Relay Engine) is a recent Bitcoin relay network that connects nodes and broadcasts Bitcoin compact blocks for shorter propagation delay [25]. FIBER sends and receives blocks to miners connected through its six servers worldwide. Cardano blockchain platform adopts Recursive Inter-Network Architecture (RINA), a new type of network technology, to propagate transaction information [187]. RINA provides a secure and programmable environment to propagate data efficiently with a short delay.

Another protocol for high-performance computing network architectures is Remote Direct Memory Access (RDMA). RDMA is mainly applied for high throughput communications in data center environments. The protocol executes a kernel-passing to directly transfer data from the Network Interface Card (NIC) to the application's memory, avoiding time-consuming memory copies. Thus, this approach can be attached to blockchain networks to increase the throughput and decrease the latency. Rubin [23], BoR (Blockchain over RDMA) [26], and CloudChain [28] are some examples of proposals that improve consensus protocols with RDMA to reduce the communication overhead.

6) Discussion: Modifications to layer zero aim to reduce the propagation of redundant information or improve block delivery by the underlying network. The most straightforward solutions attempt to optimize block intervals and block sizes to match the characteristics of the network [109], [145], [156]. SegWit and Taproot propose modifications to the Bitcoin protocol that reduce transaction size, making it possible to fit more transactions in each block [21], [183], [184]. Instead of sending whole blocks, compression protocols send only block hashes and transaction identifiers, which reduces block propagation time [22], [24]. All these approaches provide limited scalability gain, despite being widely adopted. Moreover, changing the block structure in public blockchains can result in a fork if the network participants do not widely accept the update. For example, Bitcoin Cash [188] was created by a hard-fork[6] in Bitcoin as a response from some users to the SegWit update [189]. Forks weaken the security of blockchains as an attacker requires less computational power to attack the consensus [190]. Other changes to the network infrastructure, such as using relay networks and high-performance computing network architectures, effectively speed up the delivery of blocks but are also challenging to implement in decentralized environments.

C. Layer One (L1): Improving Consensus

A rigorous comparison between existing consensus protocols is out of the scope of this paper as it has already been extensively addressed in previous surveys [150], [152], [153], [161], [191]. Proposing new consensus protocols is, by far, the most explored strategy to improve scalability in blockchains [150]–[153], [155], [191]. Nonetheless, there are several interesting works that, instead of new protocols,

[6] Hard forks are backward-incompatible modifications to the blockchain that force all nodes to update their software. Soft forks are forward-compatible modifications that do not compromise the functionality of old software versions.

propose structural changes to the consensus layer itself. In this section, we address such proposals and place them into two categories: solutions that propose changes to consensus in the main blockchain and solutions that move consensus outside the main blockchain. The first category includes directed acyclic graphs (DAG) and sharding. The second category includes sidechains supported by cross-chain technologies.

1) DAG-based Consensus: The distributed ledgers based on directed acyclic graphs (DAG) potentially increase transaction throughput while ensuring properties similar to blockchains [44]. While blockchains aggregate transactions into logically chained blocks, each transaction is chained independently in a DAG. Like blockchains, DAGs are immutable path data structures linked by hashes. The main difference between a blockchain and a DAG is that a new transaction can refer to any predecessor transaction, not only to the last one. Moreover, unlike blockchains, DAGs allow users to validate and process transactions asynchronously without relying on a round-based consensus protocol. DAGs are an example of ledger technology that provides high scalability and decentralization while sacrificing security as it deliberately allows forks to exist in the system [39], [49].

IOTA is a DAG-based cryptocurrency built to meet the machine-to-machine (M2M) micro-payments characteristic of IoT environments [32]. Their payment mechanisms and consensus protocol, formalized by Popov in 2016, use a new data structure called the Tangle. The Tangle is a distributed ledger structure that stores network transactions in a DAG rather than in a blockchain. In addition, the Tangle eliminates the distinction between clients and validators: system users validate previous transactions to issue a new transaction. A notable feature of IOTA compared to traditional blockchain consensus is that different participants in the network may have different views of the transactions. This contrasts sharply with the unique global view needed for blockchains, in which all participants have identical copies of the ledger.

Figure 9 shows an example of IOTA's Tangle data structure. Each graph vertex represents a transaction, and each edge represents a transaction validation. The user must confirm at least two unconfirmed transactions to add her/his transaction to the Tangle. Uncommitted transactions are called "tips" of the Tangle. To add a transaction to the network, the user adds the hash of two chosen tips to her/his transaction, solves a proof-of-work computational challenge, and broadcasts the result to the network. Proof of work, in this case, has a lower difficulty than Bitcoin and serves only as a mechanism to prevent transaction spam. The transaction commit procedure creates two new directed edges that confirm previous transactions and represent a generalized version of the sequence of hash functions of the blockchain.

Nevertheless, no consensus mechanism prevents the confirmation of conflicting transactions. The authors argue that this is not an issue considering honest users are expected to have more computational power than malicious nodes, like in Bitcoin. However, to accomplish this, the current version of IOTA centralizes transaction confirmation in a trusted coordinator [192]. In "The Coordicide", a published paper which proposes to replace the coordinator with new decentralized confirmation strategies [45]. However, many security challenges still need to be addressed [193].

Other examples of systems that use DAGs are Byteball, Hashgraph, and Avalanche [30], [46], [194], [195]. In Byteball, graph transactions gradually converge on the main chain using particular nodes, called witnesses, with reputation-based validation power [30]. On the other hand, Hashgraph is inspired by Byzantine Fault Tolerance (BFT) style consensus protocols based on voting among witnesses [46]. Similarly, Avalanche repeatedly queries a random subset of validators to get their preference and to decide whether a transaction should be accepted [194], [195]. Byteball, Hashgraph, and Avalanche can only guarantee the order and security of transactions if participants share the same global graph view.

2) Sharding: Sharding was first proposed for improving throughput in distributed systems and databases [31], [156], [196]. The core idea is to distribute parts of the data to different processor groups called shards and aggregate the processed data. In the context of blockchains, shards represent groups of block validators that agree upon a transaction subset. The approach parallelizes the processing of transactions in each shard and consequently speeds up consensus. The transaction throughput is expected to grow linearly with the number of existing groups/shards. While some sharding proposals address only financial applications, e.g., transferring coins between two nodes, sharding techniques can handle multiple generic computing tasks, which extends their applicability to other cases that use smart contracts. We depict an example of blockchain sharding in Figure 10.

The main challenge of sharding is to ensure the security of transactions in the blockchain. Because shards process different transactions concurrently, each shard has a different views and must define a way to avoid double-spending. This is typically done by having some overlapping information between shards. Sharding also impacts efficiency if the transaction validation needs information from distinct shards [197]. When this is the case, the need for synchronizing information between different shards produces communication overhead and extra latency. The overhead is proportional to the number of shards and the efficiency of the transaction allocation procedure. For example, some works estimate that the number of transactions shared between shards can exceed 90% of the total transactions when the number of shards is greater than 64, and nodes are randomly allocated to the shards [42], [178]. Thus, similar to traditional big data techniques that parallelize tasks in centralized clusters, sharding proposals must allocate recurring tasks in the same shards to maximize transaction throughput and avoid bottlenecks caused by inter-shard communication. Nonetheless, the random choice of nodes participating in each shard guarantees greater security.

Elastico is the first public shard-based blockchain system [31]. Each shard in the network performs the validation of

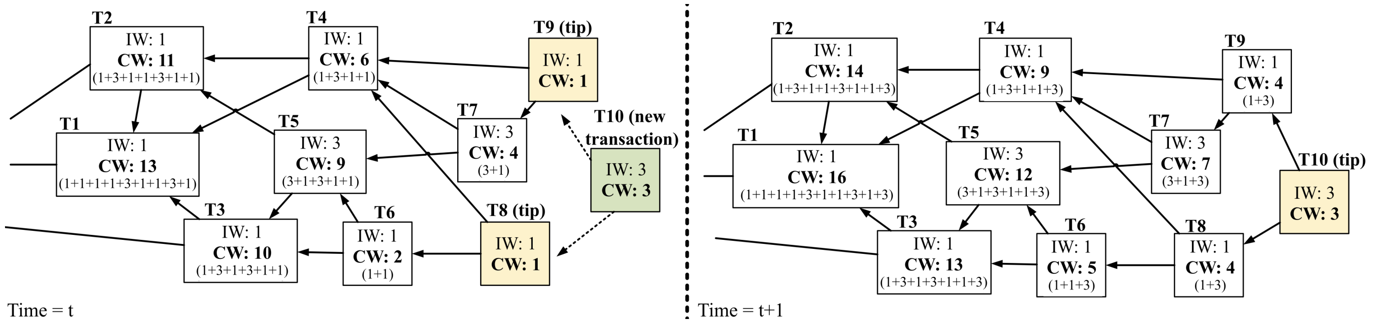

Figure 9. Example of the IOTA Tangle data structure (extracted from [161]). Transactions constitute a directed acyclic graph. Each transaction has an individual weight (IW) and a cumulative weight (CW), corresponding to the sum of the individual weights of all transactions approved directly or indirectly. After selecting and validating two tips, the new transaction, T10, becomes a tip, and its weight propagates to the cumulative weights of previous transactions. The cumulative weight of each transaction serves as a security metric that indicates how many transactions have approved it, similar to the number of block confirmations in a blockchain.

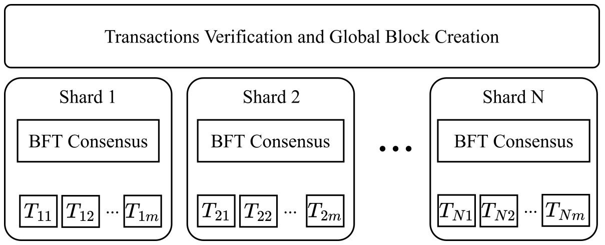
Transactions Verification and Global Block Creation
Shard 1
BFT Consensus
T_{11} T_{12} ··· T_{1m}
Shard 2
BFT Consensus
T_{21} T_{22} ··· T_{2m}
Shard N
BFT Consensus
T_{N1} T_{N2} ··· T_{Nm}

Figure 10. Execution of a consensus round in a sharding-based system. Each fragment generates a set of valid transactions through Byzantine agreement. After this step, the transaction sets are combined to form the global block.

a subset of transactions through the Practical Byzantine Fault Tolerance (PBFT) consensus protocol [164]. The participants of each shard are chosen at each epoch through a proof-of-work challenge to prevent an attacker from creating multiple identities and from corrupting the result of a shard. After this step, a particular shard, called the consensus committee, forms the final block containing the set of all validated transactions from all shards of the network. However, the system uses the benefits of sharding for transaction processing only, while storage and communication issues remain open challenges. Also, the computational overhead to select the shard members at each epoch limits the scalability gain of the proposal.

OmniLedger appeared as an alternative to Elastico which focuses on avoiding its main disadvantages [36]. Omniledger combines the RandHound and Algorand consensus protocols to provide a public collusion-resistant mechanism for randomly choosing the participants of shards [162], [198]. OmniLedger introduces a two-phase lock-and-unlock protocol called Atomix to ensure the atomicity of transactions between distinct shards. Another advantage of OmniLedger is that validators save only a part of the transaction history, which is enough to recreate the state of the shard and reduces storage overhead. In addition, the authors propose replacing the blockchain with a DAG to increase block processing concurrency. However, the use of computationally expensive operations and the need for an intense participation of network nodes in transactions between shards prevents its adoption by low processing power nodes.

Another sharding solution to increase the throughput of blockchain systems is RapidChain [37]. The proposal has greater resilience to Byzantine failures, tolerating up to 1/3 of malicious participants like traditional deterministic Byzantine agreements compared to 1/4 tolerated by Elastico and OmniLedger [31], [36], [199]. The results show that the proposal achieves approximately 4,000 transactions per second for a network with 9 shards, a throughput that is 100 times higher than Elastico and 8 times higher than OmniLedger. Dang et al. [40] propose a shard-based blockchain system using a Trusted Execution Environment (TEE) for shard formation. Specifically, the TEE generates unbiased random values to assign nodes to a shard securely. The authors also leverage TEE to improve BFT consensus protocols by removing equivocation in its failure model, achieving higher fault tolerance than other implementations. The proposal achieves a throughput of 3,000 transactions per second for a network with 36 shards, each containing 4 nodes.

Pyramid [50] is a system that predicts the intersection between distinct shards to reduce the communication overhead of inter-shard transactions. In Pyramid, some shards can store records of multiple other shards. Nodes that host part of the shards referenced in a transaction are responsible for verifying it. Therefore, Pyramid reduces both the computational overhead of complex protocols to ensure the synchronization of information between the shards and the storage of global transactions by all network nodes, restricting the storage of transactions between the shards to the edge nodes. In addition to the highlighted systems, other proposals use sharding to increase transaction throughput, such as Zilliqa [38], MultiVAC [41], and Monoxide [42]. All the sharding proposals still present a maximum throughput considerably lower than traditional payment systems [180]. However, providing scalability, decentralization, and security remains a challenge in research areas that combine blockchain and shards.

3) Cross-chain Protocols and Sidechains: DAG and sharding are proposals that change the blockchain structure or the transaction validation process. Another strategy is to keep the structure of the main blockchain and validate transactions via secondary blockchains [29], [33], [47], [154].

Cross-chain protocols constitute an essential component of sidechains, as they allow mapping the existing resources of a large and slow blockchain into a smaller and faster blockchain [154]. Despite initially being proposed for blockchain interoperability, cross-chain protocols have been used to increase blockchain scalability since they allow assets to be processed in a secondary (and possibly more efficient) blockchain. Cross-chain protocols allow interoperability between blockchains. The objective is to map the existing resources of a blockchain into another blockchain in a coordinated manner that guarantees the uniqueness of assets and avoids double-spend attacks in both blockchains. For instance, we want Alice, who owns resources in blockchain X, to send or receive assets from Bob, a user in blockchain Y. A cross-chain protocol must guarantee the atomic transfer of resources between the two users in the involved systems. There are usually two steps for transferring assets: locking the assets at the source and releasing resources at the destination.

Back et al. [29] introduce the concept of sidechains in the Pegged Sidechain platform. The basic idea of sidechains is to have a main blockchain that synchronizes users in other blockchains (sidechains). Parallel to the main blockchain, secondary blockchains execute consensus protocols that may be completely uncorrelated to the consensus in the main blockchain [47]. Due to the fewer participants, the secondary blockchains usually achieve higher throughput than the main blockchain. Users who need to perform recurrent generic tasks can even create secondary blockchains on demand.

Another advantage of sidechains is that the data published in the main blockchain is just a hash of the current state of the secondary blockchain, saving disk space on the main blockchain. Consequently, sidechains incur lower fees for transactions and executing smart contracts [33]. Finally, when the participants of the secondary chain decide to commit the changes, they transfer the new states from the secondary blockchain to the main blockchain using a cross-chain protocol.

Two-way pegs are a simple way to trade resources between blockchains. The two-way pegs can be implemented by a trusted entity, such as a trusted broker of the parties involved, or by a smart contract associated with the two blockchains [48]. Figure 11 shows an example of an exchange operation. A user of the main blockchain wants to transact on the sidechain. Thus, the user executes a transaction that locks the desired amount of resources with the help of the two-way pegs. Once the two-way peg withholds the resources in the main blockchain, it allows the release of the same amount in the sidechain. After performing the desired transactions on the sidechain, the user can follow the reverse path to receive her/his assets back in the main blockchain. The system scalability has the potential to increase considerably with the existence of sidechains, considering that hashes published in the main blockchain are less frequent.

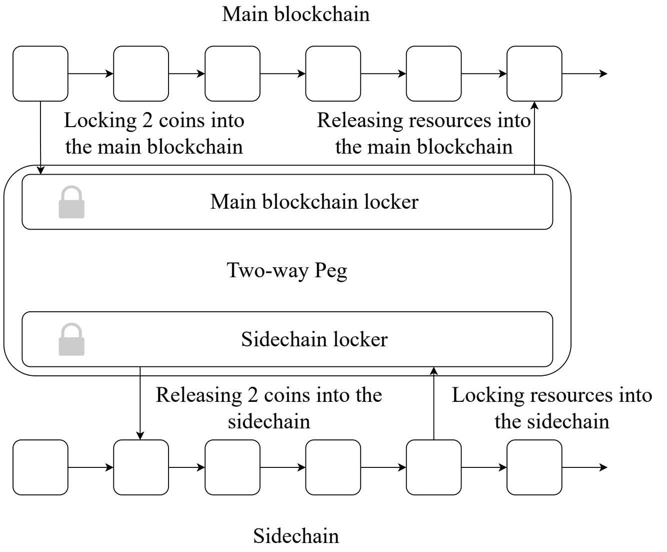
Main blockchain
Locking 2 coins into the main blockchain
Releasing resources into the main blockchain
Main blockchain locker
Two-way Peg
Sidechain locker
Releasing 2 coins into the sidechain
Locking resources into the sidechain
Sidechain

Figure 11. Illustration of a sidechain with a centralized two-way peg to provide interoperability between the blockchains. A user of both blockchains initially locks resources in the main blockchain and obtains them in the sidechain. The two-way peg oversees the two blockchains and ensures the correct transfer of assets. Finally, the user can follow the reverse path to receive the resources in the main blockchain.

The main challenge of moving consensus outside the main blockchain is to ensure security in the side blockchains. Creating smaller blockchains favors attackers, as their influence on the consensus result is inversely proportional to the number of participants or computational power. Thus, the participants of the sidechains must be carefully selected through trust and reputation or by establishing a reasonable number of participants to minimize the risk of attacks. Among the main sidechain implementations, we can highlight: Plasma [33], satellite chains [34], Chainlink [35], and Cosmos [43].

Plasma is an alternative to reduce smart contract execution rates on Ethereum [33]. Smart contracts execute outside the blockchain and publish small results of all executions as a final state in the main blockchain. Li et al. present the concept of satellite chains [34]. The goal is to create blockchains that run independently but can exchange data if necessary. The practical application is an industrial setting. Chainlink [35], [51] is a system that allows the exchange of information between blockchains and external information through smart contracts. Cosmos is a blockchain system that uses a central Cosmos [43] hub and parallel blockchains. Each blockchain runs a BFT protocol based on Tendermint [176] for fast block validation. Furthermore, the authors propose a new protocol for inter-blockchain communication to transfer resources between secondary blockchains.

4) Discussion: Scalability proposals at layer one involve mechanisms that improve consensus performance. DAG systems modify the blockchain structure to enable concurrent

Table IV
COMPARISON BETWEEN THE SCALABILITY SOLUTIONS IN THE LITERATURE.

Scalability Solution	Works	Layer	Description	Main Features	Main Limitations
FPGA	[16], [18]	HW	- Uses FPGA to implement a cache memory and store popular blocks - Uses FPGA to implements a consensus round	- Accelerates access to blocks up to 7× - Accelerates consensus by up to 12×	- Requires an FPGA, which is hard to enforce in decentralized environments
TEE	[15], [77] [14]	HW	- Uses TEE to prevent malicious behavior from participants - Uses TEE to replace the decentralized computational challenge with simple tamper-proof timers	- Eliminates time-consuming security mechanisms - Reduces the time of a consensus round and increases the protocol scalability to up to 2,300 tps	- Requires a TEE, which is hard to enforce in decentralized environments
Block size and interval modification	-	L0	- Increases the size of the block while maintaining the block interval - Reduces the time needed to validate a block	- Increases the average number of transactions in a block - Increases the number of processed transactions per second	- Requires changes in block structure, which may lead to hard-forks - Underlying network link capacity limits gains in scalability
SegWit	[21]	L0	- Reduces the size of transaction - Moves transaction signatures to an external field	- Increases the average number of transactions in a block - Prevents the transaction malleability problem	- Requires change in transaction structure, which may lead to hard-forks - Required transaction information limits gains in scalability
Taproot	[183] [184] [186]	L0	- Reduces on-chain data on multi-signature protocols by adopting Schnorr signatures - Reduces on-chain script data by adopting Merklized Alternative Script Trees	- Increases the average number of transactions in a block - Improves user privacy	- Requires changes in block structure, which may lead to hard-forks - Required transaction information limits gains in scalability
Block compression	[22], [24]	L0	- Reduces network message size by discarding information already on the mempool	- Reduces the bandwidth requirement by up to 80×	- Required transaction information limits gains in scalability
Network modification	[20], [23] [25], [26] [28], [187]	L0	- Improves network infrastructure to reduce propagation time - Executes a kernel-passing to directly transfer data from the NIC to the application's memory	- Reduces block and transaction propagation delay - Avoids memory copies	- Requires changes in network infrastructure, which is hard to enforce in decentralized environments
DAG	[32], [46] [30]	L1	- Replaces the blockchain structure for a directed acyclic graph - Allows a transaction to reference any predecessor transaction - Eliminates distinction between clients and validators	- Eliminates the requirement of a round-based consensus protocol	- Allows forks, which weaken the network security - Requires changes in the blockchain structure, which is hard in existing public blockchains
Sharding	[31], [50] [36]–[38] [40]–[42]	L1	- Distributes transaction data among groups of validators - Parallelizes transaction processing	- Accelerates consensus by splitting the problem	- Requires changes in the consensus protocols, which is hard in existing public blockchains - Introduces communication overhead in inter-shards information exchange
Cross-chain and Sidechains	[29] [33]–[35] [43], [51]	L1	- Delegates transaction processing to a faster secondary blockchain	- Saves storage space on the main blockchain	- Reduces consensus participants on secondary blockchains, which increases centralization and weakens security

and asynchronous transaction processing by any user. This approach allows forks to occur, which makes it vulnerable to double-spending attacks when users do not share the same view of the blockchain. Sharding methods attempt to speed up consensus performance by concurrently processing subsets of transactions on different validator groups. This approach faces challenges similar to big-data techniques, such as optimal distribution of transactions and inter-shard communication. Cross-chain protocols and sidechains, on the other hand, move transaction validation to secondary blockchains that are smaller and faster. This approach is primarily fit for blockchain applications in a large blockchain system, such as ERC-20 tokens on Ethereum [200].

D. Summary of Scalability in Layers HW, L0, and L1

Scalability proposals for layers HW, L0, and L1 usually involve structural modifications that are difficult to implement. Table IV summarizes the surveyed proposals on the hardware layer and layers 0 and 1. Changes to the hardware layer imply extra costs to users and cannot be enforced in decentralized environments. Overall, hardware layer solutions prioritize scalability and security over decentralization in the blockchain trilemma (described in Section III-B). Changes to layer zero alter the format of messages, causing soft forks or even hard forks in the blockchain. For an end user, this can incur incompatibility issues with some nodes on the network that prevent them from exchanging messages with nodes running different blockchain versions. Furthermore, changes on layer zero present limited scalability gain, given that a transaction or signature cannot be reduced indefinitely without a loss in security. Finally, improvements in layer one often create new systems that modify consensus and even the blockchain structure itself [32], [174], [176]. Consequently, layer one solutions are hard to integrate into existing blockchain systems.

V. Scalability in Layer Two: Payment Channel Networks

Unlike proposals on layers HW, L0, and L1, layer-two solutions do not modify blockchain systems at all since they operate on top of the lower layers. They leverage blockchain core functionalities, such as scripts and smart contracts, to implement security mechanisms that validate off-chain transactions. Such validation mechanisms are invisible to the blockchain, which sees transactions from a layer-two service as ordinary transactions. With this simple but powerful characteristic, layer-two proposals have emerged in recent years as solutions with great potential to scale blockchains efficiently without causing a significant impact on end users.

Before presenting layer-two solutions, we note that the definition of layer-two protocols is still under discussion. For some authors, any mechanism that avoids publishing transactions in the main blockchain is considered layer two, as such solutions neither rely on the primary consensus protocol to process transactions nor make all transaction data publicly available [145], [147], [149], [154], [156], [158]. This perspective classifies proposals that validate transactions through secondary blockchains, such as sidechains, as layer-two protocols. Conversely, other authors define layer-two protocols as protocols that implement their own off-chain consensus-free validation rules [13], [109], [143], [157]. Under this definition, layer-two protocols only rely on consensus to settle disputes that could not be solved via off-chain validation.

We adopt the latter concept as it enhances the innovation of off-chain validation. Besides, associating consensus to layer one provides a clearer separation between layers. In particular, we adopt the definition of layer-two protocols as proposed by Gudgeon et al. [13]:

Concept definition 2. *(Layer-two protocols). A layer-two protocol is a protocol that allows transactions between users through the exchange of authenticated messages via a medium that is outside of but linked to a layer-one blockchain. Authenticated assertions are submitted to the main chain only in cases of a dispute, with the main chain deciding the outcome of the dispute. Security and non-custodial properties of a layer-two protocol rely on the consensus protocol of the main chain.*

Hence, we classify proposals that modify consensus or use consensus as the default validation mechanism as layer one, including sidechains and cross-chain protocols. To the best of our knowledge, *payment channel networks*, *optimistic rollups*, and *zero-knowledge rollups* are the only currently known layer-two protocols for blockchain scalability.

In the following sections, we dedicate a large portion of this work to the presentation and discussion of the main challenges of payment channel networks, the most popular layer-two protocol for exchanging assets efficiently in traditional cryptocurrencies such as Bitcoin. The PCN use case covers most of the characteristics of layer-two protocols, so the reader should have gathered enough understanding after Section VI. Then, for completeness, we review the different types of transaction rollups, an innovative mechanism for processing generic off-chain computations in blockchain systems that support smart contracts, such as Ethereum.

A. Overview of Payment Channels

Payment channels are a solution to the blockchain scalability problem [201]. Unlike the solutions mentioned in previous sections, this technology operates in layer two, establishing off-chain communication channels in which users can freely send payments without validating them through a slow consensus protocol. This solution reduces the latency of payments, which now depends mainly on communication latency between users. Payment channels are a particular case of state channels in which the traded assets are coins [202].

The main idea of payment channels is to publish transactions in the blockchain only when needed. The blockchain becomes a security service that provides the starting point for payment channels and possibly is used to solve disputes involving users. Figure 12 shows the three-phase operation of a payment channel. In the channel creation phase, two users, Alice and Bob, issue a funding transaction, agreeing to transfer some of their coins to a common address cooperatively controlled by both. These coins remain unavailable in the blockchain for the entire channel lifespan but can be used to issue transactions in the channel. Bob and Alice also agree on a timelock window W that either user must wait in case he/she unilaterally closes the channel (we explain this requirement in Section V-B3). Then, when the channel is established, the users continuously update their balances in the channel via private commitment transactions. Commitment transactions are not published in the blockchain and, consequently, produce low-latency payments. Either user can trigger the channel closing phase at any time by publishing the latest commitment transaction into the blockchain. Once validated, this transaction creates an Unspent Transaction Output (UTXO) that transfers the most recent balances to their respective parties on-chain. Thus, the blockchain only knows the channel-opening and channel-closing transactions, which are seen as ordinary transactions in the system. Consequently, these are the only transactions that go through the consensus protocol and are subject to high latency and transaction fees.

One of the main advantages of payment channels is easy implementation. While some layer-one solutions like sharding and more efficient consensus protocols require core changes to the blockchain, payment channels do not require any modification at all. In fact, unlike other layer-two solutions such as transaction rollups (Section VII), payment channels can be implemented in blockchain systems that do not support Turing-complete smart contracts, such as Bitcoin. Another key advantage of this technology is that it saves resources that would be spent on transaction fees. In public blockchains, the low transaction throughput causes users to pay high fees for a transaction to be prioritized by validators. Payment channels remove transaction fees, as transactions are sent directly to the receiver. The only exceptions to this are the channel-opening and channel-closing transactions.

B. Guaranteeing Security in Payment Channels

Payment channels, like blockchains, do not require mutual trust in their security model. To accomplish this, the channel establishment procedure works as follows. When establishing a channel, the funding transaction implements a 2-of-2 multi-signature payment policy, i.e., each new transaction that spends the funds of the funding transaction must contain the signature of both users. To issue a commitment transaction within the channel, a user creates a transaction that spends coins from the funding transaction, signs it, and transfers the signed transaction to the other party. This signed transaction serves as a payment guarantee to the other party, who can sign and issue the transaction on the blockchain to redeem his funds. Thus, it is not possible for one of the parties to issue transactions to itself and steal coins from the channel since an agreement between the two users is required.

Locking coins in a transaction that requires the signature of both participants, however, introduces a vulnerability. Assuming a channel formed by users A and B, user B may act maliciously by refusing to issue signed transactions and also refusing to sign transactions issued by A. This way, A's coins are forever stuck in the payment channel and unavailable to be used in the blockchain. To address this potential vulnerability, users A and B must generate a refund transaction, which guarantees the user a refund in case of malicious behavior by the counterpart. This transaction is created before the channel is established and exchanged by users. Thus, user B has a refund transaction signed by user A, and user A has a refund transaction signed by user B. In the event of a malicious action by one of the parties, the counterpart issues the refund transaction on the blockchain and reclaims their coins. The refund transaction only serves as an initial guarantee and is no longer valid after the first transaction in the payment channel. For this invalidation to occur safely, the system must correctly revoke the latest state and introduce a new one, updating the channel state.

1) Updating Channel States: A key aspect of payment channel security is channel state updates [13], [54]. If channel updates are incorrectly performed, a malicious user can take advantage of his counterpart by publishing an old state that benefits him. In the example of Figure 12, users perform three off-chain transactions: Alice issues the first commitment transaction, TX_1, sending 2 coins to Bob; then, Bob issues a second transaction, TX_2, sending 1 coin to Alice. Finally, Alice issues a third transaction, TX_3, transferring 3 coins to Bob. In this case, Alice can act maliciously and close the channel by publishing the second transaction, in which she had a balance of 4 coins instead of the current balance of 1 coin. Also, Alice could act maliciously by sending the refund transaction to the blockchain, redeeming the 5 coins she initially owned instead of the 1 coin she currently owns. This vulnerability happens because the blockchain does not know which transaction is in the most recent state since it does not keep track of off-chain transactions. Payment channels, however, do not need to maintain a complete ordering of off-chain transactions like blockchains; it suffices to keep track of the latest state. In the example, it is enough to enforce that only TX_2 is valid, i.e., there must be a way to revoke TX_1 and consider TX_2 as the current state. We explain how to revoke old states in the following sections.

2) One-way Payment Channels: Several proposals have been made to solve the problem of state updates in payment channels [52]–[54]. In 2013, Spilman [52], [203] proposed one of the first state update mechanisms for Bitcoinj[7] payment channels, which only works for one-way payments. This mechanism imposes conditions for the redemption of coins using the programming language of Bitcoin, the Bitcoin

[7] Bitcoinj is a Java implementation of the Bitcoin protocol. Available at https://bitcoinj.org/

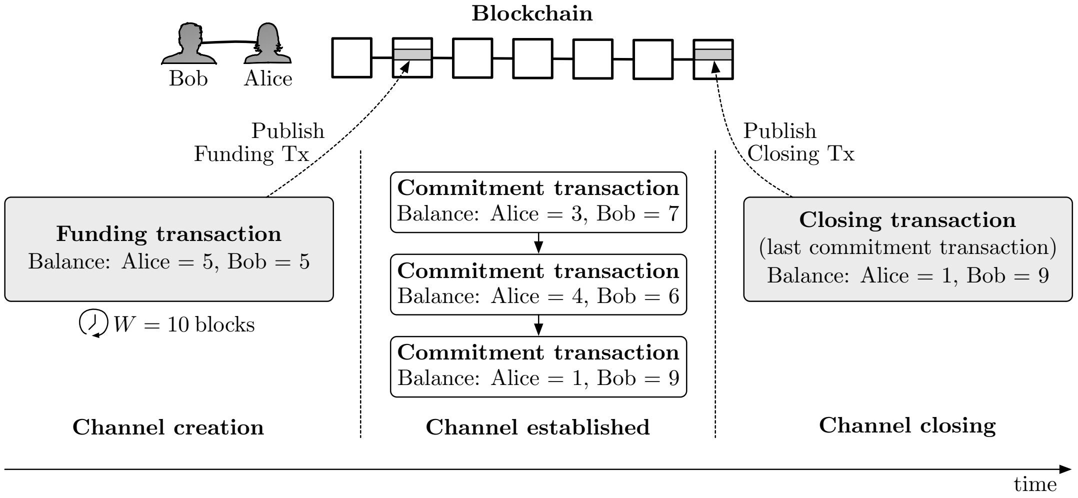

Figure 12. Payment channel operation. Users Alice and Bob contribute 5 coins each to issue the funding transaction and create a payment channel. After establishing the channel, Alice and Bob can exchange coins by issuing private commitment transactions that rewrite their balances. For example, Alice sends 2 coins to Bob by signing the first commitment transaction, which changes her balance to 3. To close the channel and claim the coins on-chain, Alice or Bob can publish the commitment transaction containing the most up-to-date balance.

script [204]. In the proposed mechanism, before establishing a channel, users A and B generate a refund transaction that contains the following conditions to close the channel: (i) the locked coins can be redeemed after time t counted from the publication of the funding transaction, or (ii) the coins can be redeemed immediately if the counterpart agrees to the refund. The first condition guarantees that coins cannot be locked in the channel forever, while the second condition guarantees that the two users get a refund immediately if there is a transaction signed by both, proving they agree with it. After channel establishment, user A starts to send coins by issuing commitment transactions to user B, who can sign and publish the transactions in the blockchain, close the channel, or wait for a new state change. It is easy to see that this channel design works only as a one-way channel. While user B receives transactions signed by A, the same does not happen in the opposite direction. Furthermore, even if user B returns a coin to user A, there is no guarantee that B will not publish an old transaction on the blockchain in which he had a higher balance. Thus, the correction of state replacement is incentive-based since the user who receives the coins has nothing to gain by publishing an old transaction. Any rational user who receives payments will always post the latest state as it benefits them the most [13].

The creation of a payment channel that works only in one direction, however, restricts the potential applications of this technology. Most day-to-day applications require payments in both directions, e.g., refunds for purchases or cashback applications. Furthermore, two users who share a payment channel can take on independent roles, buying and selling products to each other. In this case, one-way channels eliminate the possibility of safely sending payments in opposite directions. Users who want to reverse the roles of sender and receiver would have to create one channel in each way, which implies more locked coins, more transaction fees, and more time to establish the channels.

3) Bidirectional Payment Channels and State Revocation: Creating bidirectional channels involves state revocation: revoking all previous states from both parties is necessary to prevent malicious action. Nevertheless, blockchains do not allow the creation of revocable transactions [54]. Once signed and posted on the blockchain, the transaction cannot be canceled. It is possible, however, to create policies that discourage users from issuing old transactions and acting maliciously [53], [54]. One of the early forms of state replacement was a time-blocking replacement. In this model, all transactions in the channel have a time lock, i.e., the coins in the transaction can only be spent after a time window W counted from its publication in the blockchain. This model uses the blockchain as a reference to guarantee synchronization, with the time window defined in terms of the number of blocks in the blockchain. The time window decreases each time the payment direction reverses. Thus, in a channel between two users A and B, user A can issue a transaction TX_1 to B with a 30-minute time lock, approximately 3 blocks on the Bitcoin network (recall that one block takes on average ten minutes, see Section IV-B1). If B wants to pay A, B issues TX_2 with a time lock of 20 minutes, approximately 2 blocks in the Bitcoin network. That way, if B acts maliciously and issues the previous transaction, TX_1, in the blockchain, user A can issue TX_2 and redeem the coins before B since TX_2 has a shorter time lock. Despite safely ensuring state replacement, this model has two disadvantages: (i) user A must be online and constantly checking the blockchain to monitor the actions of B; and (ii) W is limited. The initial value of W, defined by the two users that create the channel, is a difficult choice. A high value for W allows for more state updates but locks coins longer when a user closes the payment channel. A low value of W allows few channel updates, reducing the channel expiration date.

Poon and Dryja [54] proposed a state substitution model that has become the standard for payment channels. The model creates bidirectional channels while discouraging malicious behavior through financial punishment. Suppose it is proven that a user acted maliciously by publishing an old state in the blockchain. In that case, the channel counterpart can contest the transaction and redeem all the coins in the channel, including the attacker's. Each payment generates a pair of asymmetric commitment transactions: user A has a commitment transaction signed by user B, and user B has a commitment transaction signed by user A. Each transaction has a corresponding secret with the following condition: if the counterpart knows the secret and reveals it within a period of t from the transaction publication, it can redeem all the coins in the channel. In practice, the transaction secret is a private key that transfers the commitment transaction outputs to the address of the user who did not publish the transaction. The key verification is done automatically using Bitcoin script as a root of trust, public to both participants. To issue new payments on the channel, users change the state by exchanging signatures and revealing the transaction secret associated with the previous state. Thus, if any user publishes an old state, the other user can immediately claim all the coins using the previously-revealed private key. In contrast, the user who closed the channel can only redeem coins after W blocks.

The procedure for replacing states executes as follows. Assume a channel between Alice and Bob in which, initially, each user has 5 coins and Alice wants to send 1 coin to Bob. To send the payment, Alice generates an asymmetric key pair (PK_A, SK_A), where PK_A represents Alice's public key and SK_A represents the secret key she generated. Bob performs the same procedure, generating a key pair (PK_B, SK_B). Alice then creates a commitment transaction TX_{AB} containing the following conditions: (i) Alice has 4 coins that she can claim immediately if the transaction is published; (ii) Bob has 6 coins, which can only be claimed after a period W or if Alice authorizes it before the period ends; (iii) the public key of this transaction is PK_B and Alice can redeem Bob's 6 coins if she reveals SK_B in time $t_i, t_i < W$. Alice signs the transaction TX_{AB} and sends it to Bob with the secret key of the previous state. Bob performs the same procedure, generating the transaction TX_{BA}, which contains the following conditions: (i) Bob has 6 coins that he can claim immediately after the transaction is published; (ii) Alice has 4 coins, which can only be claimed after a period W or if Bob authorizes it before the period ends; (iii) the public key of this transaction is PK_A and Bob can redeem Alice's 4 coins if he reveals SK_A in time $t_i, t_i < W$. Bob signs the transaction TX_{BA}, sends it to Alice, and reveals the transaction's secret key for the previous state. Thus, if either party publishes the latest state, only conditions (i) and (ii) can be triggered since the secret keys SK_A and SK_B will be shared in the next state update. However, if an old state is published, condition (iii) can be triggered immediately as the previous secret keys have already been revealed.

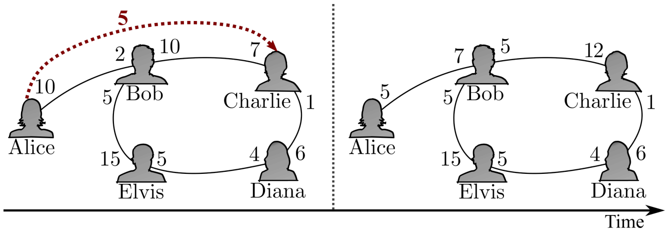

Figure 13. Example of payment in a payment channel network. To send Charlie 5 coins, Alice uses the existing channel with Bob, who forwards the coins to the destination. The payment modifies the balance of the channels involved in the payment path. Hashed Timelock Contracts (HTLC) guarantee the security of this procedure.

C. Building Payment Channel Networks with Hashed Timelock Contracts (HTLC)

Despite providing fast and secure payments, a payment channel can only be established between a pair of users. Thus, more than this solution is needed to solve blockchains' scalability problem as it would require users to establish channels to every payment destination [54]. As a consequence, users who wish to transact with multiple destinations must: (i) have a large number of coins which will be locked in the blockchain to allocate funds across multiple channels; and (ii) pay high transaction fees on each funding transaction and wait for consensus to open the channels. Such requirements hinder the use of this technology in everyday life, in which fast payments to several different entities are common.

The scalability problem can be solved with a *payment channel network* (PCN), which interconnects the existing payment channels created by users in the blockchain. PCNs allow users to transact with each other quickly with low fees, even if they do not share a direct channel. Figure 13 shows an example of a PCN with 5 participants. In the picture, Alice wants to transfer 5 coins to Charlie, with whom she does not have a payment channel. Alice, however, has a channel with Bob, which has a channel with Charlie. Hence, Alice can transfer 5 coins in her channel with Bob, who then transfers 5 coins in his channel with Charlie, completing Alice's payment. Note that payment channels are independent, so Bob cannot transfer the coins from his channel with Alice to the channel with Charlie. Bob receives Alice's commitment transaction on his channel with Alice and generates a commitment transaction of equal value for Charlie.

Payment channel networks must also provide mutual trust between the participants. In the example of Figure 13, Bob could act maliciously by receiving Alice's coins without forwarding the payment on his channel with Charlie. To prevent this, a particular type of contract called a Hashed

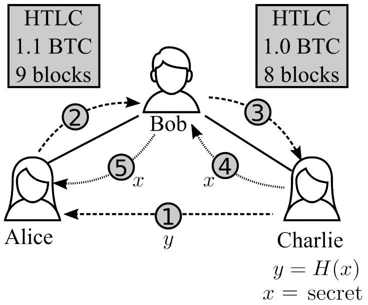

Figure 14. Steps to make a payment on a payment channel network. 1) Charlie, the receiver, generates a secret and sends the hash of this secret to Alice. 2) Alice does not have a channel with Charlie and uses Bob as an intermediary, establishing a contract that promises to deliver coins to Bob if Bob reveals the secret generated by Charlie. 3) Bob performs the same procedure with Charlie. 4) Charlie reveals the secret to Bob and claims his coins. 5) Bob reveals Charlie's secret to Alice, claiming his coins and finishing the payment.

Timelock Contract (HTLC) guarantees that coins in a channel will only be transferred if two conditions are met [54], [205]:

1) **Secret disclosure.** Bob only receives Alice's coins if he reveals (to Alice) the secret that generated the hash included in the contract.
2) **Time limit.** Bob must reveal the secret within a limited time, or else Alice considers Bob gave up the contract and claims the coins back.

For example, assume Alice wants to send coins to Charlie, using Bob as an intermediary. Using HTLCs, Alice only gives the coins to Bob after she learns that Bob forwarded the coins to Charlie. The proof that Bob forwarded the coins to Charlie is the secret that Charlie generated. Charlie only gives the secret to Bob after he receives the coins, i.e., when he is sure Bob has kept his promise. Then Bob can relay the secret to Alice, proving that he kept his promise, and receive Alice's coins. The secret generated by Charlie works like a digital receipt. Each HTLC contains a value $y = H(x)$, where $H(x)$ is the result of the hash function of a secret x generated by the receiver and a timeout t. HTLCs can be redeemed by revealing the secret x or canceled if no secret is revealed before t. In the meantime, the coins are locked (or in-flight). Each intermediary user creates an HTLC with the next hop containing the same y value, creating a payment chain. The receiver, who knows x, reveals the secret to the last hop and triggers the backward unlocking of the payment chain.

In Figure 14, imagine Alice wants to send a coin to Charlie. Charlie generates a random value x, calculates its hash, $y = H(x)$, and sends it to Alice. Alice establishes an HTLC with Bob, informs Bob that Charlie is the next hop and promises to deliver a coin to Bob if Bob reveals x within 9 blocks (approximately 90 minutes in Bitcoin). Bob then generates an HTLC with the same condition for Charlie and sets the timeout to 8 blocks (approximately 80 minutes in Bitcoin). Charlie, who initially generated the x value, reveals x to Bob to redeem his payment. Bob, in turn, reveals x to Alice to redeem the payment in the channel between him and Alice, completing the payment. The maximum amount of coins a node can forward in an HTLC is bounded by the number of coins the node previously allocated in the payment channel. In the example of Figure 13, Bob could not forward payments of more than 10 coins in the channel with Charlie.

HTLCs are the core enablers of multi-hop payments, which, in turn, are what make payment channel networks effective. Due to the two conditions that must be met, HTLCs are said to be locked by time and hash. More importantly, *HTLCs establish trust among users in the payment channel network by guaranteeing that each payment can only be claimed by revealing the payment's secret or by releasing the locked coins after a timeout.*

Note that despite being called contracts, the mechanism of HTLCs follows a simple logic that can be implemented even in systems that do not support smart contracts. For example, HTLCs are implemented in Bitcoin as simple if-then-else clauses inside the script of commitment transactions [54]. This simplicity increases the applicability of payment channel networks, which could be implemented in any blockchain system.

Off-chain payments vs. On-chain transactions. Now that we have presented the concept of HTLCs, it is useful to precisely define the concept of an *off-chain payment*:

Concept definition 3. *(Off-chain payment). An off-chain payment, or simply a payment, is the off-chain transfer of assets between a sender and a receiver through the channels of a payment channel network. While in-flight, the payment establishes a sequence of HTLCs, called a payment chain, in the channels of the payment path. Each HTLC may be claimed by the respective channel parties on-chain.*

Such payment definition is needed to clarify the difference between payments and transactions, which we omitted in the previous sections for simplicity. On the one hand, a transaction is the signed data exchanged between two parties that can be published to transfer assets on-chain, e.g., commitment transactions in a channel. On the other hand, a payment is an off-chain transfer that needs an end-to-end sequence of HTLCs to occur. Because HTLCs reside inside commitment transactions on each channel, a multi-hop payment involves one transaction per channel in the payment path. Henceforth, we refer to this definition whenever we mention the terms "payment" or "off-chain payment".

It is important to note that each intermediary in the path of a payment receives a routing fee[8] which serves as an incentive to forward payments. This routing fee is equal to the difference between the value of the incoming HTLC and the value of the outgoing HTLC (respectively, the HTLC from Alice to Bob and the HTLC from Bob to Charlie in Figure 14). Senders pay fees by setting the value of the

[8] Routing fees or payment fees are off-chain fees charged by intermediaries to forward payments. They should not be confused with the transaction fees needed to publish a transaction in the blockchain.

first HTLC as the sum of the payment value and all the routing fees in the path. Then, the following HTLCs subtract the routing fees per hop. This procedure ensures that every hop will receive more than they have forwarded when the payment is completed. In addition, each intermediary in the payment chain must set a timeout shorter than the previous hop timeout. This difference ensures that all intermediaries in the payment path have enough time to redeem coins. Thus, if Alice generates an HTLC with timeout t with Bob, Bob must generate an HTLC with timeout $t_i < t$ with Charlie. Otherwise, Bob risks losing funds.

D. PCNs: The Lightning Network

The Lightning Network, proposed by Poon and Dryja [54] for Bitcoin in 2016, was the first implementation of the PCN technology. The system issued its first channel-opening transaction in 2017. As of February 2023, it is the largest known payment channel network with more than 16,000 nodes and 80,000 payment channels distributed around the world [143], [148], [149].

The number of nodes and channels tripled from January 2020 to August 2021, showing rapid network growth. Thus, the Lightning Network is the reference implementation of a payment channel network today, providing real-time digital payments for several applications [206]. The development of the Lightning Network is even mentioned as a key factor for the adoption of Bitcoin as an official payment method in El Salvador [207].

Lightning has a strong focus on providing user anonymity, just like Bitcoin. As such, it implements several privacy-preserving mechanisms, such as user identification through public keys only. Payments use onion routing [208], a transport protocol that encrypts packets at every hop, mostly known for its Tor network implementation. This mechanism ensures that no payment intermediary knows the full payment path. Intermediaries charge two types of routing fees when forwarding payments: a base fee and a proportional fee[9] [210]. Base fees are a fixed amount charged to each forwarded payment, regardless of the payment value. Proportional fees are charged on top of the amount to be forwarded. Despite featuring two types of routing fees, Lightning's fees are orders of magnitude lower than the fees charged for publishing a transaction on Bitcoin, making it especially attractive for real-time micropayments [211].

The network implements bidirectional payment channels precisely as described in Section V-B [54]. However, users have the option to publicly announce their channels to be used as payment routes, or keep them private to ensure greater privacy. The disclosure of channels in the network occurs through channel announcement messages that the network broadcasts in gossip fashion [212]. In this model, a participant broadcasts a message to a specific number of neighbors, who repeat the message to their neighbors until the entire network knows the channel. The network nodes build the network topology containing the active channels with their respective participants and capacities from the received messages. Private channels are not advertised and therefore do not appear in the topology. Thus, the number of channels a node knows is a lower limit than the actual number of channels in the network, which is difficult to estimate. Regardless of channel status, channel balances are known only to the channel parties, while channel capacity may or may not be disclosed. On the one hand, this prevents outside observers from tracking payments by monitoring channel balances, compromising user privacy. On the other hand, the unknown balance hinders choosing paths with sufficient funds to complete a specific payment.

Lightning standardizes the message formats and protocols used in the network through the Basis of Lightning Technology (BOLT) [209], documents inspired by Internet RFCs that formally describe how the network should be implemented. Currently, the Lightning Network provides 11 BOLTs, which specify the format and exchange of messages for opening and closing channels, payment encoding, routing protocols, and other specifications.

As the main PCN, several works analyze the Lightning Network. Seres et al. [87] analyze the topology of the Lightning Network in January 2019. The results show a strong centralization, with few nodes concentrating most channels in the network. The authors analyze the robustness of the network, simulating attacks directed at the nodes that present the highest degree. They show that removing the top 37 nodes reduces the network capacity by 50%. Similarly, Lin et al. [88] assess the income concentration in the Lightning Network from January 2018 to July 2019. The income concentration of a node in the network can be measured by checking the public capacity values of the channels in which the node participates. The authors verify a tendency of centralization around the higher degree nodes, forming core-periphery structures in which the hubs form the core and the periphery presents star-like substructures. The results show that removing the hubs partitions the network into multiple components, making it vulnerable to topology-based attacks [79].

E. PCNs: Other Implementations

Other PCNs have also been proposed over the last few years with some differences to Lightning [57], [63], [89]. We highlight the most popular ones next.

1) Raiden: The most relevant alternative to the Lightning Network is Raiden, an off-chain payment channel network for the Ethereum blockchain [89]. Raiden, defined by its developers as "Ethereum's version of Bitcoin's Lightning Network", implements operations that are very similar to Lightning's. Firstly, the users must provide balance proofs, locking funds publicly in the blockchain. The tokens to be locked are deposited in a smart contract that handles the Raiden transaction logic. After this phase, users can send payments for as long as they have sufficient funds. An HTLC

[9] The proportional fee is also named "fee rate" in the documentation of Lightning [209].

secures the payments, similarly to Lightning. However, an advantage of Raiden over Lightning is the support for trading tokens in general instead of only coins. It is possible to create a Raiden network per cryptocurrency that operates on top of Ethereum. There are nearly 15 thousand addresses that hold Raiden tokens, accounting for a total of $3.71M [213], [214]. Today's largest Raiden network is still under development and contains 128 nodes and 148 channels [215].

2) Sprites and Pisa Sprites: Sprites [57] is a PCN and state channels implementation (recall that state channels are a generalization of payment channels) that differs from Lightning by proposing constant lock times between linked payments and partial withdrawals/deposits. The constant lock time is obtained with a smart contract, modifying the current HTLC notion of disputes. The difference is that Lightning resolves disputes in a serial manner, one channel at a time, while the smart contract of Sprites can do this procedure concurrently. The notion of partial withdrawals/deposits in a channel allows users to update the channel capacity on the fly, without closing the channel. Thus, it reduces the need for costly on-chain operations. This feature is also possible due to the smart contract logic. Pisa improves Sprites by introducing a custodian service that allows users to disconnect for long periods without risking losing funds [76]. The custodian is similar to the watchtowers in Lightning, which alleviates the assumption that parties must always be online to guarantee security in the final state channel result. Nevertheless, Sprites and Pisa Sprites only support the Ethereum network as the underlying blockchain because they need smart contracts to implement the channel logic. The projects also lack large-scale adoption by the cryptocurrency community.

3) Trinity: Trinity [63] is a state channel implementation on the cryptocurrency Neo. It has the same purpose as Lightning: to improve throughput and offload the blockchain by executing transactions off-chain. The blockchain is used only to open the state channel, publish the final state, and solve disputes. Participants in the network must provide a "proof of assets" to execute transactions off-chain, which is similar to reserving some financial resources in the blockchain. Neo, however, has a small market value since 1 NEO is worth 0.0004 BTC, and the Trinity project has yet to release a new version since 2018 [216].

4) Blind Off-chain Lightweight Transactions (BOLT): BOLT [58] is a PCN proposed to guarantee anonymous payments, designed for ZCash and other anonymous cryptocurrencies. Anonymity is enforced with technologies such as pseudorandom functions and non-interactive zero-knowledge proofs. BOLT defines three types of channels: unidirectional, bidirectional, and indirect. Unidirectional channels allow a consumer to pay with recurrence to a merchant. Bidirectional channels are used for recurrent payments between two parties in both directions. Finally, indirect channels allow users to make payments even though they do not share a direct channel. In this case, users send payments through untrusted intermediaries using a revocation scheme similar to commitment transactions on Lightning. BOLT does not provide a large-scale implementation, thus it is difficult to assess its advantages and drawbacks in practice.

5) TumbleBit: TumbleBit [59] is a unidirectional payment channel hub compatible with the Bitcoin protocol. TumbleBit maintains user privacy using an intermediary called Tumbler, which cannot link the payments between the parties involved in transactions. The Tumbler is a special user that mixes the received transactions with cryptographic techniques to ensure that the blockchain will not record any information about transactions between users. Users can also use Tumblebit as a mixer to ensure privacy in non-recurrent transactions. Tumblebit is centralized and does not support payment forwarding through generic untrusted intermediaries.

6) Teechain: Teechain [15], [77] is a PCN that uses Trusted Execution Environments (TEE) to provide security guarantees for off-chain transactions. Teechain allows disputes to be solved asynchronously instead of during a dispute period. To accomplish this, Teechain moves the root of trust from the blockchain to TEEs, ensuring that the nodes always act honestly. The architecture uses special CPU instructions provided by Intel's Software Guard Extensions® (SGX) to create memory regions called enclaves isolated from the operating system to implement these trusted environments. In the proposed system, the client application maintains deposits within enclaves and manipulates the balances of these deposits when receiving and sending transactions through payment channels. The application with enclaves only interacts with the blockchain at deposit creation and completion. Before creating a deposit, the user must utilize an Intel attestation mechanism to ensure that the application runs in a genuinely trusted environment and that it will honestly follow the protocol. Furthermore, the system replicates data between device enclaves that participate in a committee, to prevent a single point of failure. Teechain is compatible with the Bitcoin network, but instead of using HTLC to transfer assets in the PCN, Teechain locks the funds in the TEE. The remaining operations are similar to Lightning. Teechain, however, has the disadvantage of needing specific hardware. Thus, like proposals on the hardware layer, the adoption of this payment channel network is limited by the cost to the end user.

7) Comparison with Lightning: Despite many PCN proposals, the Lightning Network is by far the most widely adopted and the reference of a PCN implementation. Other PCN proposals often lack large-scale implementations and community validation, hindering their adoption and gain of market share. The Lightning Network often incorporates several ideas from alternative PCNs, centralizing the development of a canonical PCN model. Hence, in the next section, we refer to Lightning as the default example of a payment channel network and address the main PCN open challenges.

VI. Challenges in Payment Channel Networks

PCNs are a new technology that still needs to be extensively studied. Many research questions regard the large-scale adoption of PCNs. This section addresses the main open challenges of PCNs and presents several efforts found in the literature to solve them. We separate the challenges into topics: (i) payment routing; (ii) channel rebalancing; (iii) network design; (iv) security and privacy; (v) attacks in PCNs; (vi) payment concurrency; (vii) payment load balancing, congestion control, and scheduling; (viii) PCN simulation; and (ix) support for light nodes.

A. Payment Routing

Although payment channel networks are peer-to-peer networks, the way payment channels work is unique regarding payment routing. The main specificity of payment channels is that recurrent coin transfers from one party to another in the same channel indefinitely reduce its capacity in that direction since each forward operation moves some balance of the sending party to the receiving party. Therefore, the forwarding capacity of a channel directly depends on how many payments have already been sent. This particularity is the main difference of routing in payment channel networks compared to traditional datagram networks, in which forwarding packets reduces the capacity of a link only while packets are in transit. For example, in a traditional packet-switched network, even though a 1 Gb/s link is forwarding packets at 200 Mb/s and its available capacity is temporarily reduced to approximately 800 Mb/s, the capacity will return to the original value once the packets are transmitted. In contrast, if a payment channel with a capacity of 1,000 coins forwards 200 coins in a specific direction, its capacity in that direction is reduced to 800 coins unless there is a transfer of coins in the opposite direction. This characteristic highlights the need to balance payments in each direction and makes it challenging to use maximum flow approaches for routing, which are commonplace in packet-switched networks.

Another critical characteristic of payment channel networks is that the balances of each channel are private, i.e., only the two parties directly involved in the channel know its current state. The total channel capacity, on the other hand, is public and available on the blockchain. This creates a challenge for routing algorithms: estimate the current state of channels from their total capacity. If this is not done correctly, the path choice algorithm may use channels that had a sufficient balance at the opening time but have already been exhausted by other payments. In this case, the payment fails and must be attempted again, increasing the user's latency and reducing system efficiency. In protocols that split the payment into multiple paths, failure of part of a payment also compromises the atomicity of the payment. This can lead to situations where the buyer correctly pays n coins for a product, but the seller only receives $n - x$ coins, where x is the sum of the values of the failed payments.

1) Routing in the Lightning Network (Default Protocol): The standard procedure for discovering payment paths in the Lightning Network uses a trial-and-error single-path protocol based on Dijkstra's algorithm for selecting the shortest atomic path. The user runs Dijkstra's algorithm to find the shortest path to the receiver and then tries to make the payment on that path. Suppose the payment fails in some channel because an intermediary does not have enough balance to forward the payment. In that case, the protocol removes the channel in which the payment failed from the graph and reruns the Dijkstra algorithm. The Lightning Network defines the sum of the fees to be paid in a path as the standard shortest path metric, i.e., the algorithm selects the path that minimizes routing fees. This path-choosing procedure intentionally sacrifices optimality to provide speed, as it considers most paths will succeed. Nevertheless, this assumption may not hold true for high-value payments. The protocol also tends to imbalance the most central channels in the network, creating the need for constant rebalancing operations. In practice, the default LN protocol is highly dependent on the current topology of the Lightning Network, with a small set of channels that hold a high amount of allocated coins and are rebalanced frequently [87]. It is unclear how the protocol would perform in the long term if the network topology changes.

2) Atomic Multipath Payments: Atomic Multipath Payments (AMP) are the default multipath routing protocol for the Lightning Network [67]. The primary goal of AMP is to enhance the reliability, efficiency, and privacy of payments by splitting them into smaller parts that can be routed through different paths in the network. This way, users can send large payments through small channels without incurring large failure probabilities. In the example of Figure 13, the maximum amount of payments Alice can transfer to Charlie using the standard (single-path) LN protocol is limited by the channel with the lowest capacity, which has 10 coins. Thus, Alice cannot send a 12-coin payment as no path supports such an amount. Using AMP, the payment can be split into two parts: a 10-coin transfer via Bob → Charlie and a 2-coin transfer via Bob → Elvis → Diana → Charlie. The main advantage of AMP besides increasing payment success ratio[10] is the atomicity of payments: Charlie can only claim the payment when all parts have arrived. On the other hand, AMP dramatically increases the number of HTLCs in the network since each payment part generates a new HTLC. The increase in the number of HTLCs also incurs a proportional increase in routing fees for the user, as each independent HTLC means one extra fee to be paid. Thus, AMP is an efficient alternative to the standard protocol in cases where the payment is too large to fit a single path, but it is not likely to become a new standard in the future.

3) Flare: Flare is an early alternative routing protocol for payments in the Lightning Network [55]. The main innovation of Flare is that it eliminates the need for storing the complete topology of payment channels. The algorithm consists of two phases: (i) a proactive construction of routing tables that store a partial view of the connections in the network; and (ii) a reactive probing of channels based on a routing request, which

[10] We consider the terms "payment goodput" and "payment success ratio" as synonyms for the amount of payments that are delivered successfully over the amount of sent payments.

collects dynamic information such as channel balances and forwarding fees to rank candidate paths. Then, the sender chooses the best-ranked path and adopts source routing to send the payment. The routing tables are not built like in traditional hop-by-hop routing but instead contain a subgraph of the topology in which all possible paths are known. They also store the path to a few beacon nodes which are not in the node's neighborhood, so that nodes can reach other nodes which might be far away in the topology. When a payment must occur, the sender and receiver exchange routing tables and look for common nodes to build possible paths. Thanks to the beacon nodes, this procedure ensures paths can be found with a high probability even if the sender and receiver do not share nodes in their neighborhoods.

The analysis of Flare focuses largely on the amount of data spent to maintain routing tables and on the minimum amount of beacons needed to reach all nodes in the network. The simulations on a 2,000-node Watts-Strogatz graph show that 5 random beacons are needed to ensure nodes can send payments to any other node in the network, which yields tables of around 150 nodes. In a 100,000-node network, the minimum number of beacons increases to 6 with tables of 600 nodes, indicating that the solution is scalable. However, it is difficult to know whether this perceived scalability remains true when analyzing table sizes in bytes, as Flare tables also store the paths to each node. Other results in the paper indicate that having a partial view of the network increases the size of the shortest path by up to 13 hops as viewed by the sender. Such an increase could be prohibitive in real use cases, as each extra hop would incur an extra routing fee. The protocol, despite being proposed by Lightning developers, has never been fully implemented in the Lightning Network.

4) Spider: Spider is a payment routing protocol that splits transactions into parts for sending over multiple paths, similar to AMP [90], [217]. The main novelty of Spider in comparison with other protocols is that it introduces payment parts that are bounded by a fixed maximum-transaction-unit (MTU) value. Besides increasing the probability of payment delivery, bounded payment parts allow payments to be processed like packets on the Internet. Routers in Spider perform congestion control through payment queues that are only served when the channel has enough funds. Otherwise, the router holds payments in the queue and waits for payments in the opposite direction. Routers also notify payment senders in case some part remains in the queue for too long, so that senders can decide whether to abandon it or try other paths. The congestion control mechanism ensures channels remain balanced, improving channel longevity at the expense of payment latency.

The evaluation of Spider's prototype shows that the MTU approach significantly improves payment delivery, especially in scenarios with large transactions ($>$164 euros) and low-capacity channels ($<$1,000 euros). In comparison with protocols like single-path LN [54], SilentWhispers [218], Flare [55], and SpeedyMurmurs [68], Spider improves payment success ratio by up to 1.8× in a network of 106 nodes snowball-sampled from the Lightning Network. With low channel capacities, it completes up to 3× more payments than single-path LN. Spider also increases the longevity of payment channels by up to 4× in comparison with the aforementioned protocols, demonstrating the main strength of the congestion control mechanism. On the other hand, the average payment latency of Spider is up to 2.5× higher than landmark protocols like Flare, SilentWhispers, and SpeedyMurmurs for large transactions. Similar to AMP, Spider multiplies the number of HTLCs in a payment proportionally to the MTU value, which introduces management overhead and increases the feed to be paid. Spider also ignores routing fees when selecting routes and does not guarantee payment atomicity, which can make it difficult to adopt it in practice.

5) Flash: The Flash protocol is an adaptive routing proposal that differentiates the routing methods of high-value (elephant) payments from low-value (mice) payments [80]. The authors argue that elephant payments are more likely to fail due to a lack of channel funds, so the protocol should adopt an optimized method to select paths. Thus, Flash develops a modified version of the Edmonds-Karp max-flow algorithm [219] which probes paths before sending payments. On the other hand, the protocol adopts a simple trial-and-error approach for mice payments, as they are likely to succeed even if path selection is sub-optimal. As mice payments are far more common in practice, this approach saves resources that would otherwise be spent on complex max-flow path selection and payment splits.

The authors of Flash compare a prototype of the protocol with Spider [90], SpeedyMurmurs [68], and shortest path selection with the number of hops as cost. Simulations in a 1,870-node snapshot of the Ripple network [220] and in a 2,511-node snapshot of the Lightning Network [54] show that the performance of Flash is equivalent to Spider's concerning payment success ratio and that both protocols outperform SpeedyMurmurs by roughly 10%. In a Watts-Strogatz network with 100 nodes, the success ratio of Spider is 8.8% higher than Flash's on average, which the authors of Flash attribute to Spider's congestion control. However, Flash outperforms Spider by 2.3× and SpeedyMurmurs by 5× concerning payment success volume (i.e., the total amount that reaches the target) in the Ripple network, demonstrating the max-flow approach of elephant payments is indeed effective. Flash's average processing latency is roughly 19% better than Spider's, which is a clear improvement but still means 4× the delay of simple shortest-path approaches. We conclude, thus, that Flash provides a significant improvement over Spider concerning success volume, but suffers from similar problems concerning payment latency. Moreover, Flash's max-flow approach without congestion control facilitates channel exhaustion since the algorithm sends as much value as possible through paths.

6) Pickhardt Payments: Pickhardt et al. propose a routing protocol that estimates payment channel balances to minimize uncertainties [116]. The initial estimate predicts that half of the channel capacity is allocated to each party. With this estimate, the protocol selects paths whose channels have the highest balance and tries to send payments through them. If the payment fails on any channel, the algorithm knows that the balance of that channel is less than the payment value and updates its topology to reflect this new information. The same occurs when the payment is successful, as it is certain that the channel balance decreased from the payment amount. The protocol can collect information about the state of the network through such updates while issuing payments, which serves as flow control for future payments.

Another novelty of Pickhardt payments is the introduction of a mixed-metric approach that considers a combination of routing fees and delivery probabilities as the cost of a channel. The user can then tune a parameter to decide whether to put a larger weight into delivery probabilities, maximizing the chance of successful payment, or into routing fees, minimizing financial costs. The algorithm then solves a min-cost flow problem to find paths that minimize the overall combined weight. Although we cannot quantitatively estimate the efficiency of the protocol as it has yet to be compared with the other protocols, the flexibility of this approach seems promising in use cases where payments cannot fail. The main criticism of Pickhardt payments is its high latency due to the complexity of topology updates and path selection mechanisms [221].

7) CoinExpress: CoinExpress introduces a dynamic routing protocol to minimize channel balance uncertainty [66]. In the protocol, the user probes the network to reserve channel balances before sending the payment. CoinExpress adopts a variation of the well-known Ford-Fulkerson algorithm to find the paths that maximize the flow sent in the network [222]. In each step, the algorithm iteratively probes known channels using Breadth-First-Search (BFS) and updates the candidate path set whenever an augmenting path (i.e., a path that increases the amount of flow sent) is found. However, in contrast with other max-flow approaches like Spider and Pickhardt payments, CoinExpress considers a timeliness side constraint to ensure the payment latency is bounded. Paths that would violate a predefined deadline are discarded. The protocol also defines a locking mechanism to avoid race conditions and deadlocks when multiple senders attempt to reserve channel balances simultaneously. The path-finding algorithm stops when the flow amount is equal to the payment value or if no more augmenting paths can be found. In the first case, the protocol proceeds to send the payment; otherwise, it cancels it.

The probe-then-send approach ensures payment atomicity and increases payment success ratio by up to 4× compared to (single-path) shortest-path and widest-path approaches. However, the timeliness constraint reduces the performance in about 20% compared to a push-relabel max-flow approach [62], which the authors argue is the price to pay for only selecting paths that meet a desired deadline. The main advantage of CoinExpress is the payment latency it provides, which for small payments is approximately 5% more than single-path algorithms despite it being a multipath max-flow algorithm. However, without a detailed analysis, it is unclear whether this effect could be caused by small payments not requiring more than one iteration of the max-flow algorithm. It is also unclear how CoinExpress fares against other max-flow approaches for large payments. As with other probing protocols, CoinExpress may eventually flood the network with probing messages in high-load scenarios and sacrifice channel privacy. The proposal does not consider transaction queues, congestion control, and multi-metric path discovery.

8) HushRelay: Another approach inspired by classical max-flow algorithms is HushRelay [95]. In HushRelay, nodes run an adapted version of the Push-Relabel algorithm in which each node tries to push excess flow along its neighbors via probing messages [223]. The nodes can either accept the flow or reject it via acknowledgments. The protocol also preserves payment privacy[11] by replacing the sender and receiver with dummy nodes that do not identify them as users. The authors compare the protocol against SpeedyMurmurs [68] and show HushRelay achieves up to twice the payment success ratio with half the latency in 25,000-node scale-free graphs. However, there is no clear discussion on why the algorithm is so efficient nor comparisons with other routing protocols. A possible explanation is that, like CoinExpress, we expect HushRelay's multi-path probing approach to improve payment success ratio in comparison with single-path approaches but it can also overload the network with messages. The authors do not present results regarding communication overhead, nor multi-metric approaches or congestion control techniques.

9) SilentWhispers: Malavolta et al. propose SilentWhispers, a routing protocol in PCNs that contains privacy guarantees and seeks to reduce the overhead generated by storing the complete network topology in each user [218]. SilentWhispers embeds the payment network into a geometric space so that pathfinding can be guided with spatial information. However, only a small number of nodes, the landmarks, know the complete embedding. The other nodes only compute their distance and routes to all landmarks. Then, whenever a user sends a payment, it sends the payment to a landmark, which will relay it to the destination. A querying mechanism helps in this private route discovery, where nodes can inquire about potential next hops without revealing the entire route or payment details. The main advantage of this approach is that users need only to maintain the paths to the landmarks. The hierarchical architecture helps to preserve user and path privacy and is better adapted to scenarios with resource-limited devices that cannot calculate routes and maintain a complete network topology. The main drawbacks are the centralization caused by the landmarks, which can be malicious, and the frequent

[11] A payment is considered private if the sender, receiver, and payment value are confidential to the involved parties. We elaborate more on privacy definitions for PCNs in Section VI-D.

channel reuse [68]. Using landmarks can also lead to longer paths than necessary. This type of routing, despite presenting a solution with more significant potential for the inclusion of lightweight devices, still needs practical observations and large-scale tests.

10) Rapido: Rapido is a routing protocol that splits the routing procedure into phases and allocates them to dedicated modules [96]. In the first phase, the network is split into regions that elect a beacon node each. The beacon node is responsible for delivering payments inside its region or relaying them to other regions. Rapido splits routing into two different parts. The first one, called the proactive part, gathers static path information, such as paths to all elected beacons. The second part, called the reactive part, focuses on dynamic network information, such as how the capacity is split among participants in the path. Then, whenever a user needs to send a payment, it first leverages the proactive routing module to compute paths to beacons. The reactive module probes the paths for dynamic information that will be fed to the value distribution module, which calculates the optimal payment split to allocate to each path. The value distribution module considers the balance of payment channels in the path, which, similar to Pickhardt Payments, indirectly provides congestion control. The work also proposes a new structure named D-HTLC (Distributed HTLC) that provides privacy of the payment value. Because nodes compute routes to beacons, Rapido's routing algorithm can be considered a form of multipath landmark routing. The authors of Rapido compare the proposal with the LN default routing protocol and show that it improves the payment success ratio by up to 3x at the expense of up to 20% more latency in pathfinding. The proposal is yet to be compared with other multipath protocols.

11) SpeedyMurmurs: SpeedyMurmurs mitigates the drawbacks of SilentWhispers by adopting an embedding-based path discovery algorithm [68]. With path embedding, each node is assigned a coordinate that can be used to calculate the distance between the node and a target destination. Thus, senders can calculate distances and find "shortcuts" in the network that skip landmarks without maintaining the complete topology. Landmark influence is also reduced to building embeddings instead of routing every transaction. SpeedyMurmurs keeps the privacy guarantees of SilentWhispers while improving the payment success ratio by up to 40% and latency by up to 5x. It also provides an efficient network stabilization mechanism which keeps embedding up-to-date with a small overhead. The main shortcomings of the protocol are the susceptibility to congestion in the network, which heavily impacts pathfinding latency, and the lack of payment atomicity. Besides, like SilentWhispers, SpeedyMurmurs ignores routing fees, making it difficult to assess whether the protocol might be adopted in practice.

12) CheaPay: CheaPay proposes a routing protocol based on the Bellman-Ford algorithm that aims to minimize routing fees to be paid along a path [81], [225]. Apart from fees, their approach also considers two critical aspects of payment routing to find paths: payment feasibility, i.e., whether all channels along a path will be able to forward the payment, and payment timeliness, i.e., whether the payment can be fulfilled before the HTLCs in each channel expire. The algorithm enforces these constraints by iteratively checking candidate paths and discarding them whenever either condition is not met. The main difference of this work in comparison with others is that it relies on routing tables instead of source routing. PCN nodes exchange link-cost information (in this case, routing fees) using a distributed Bellman-Ford algorithm like in classical networks to build and maintain the routing tables. However, the work does not support multiple paths or metrics nor preserves transaction privacy. It also lacks quantitative comparisons with other payment routing protocols.

13) RobustPay: RobustPay is a routing protocol that aims to increase payment success probability by sending payments through redundant paths [224]. This way, payments could be completed even if nodes along one of the paths become unresponsive. The protocol invokes CheaPay to find minimal-fee paths that hold the properties of payment feasibility and timeliness [81]. It then uses a variant of Suurballe's algorithm to find two disjoint minimal paths and sends the payment through both paths [226]. The work, however, does not truly support multipath because the HTLCs in one of the paths are canceled whenever the payment has been claimed in the other path. Hence, the extra paths provide redundancy. Also, the authors modify the default HTLC implementation to ensure paths can be unlocked quickly and safely. Like in CheaPay, paths are found through routing tables built via the distributed Bellman-Ford algorithm, which reduces pathfinding delays. RobustPay maintains a similar payment success ratio when compared to CheaPay despite providing the extra robustness. However, both RobustPay and CheaPay lack comparison with other protocols. RobustPay may also cause deadlocks in high-load scenarios due to the extra path locks.

14) AODV-based Routing: Hoenisch and Weber propose adapting the Ad-hoc On-demand Distance Vector (AODV)-based routing protocol for payment channel networks [69]. The key idea of the proposal is to acknowledge that a payment channel network is inherently dynamic because of constant changes in channel balances and routing fees and because of node churn. Hence, routes should be discovered on demand, and nodes should not be assumed to maintain a synchronized topology proactively. As in traditional AODV-based routing, whenever a sender wants to send a payment, it floods the network with route request messages that will be forwarded hop-by-hop until they reach the desired destination. Then, the receiver communicates the complete route via a route response message to the sender, which can proceed to issue the payment normally. The main characteristic of this approach is that it trades the message overhead needed to maintain a synchronized topology for an on-demand message overhead whenever a route is needed. Thus, it is most fit for dynamic environments with many light nodes, in which synchronizing nodes is infeasible. The on-demand approach also mitigates

Table V
COMPARISON BETWEEN ROUTING PROTOCOLS FOR PAYMENT CHANNEL NETWORKS.

Reference	Routing type	Global view[1]	Congestion control	Support for multipath	Support for multimetric	Payment privacy[2]	Payment atomicity	Other observations
Lightning Network (default) [54]	Source routing	✓	✗	✗	✗	✓	✓	Based on Dijkstra's shortest path algorithm
Lightning Network (AMP) [67]	Source routing	✓	✗	✓	✗	✓	✓	Ensures atomicity for multipath payments
Spider [90]	Source routing	✓	✓	✓	✗	✗	✗	Introduces transacion queues for congestion control
Flash [80]	Source routing	✓	✗	✓	✗	✗	✗	Differentiates routing methods for elephant and mice payments
Pickhardt Payments [116]	Source routing	✓	✓	✓	✓	✓	✗	Adopts payment probability as a routing metric
CoinExpress [66]	Source routing	✓	✗	✓	✗	✗	✓	Inspired by the distributed Ford-Fulkerson algorithm
HushRelay [95]	Source routing	✓	✗	✗	✗	✗	✗	Inspired by the distributed push-relabel algorithm
Flare [55]	Source routing	✗	✗	✗	✗	✗	✓	Finds paths based on local subgraphs and node beacons
SilentWhispers [218]	Landmark routing	✗	✗	✓	✗	✓	✗	Relies on landmarks to calculate routes and forward payments
Rapido [96]	Landmark routing	✗	✓	✓	✗	✓	✗	Relies on landmarks and path probing
SpeedyMurmurs [68]	Path-embedding routing	✗	✗	✓	✗	✓	✗	Uses path-embedding based on coordinates and distances
CheaPay [81]	Hop-by-hop (proactive)	✗	✗	✗	✗	✗	✓	Builds routing tables using the Bellman-Ford algorithm
RobustPay [224]	Hop-by-hop (proactive)	✗	✗	✗	✗	✗	✓	Improves CheaPay with redundant paths to tolerate node faults
AODV-based Routing [69]	Hop-by-hop (reactive)	✗	✗	✗	✗	✗	✓	Finds routes on-demand via route request and replies
Ant Routing [97]	Hop-by-hop (reactive)	✗	✗	✗	✗	✗	✓	Finds routes using pheromone seeds, like ant colonies.

[1] If marked "✓", global view means the sender must know the complete topology of the network to find routes and issue payments.
[2] Payment privacy is defined here as the secrecy of the sender/receiver pair, of the payment value, and of the payment path. We mark "✓" for protocols in which all of the aforementioned information is kept private to nodes that are involved in the payment. We elaborate more on privacy definitions in Section VI-D.

the impact of zombie channels[12], since unresponsive nodes will never respond to the route requests. The proposal has not been evaluated with respect to latency and payment success ratio, nor compared with other routing protocols.

15) Ant Routing: Grunspan et al. propose a routing protocol that finds paths in the network inspired by colonies of ants [97]. The idea is that both sender and receiver generate the same random pheromone seed and broadcast it to random neighbors in the network. Intermediary nodes gossip the seed until some node receives the seed from different neighbors, causing a match like ants touching antennas. When the seeds match, the intermediary node sends a confirmation seed to the payment sender that a path exists and that it can proceed with the payment. The sender then issues the payment to the neighbor from where the confirmation came, and the payment gets routed in the network via the path found. Because the pheromone seed is broadcast, multiple matches corresponding to multiple possible paths can occur. However, like with RobustPay, the extra paths are not used for delivering multiple payment parts but rather to improve payment privacy and allow the sender to choose between paths with different fees. The protocol does not rely on routing tables but creates an Adelson-Velsky and Landis (AVL) tree that associates each payment with the next hop in the path [97]. It is not clear whether such an approach is scalable or efficient, as it lacks formal evaluation and comparison with other protocols.

16) Summary of Routing Protocols and Discussion: Table V summarizes the main points of the proposals presented for the routing challenges in PCN and compares them with the implementation of the Lightning Network. In the table, the topology overview line indicates whether the user must know the complete network topology to issue a payment. Proposals that adopt source routing, such as Lightning Network, Spider, Flash, CoinExpress, and Pickhardt et al., assume senders know the complete network topology to compute paths. The main advantages of this approach are guaranteeing the predictability of routing fees and allowing payments to be onion-routed. However, senders must be synchronized with the network and spend energy on pathfinding. Thus, source-routing should be adopted when it is expected that nodes

[12] Zombie channels are channels in which one node has become unresponsive, but it is still unclear whether they will come back. This is a significant issue in payment routing since the channel is still officially available but cannot route payments.

in the network are synchronized and have enough computing power. In CheaPay and RobustPay, routing tables indicate the optimal paths and payments are routed hop-by-hop, saving resources. This approach transfers the computing of tables to capable nodes, enabling light nodes to participate at the expense of unpredictable fees and weaker privacy. AODV-based routing also routes hop-by-hop but adopts a reactive mechanism for discovering routes instead of maintaining routing tables, which slows down the routing process in static networks. Therefore, AODV should only be used when tables cannot be efficiently maintained. SilentWhispers and SpeedyMurmurs provide a hybrid solution that requires the sender to compute a partial path to one of the landmarks which are responsible for forwarding the payment. This approach also reduces the sender's effort but could lead to common routing attacks, such as sinkholes, if the landmarks are adversaries.

We also observe that several protocols adopt multi-path approaches for sending payments, seeking to increase the payment success rate through payment splitting. The key idea is that small payments fit into low-capacity channels, increasing the probability of delivery. However, surprisingly enough, not all protocols consider fixed routing fees a cost metric despite them being the default metric for most payment channel networks [66], [68], [90], [95], [218]. We argue that such protocols are difficult to adopt by default in practice, as splitting would incur extra fees that would make them unattractive for users. Multi-path approaches should consequently be preferable only when payments do not fit into a single path. Furthermore, multi-path payments raise concerns about atomicity, as some payment parts may fail while others reach the target. The atomicity of payments is guaranteed by all single-path protocols and the AMP [67] and CoinExpress [66] protocols, which cancel all payments if a path cannot deliver a payment part. In particular, AMP can be used as a generic atomicity service for multipath non-atomic routing protocols [94].

Some protocols present unique characteristics which separate them from the others. Pickhardt et al. [116] present the only protocol that permits multi-metric optimization, considering fees and delivery probabilities. Multi-metric optimization may be useful for applications with specific requirements, such as a minimal payment delivery probability regardless of the amount of fees to be paid. SilentWhispers, SpeedyMurmurs, Rapido, and HushRelay are the only protocols that guarantee payment privacy, i.e., the confidentiality of the sender/receiver pair, payment value, and payment path. Protocols that probe candidate paths with payment values, such as CoinExpress, or that adopt distributed max-flow algorithms, such as HushRelay, disclose payment information to nodes that may not be involved in the chosen path, therefore compromising payment privacy. We observe that the protocols that ensure payment privacy are those that adopt source routing without probing.

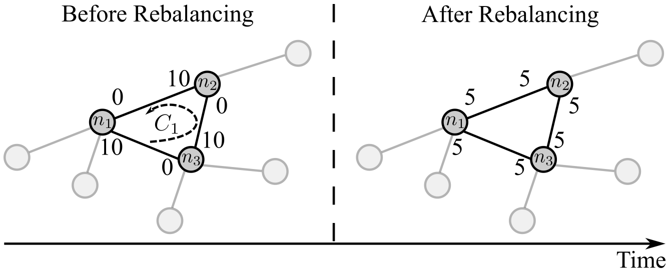

Figure 15. Active rebalancing of channels through circular self-payments in a payment channel network.

Despite being the most extensively studied topic of PCNs, payment routing still presents open challenges and opportunities for research. For example, payment latency in the Lightning Network is still in the order of seconds, which needs to be improved for some applications. Current routing protocols in payment channel networks do not cover this since they aim to minimize routing fees or maximize the probability of payment completion. Also, no works analyze in depth the efficiency loss caused by routing payments individually as opposed to a global payment scheduler that routes payments optimally. This comparison would yield the "price of anarchy" of routing protocols in payment channel networks, a concept that measures how the efficiency of a system degrades due to the selfish behavior of its agents [227]. Finally, most routing protocols assume nodes share a globally synchronized topology and are always online, which excludes mobile nodes with low resources and intermittent connectivity. Recent efforts on the Lightning Network attempt to improve this scenario, but need more formalization and extensive testing [228].

B. *Channel Rebalancing*

The channel liquidity distribution directly influences its capacity to forward payments and overall payment goodput. Thus, keeping channels balanced is a crucial concern in payment channel networks. Routing protocols that adopt congestion control mechanisms, namely Spider [90], Pickhardt payments [116], and Rapido [96], indirectly contribute to reducing channel imbalance. However, as most applications have a well-defined tendency for payment flows, e.g., from buyers to sellers, more than congestion control is needed to maintain balances in the long run. One of the most significant challenges in payment channel networks is how to propose efficient rebalancing mechanisms that preserve the duration of the channels and maximize their usefulness. In this section, we describe and compare several proposals found in the literature.

1) Passive Rebalancing: The simplest and most used form of rebalancing is to encourage payments by changing the forwarding fees being charged. In this method, also called *passive* rebalancing in the literature, intermediaries increase their fees in a given channel whenever they detect that a channel is imbalanced [54], [82]. The idea is that, as the fees increase, fewer users will choose the channel in that direction, and the channel will slowly balance itself as payments arrive in the opposite direction.

In general, intermediaries manually adjust fees according to convenience, but some tools automate the process [229], [230]. Differently, Di Stasi et al. [65] propose a new fee function that discourages payments that promote further channel imbalance. Similar to the Lightning Network, the fee function has a fixed fee, accounting for operational costs, and a proportional fee that depends on the size of the payment. The proposed proportional fee has two different slopes: s_{low} and s_{high}. The slow slope, denoted by s_{low}, accounts for the part of the payment that decreases the channel imbalance, while the steeper slope, s_{high}, is applied to the rest of the payment. Thus, payments that further increase channel imbalance will pay higher fees due to the steep slope. Likewise, payments that reduce channel imbalance pay low fees due to the slow slope, which helps to maintain channel balance.

The main advantage of passive rebalancing lies in recovering the channel usability while avoiding both on-chain and off-chain payment fees. Thus, passive rebalancing is the cheapest rebalancing method for the end user. Nevertheless, passive rebalancing relies on payment flow demands uncertain on the channel's counterparty side. A node may lower its fees and see little effect on its balance if no demands are in the opposite direction. Furthermore, the feasibility of this approach is restricted to central and richly connected nodes that act as intermediaries for other payment flows. This approach works best on the Lightning Network, as its default algorithm considers fees the primary metric for choosing paths. However, whether this strategy is efficient with algorithms considering other parameters is uncertain.

2) Active Rebalancing Via Circular Payments: In *active* rebalancing techniques, users issue payments to entities or themselves to receive outbound or inbound liquidity. The main advantage of this technique is that it does not rely on uncertain payment demands. Nevertheless, as users have to issue payments, active rebalancing introduces fees, presenting itself as a more expensive alternative than passive rebalancing. Active rebalancing methods can be further categorized into circular and non-circular payments.

In the circular payments scheme, depicted in Figure 15, users leverage the network topology to issue self-payments and rebalance the channels along the payment path. While circular payment techniques successfully avoid the blockchain to refund the channel, the amount of coins is restricted by the balances on the circular route. Circular payments can be further divided into two types: local and global protocols. Each user uses local protocols to calculate a set of rebalancing transactions that are optimal for their channels.

Imbalance measure. Pickhardt and Nowostawski introduce a network imbalance metric and a rebalancing method that minimizes local imbalance between channels of a user [94]. The idea is that the user constantly calculates the difference in balance between all the channels in which it is involved in order to keep them in balance with each other. Suppose there is a significant disparity between the two channels. In that case, the user makes a cyclical payment to itself from the channel with the highest balance to the channel with the lowest balance, changing the distribution of balances. Thus, the goal is that the user can always route payments equally in any direction. The authors show that this local heuristic improves the network's global balance if nodes collaborate with each other. Nevertheless, this collaboration is difficult to guarantee in a decentralized system where users act independently. The work also lacks significant experiments on the impact of this strategy if only a fraction of the nodes adopt the proposal.

Pickhardt and Nowostawski evaluate their strategy using a Lightning Network snapshot. The authors verify that the network imbalance, measured by a normalized Gini coefficient, quickly drops with the adoption of their solution, going from 0.5 to 0.2. They also quantify the number of rebalancing operations in the network, which varies from 10,000 to 100,000 operations. Furthermore, using the author's strategy increases the number of paths that are able to forward a single satoshi from 11.2% to 98.3% paths. Nonetheless, the evaluation is limited as it lacks a comparison with other proposals and experiments in a highly dynamic environment, where nodes can open and close channels whenever they want.

REBAL. Awaethare et al. [115] propose a local and circular rebalancing algorithm called REBAL. REBAL accounts for the channel's workload history to decide on the rebalancing amounts. To motivate their work, the authors simulate a PCN environment by using the Lightning Network topology and a real-world Ripple transaction wokload [68], [220]. They show that even with initially balanced channels, over 63% of channels become imbalanced after 200 seconds, with 80% of the channel balance being concentrated on one side of the bidirectional channel. REBAL rebalances the maximum number of channels by selecting the longest cycle the node is involved in at the start of the rebalancing algorithm. REBAL has the major advantage of keeping channel balance information private due to its local rebalancing. REBAL also allows intermediary nodes to redirect payments through an alternative path, which avoids halting payments while the rebalancing occurs. However, the authors do not get into detail on how the re-routes work, especially in an onion-based source-routing PCN such as the Lightning Network.

The authors use the Spider simulator to implement REBAL and compare it with other routing proposals for the Lightning Network. The simulation shows that a rebalancing operation takes around 2 seconds and that the transaction success rate is higher when the node rebalances its channel every 40 seconds. In that configuration, REBAL processes more than 2× the transaction volume of other routing algorithms, such as Spider, landmark routing, and waterfilling, when the transaction generation rate is small. Furthermore, the simulation shows that the transaction success rate is higher if the threshold to determine whether the channel is imbalanced

Table VI
PER-FEATURE COMPARISON OF REBALANCING ALGORITHMS IN THE LITERATURE.

Reference	Rebalancing Type	Circular Payments	Global Rebalancing	Privacy Guarantees	Transaction on the Blockchain	Other Features
Imbalance measure [94]	Active	✓	✗	✓	✗	Allows users to share balance to acheieve more efficient rebalancing
REBAL [115]	Active	✓	✗	✓	✗	Considers workload history to calculate rebalancing
REVIVE [61]	Active	✓	✓	✗	✗	Consensus to establish fault-tolerance
Hide&Seek [128]	Active	✓	✓	✓	✗	Uses Multi-Party Computation to ensure privacy
Submarine Swaps (Loop) [129]	Active	✗	✗	✓	✓	Exchanges on-chain funds for off-chain balance
Splicing [60]	Active	✗	✗	✓	✓	Requires users to close and reopen the channel
Shaduf [130]	Active	✗	✗	✓	✓	Allows fund shifting through a single on-chain transaction
Fee management	Passive	✗	✗	✓	✗	Uses fee to (des)incentivize routing
Di Stasi [65]	Passive	✗	✗	✓	✗	Proposes a new fee that (des)incentivize routing through (im)balanced channels

is set to 50% of the channel capacity. The authors, however, fail to evaluate crucial aspects of REBAL, such as success rate under asymmetrical payment demands, costs, and how REBAL affects other channels' balances.

REVIVE. Khalil and Gervais [61] propose Revive, a circular and global rebalancing algorithm that takes advantage of cycles in the network topology to rebalance channels, reducing the need to resort to the blockchain. In global rebalancing protocols, users send their desired balances to a third party that calculates a set of transactions among every user that better fits their needs. In Revive, the elected leader receives rebalancing requests from multiple users and calculates a set of transactions that must be performed. This set of transactions considers user requirements and must ensure that users do not lose funds. Thus, the proposed algorithm shifts coins between channels, respecting the rebalancing preferences provided by users and conserving the credit allocated by each user on the network. REVIVE has the major advantage of computing very efficient rebalancing operations given that it uses information from a high number of nodes. The algorithm, however, compromises users' privacy: to calculate the set of rebalancing transactions, the leader must know the balance of the channels involved. Furthermore, the authors do not evaluate REVIVE, which makes it difficult to assess the run time of the complete protocol, from leader election to the receiving of rebalancing transactions.

The choice between local and global rebalancing techniques usually presents a trade-off between privacy and efficiency. Local rebalancing focuses on local information to calculate the optimal set of transactions, which can worsen third-party channels' balances despite protecting user privacy. On the other hand, global rebalancing techniques efficiently calculate a rebalance operation that is optimal for the whole network. Nevertheless, this technique usually involves trusting a third-party or revealing private balance information.

Hide&Seek. Avarikioti et al. [128] propose a privacy-preserving global and circular rebalancing technique called Hide&Seek. It uses multi-party computation (MPC) to improve on REVIVE's solution and reach a fully private solution. In Hide&Seek, selected participants jointly compute the optimal solution of a linear programming problem. The authors model the rebalancing problem as a linear programming problem and solve it to find the maximum circulation payment flow that meets the rebalance demands from each user. Although Hide&Seek offers additional privacy guarantees, using MPC adds a significant overhead, which results in slow rebalancing operations [128], [133]. Furthermore, the authors fail to evaluate Hide&Seek, which makes it difficult to quantify the MPC's overhead and Hide&Seek's improvement in payment success rate.

3) Active Rebalancing Via Non-circular Payments: Non-circular rebalancing techniques usually involve sending coins to a user or entity and receiving the same amount as inbound or outbound liquidity. In the Lightning Network, this happens through two main alternatives: submarine swaps [129] and splicing [60].

Submarine swaps. Submarine swaps present a trustless exchange of on-chain funds for off-chain balance [129]. Thus, a user A that wishes to rebalance a channel issues a transaction in the blockchain to user B and receives the same amount back from user B on its off-chain channel. This technique is named "*submarine swap*" because the payments of Layer 2 users have to "immerse" to a lower layer, Layer 1, to receive the funds back on Layer 2, like a submarine immersing in the ocean. The primary tool to perform submarine swaps in the Lightning Network is *Loop*, which executes either *Loop In*, which converts a Bitcoin payment to a Lightning payment, or *Loop Out*, which converts Lightning payments to Bitcoin payments [129]. While submarine swaps allow users to acquire liquidity without closing the channel, they request on-chain payments that are expensive and time-consuming. Although submarine swaps became a popular solution in the LN, it still lacks quantitative analysis comparing it to other proposals.

Splicing. Splicing is a Layer-one rebalancing technique that resorts to changing the channel's capacity through a blockchain transaction [60]. In Splicing, users must close and reopen the channel with a different capacity by issuing an on-chain transaction. The splicing operation is called *Splice In* when the new channel capacity is higher than the closed one and *Splice Out* when the new channel capacity is lower. Although Splicing allows users to reopen the channel with a new capacity, the channel must be closed and reopened through a blockchain transaction, which is expensive and time-consuming. Similarly to submarine swaps, Splicing lacks quantitative analysis.

Shaduf. Ge et al. [130] introduce Shaduf, a non-circular active rebalancing protocol that leverages a one-time binding on-chain transaction to allow many-times channel rebalancing off-chain. Shaduf relies on a Layer 1 transaction, avoiding cycles to move funds. Instead, users can move funds through adjacent channels as often as they want with only one on-chain transaction. Similarly to Loop, however, Shaduf is more expensive when compared to off-chain rebalancing, as it introduces on-chain fees. Nonetheless, it allows users to move funds through channels without requiring a specific topology to achieve the rebalancing.

Ge et al. implement Shaduf and compare it to REVIVE [61] and the standard Lightning Network operation. While REVIVE enhances the transaction success rate by about 7% the Lightning Network standard operation, Shaduf achieves an enhancement from 10% to 22% as channel capacities get larger. The authors also verify Shaduf's success ratio when the payments are highly skewed in the network. In their simulation, some nodes receive the roles of merchants, who receive more payments than they issue, while other nodes are customers, who make more payments than they receive. Under skewed payments, Shaduf increases the payment success rate of the default LN operation in 15% while REVIVE enhances 7.5%. The simulation also shows that Shaduf costs at most 8.13 USD while closing and reopening the channel costs around 3.07 USD. Nevertheless, Shaduf allows unlimited channel rebalancing while closing and reopening the channel must be repeated every time the channel is depleted.

4) Summary of Rebalancing Mechanisms and Discussion: Table VI summarizes the categories and proposals on rebalancing algorithms. The Lightning Network features a combination of active and passive methods, as well as channel reestablishment through blockchain transactions. The network's central routers often regulate their balances passively through fee management, as they constantly receive payments from all directions. Less central users adopt the circular method if they have channels with sufficient capacity or recreate the channel by reserving more coins through a direct transaction on the blockchain. There is still no solution that guarantees the balance of the network systematically.

Channel balancing is one of the most important research topics in payment channel networks. Most researchers propose an active rebalancing mechanism, as passive rebalancing depends on uncertain demands and is not widely available to every node in the network. In particular, a great part of current approaches proposes an active rebalancing through circular routes [61], [94], [115], [128] due to its simplicity and low cost. Nevertheless, there are still open challenges to reach an efficient active rebalancing solution that maintains user privacy. While global rebalancing mechanisms are usually effective, they rely on users disclosing private information. On the other hand, local rebalancing mechanisms keep channel balance private but are usually ineffective. Future designs on global active rebalancing need to provide stronger privacy guarantees while reducing this efficiency loss. Using modern cryptographic techniques, such as MPC in Hide&Seek, is promising but requires further evaluation to assess its overhead and feasibility. Alternatively, local rebalancing proposals need studies on its convergence, how it impacts other nodes' balance, and how far the local approach is from the more efficient global rebalancing. There are still open challenges for defining rebalancing policies to define when to rebalance [231]. Some approaches [115], [129] define a channel balance threshold to define whether a channel is balanced. This is sub-optimal as it ignores dynamic payment demands in the network. Similarly, defining how many coins to allocate to a depleted channel and from which channel to allocate is still an open challenge. Some proposals split the channel capacity equally between the two parties when rebalancing [61], [94], which is also sub-optimal when payment demands are unequal in both directions of the channel.

C. Network Design

As the Lightning Network rapidly grows in terms of participants, the research community delves into the strategies of where to create a channel in the topology and how much funds to allocate to it [98], [99], [117], [118], [134]. These choices are crucial to new nodes that want to have a profitable channel by maximizing the income from routing fees. Furthermore, this challenge is essential to some Lightning Network implementations, which automate channel opening through autopilots [83], [135]. The literature on network design in the Lightning Network is rich. While some studies analyze existing node attachment strategies and their impact on the network [99], [117], [118], others propose novel attachment strategies to nodes [98], [100].

1) Network Design Analysis: Some papers study the topology of payment channel networks using game theory [99], [118]. The idea is to analyze how the network was initially formed, how it is evolving, and compare the efficiency loss of this independently formed network with a centralized alternative. Avarikioti et al. [99] study the network's topology when players act selfishly by weighting the benefits of opening a payment channel, determined by the potential transaction forwarding fees and the routing price, and the costs of opening a channel. To analyze this, the authors model the expected fee reward of a channel using betweenness centrality, which is based on the shortest paths passing through the node. Furthermore, the authors use closeness centrality to predict the number of fees paid when issuing transactions to other players in the network. The paper then demonstrates that the problem of finding the best response, i.e., where to add the node in the network knowing every player's strategy, is NP-hard. Also using game theory, Wan et al. [134] study the existence of pure Nash Equilibrium in the price-setting game between two payment hubs. They compare the optimal revenue hubs can achieve when cooperating instead of competing. They find that the competitive nature of PCNs will result in much lower fees with the increase in network capacity.

In another direction, Lange et al. [117] analyze possible node attachment strategies and their long-term impact on the network. The authors use common graph-theory strategies, such as random, k-median, or highest-degree attachment, and evaluate payment success rates, fee amounts, income inequality, and the network diameter. The results show that, from a selfish point of view, centralized attachments provide better short-term results, such as a higher transaction success rate. On the other hand, decentralized strategies provide better long-term results to the network overall. For example, a participant using the k-median approach to create channels can concentrate almost 3% of all payments routed by opening 15 channels. Nevertheless, as the number of participants using k-median grows, the network diameter and inequality soar, becoming twice the value of k-center and random approaches.

2) Attachment and Balance Planning: The question of how many coins to allocate to a newly created channel is fundamental in payment channel networks. If a user creates a channel with low funds, it must constantly be refunded through the blockchain. On the other hand, allocating too many funds to a channel is inefficient, as other applications might use these coins. To solve this problem, Li et al. present PnP, a secure balance planning for PCNs [98]. PnP uses estimated payment demands to calculate a chance-constrained optimization problem that minimizes channel deposits efficiently. Furthermore, in PnP, nodes use a cryptographic sortition protocol to randomly select a committee that runs the proposed algorithm, removing the need for a centralized or trusted third party.

Li et al. implement PnP as a service to interact with LND, an implementation of the Lightning Network. The authors find the effect of demand estimate error on the number of satisfied payment demands of PnP is only around 3%. Furthermore, the authors also verify that PnP satisfies around 95% of payment demands and creates on average 82 channels with only 4 being exhausted. The authors refrain from comparing PnP with other work in the literature as PnP was the first balance planning protocol for PCNs.

In a different direction, Esroy et al. [100] study how to make payment channels profitable. More specifically, the authors analyze how users can maximize profit by evaluating where to create a channel and how much to charge for routing fees. The authors formulate the maximum reward improvement problem (MRI): finding k channels incident to a given node n that maximize the expected reward of n. This problem is based on the maximum betweenness improvement problem (MBI) [70] with the difference that MRI also tries to choose an optimal fee policy. Then, the paper shows that the MRI problem is NP-hard and proposes a greedy algorithm that tries multiple channels, calculates the expected reward, and connects to those that give the maximum reward. To calculate the expected rewards, the authors use the betweenness centrality graph metric and a function that defines the fees to be charged.

The authors also evaluate their proposal on a snapshot of the Lightning Network. The results show that their proposed method of fee optimization increases by 2× the reward in comparison with centrality-based attachment strategies. Although greedy attachment algorithms such as the one proposed by Ersoy et al. present a high reward to the user, the work of Lange et al. [117] also show that their running time is significantly higher than other attachment strategies. While strategies such as connecting to the highest degree nodes, k-Median, and k-Center take only a few seconds to suggest channels, greedy strategies such as MBI take over 24,000 seconds (around 6 hours) to suggest 10 channels. As the network is highly dynamic, the greedy strategy might be computing channels in an outdated version of the network topology.

3) Autopilots: Multiple Lightning Network implementations offer *autopilots* that automatically open channels with multiple users [83], [135]. The autopilot of LND [135], an implementation of the Lightning Network, allows users to set parameters such as the maximum number of channels, the minimum and maximum size of the channel, and the number of funds to be allocated, and opens channels automatically for users. The autopilot uses the Barabasi-Albert model [232] of preferential attachment, randomly selecting where to add a node with a probability distribution based on the existing node's degree. In that case, the autopilot favors connections to already highly central nodes. Although this model is the standard strategy adopted by the autopilot, the LND implementation provides an interface that supports the implementation of alternative preferential attachment models.

Pickhardt [83] introduces *lib_autopilot*, a Python library that uses statistics to recommend channel openings. The implementation allows users to select preferential attachment strategies and offers four main heuristics: random, central, decreased diameter, and richness. The random heuristic follows the Erdős–Rényi model [233] and chooses the prospect channel partners from a uniform distribution. In the central strategy, the probability distribution of prospect partners is weighted by their betweenness centrality metric, which favors more central nodes. Alternatively, the decreased diameter heuristic prefers nodes with lower connectivity, and the richness heuristic draws nodes from a uniform distribution of the richest nodes, i.e., nodes with the highest capacities, in the network. The proposed autopilot also estimates the number of funds the user should allocate to the channels based on the channel capacities of the new partners. Both LND's autopilot and lib-autopilot still lack quantitative comparisons between each other and with other proposals.

4) Summary of Network Design and Discussion: Network design works mainly focus on modeling the network's current state and proposing methods for optimally opening channels. Choosing where to place new payment channels is particularly important in PCNs as it creates a trade-off between decentralization and efficiency [117]. Current implementations of the Lightning Network offer autopilots that automatically create channels without considering centralization. While attaching new channels to nodes with large capacity and numerous connections produces short paths and low payment fees, it also contributes to centralization in the network. The centralization facilitates topological attacks and leads to other long-term security vulnerabilities [79].

Several studies analyze the Lightning Network topology and attachment strategies [99], [117], [134] but efficient autopilot solutions remain largely unexplored in academia. Current autopilots [83], [135] use inefficient heuristics such as connecting to nodes with the highest betweenness centrality or degree. These heuristics, although fast, usually disregard potential financial gains for the user. Future autopilot designs must also acknowledge the trade-off between efficiency, meaning what is best for the user, and decentralization, meaning what is best for the network [117]. Possible solutions involve a hybrid approach that combines short-term financial gain with long-term network stability.

Furthermore, a significant part of current network design proposals assumes a uniform payment distribution among nodes in the network. This assumption is a result of PCNs keeping payment information private, which leads researchers to model sender-receiver pairs following a familiar distribution. Despite its popularity, this assumption might not accurately reflect the traffic in the network. Future designs should account for existing literature on PCN traffic analysis and diverse payment distribution to create more accurate attachment strategies [87], [91].

D. Challenges in Security and Privacy

Providing user privacy is arguably one of the main features of blockchain systems, especially for public cryptocurrencies such as Bitcoin and Ethereum. Hence, the off-chain mechanisms involved in payment channel networks must be carefully designed to avoid leaking information that may compromise user privacy. In this section, we discuss and classify the privacy challenges of PCNs: node anonymity, link privacy, and payment privacy.

1) Node Anonymity: As in blockchain systems, users in a payment channel network should not be personally identified. Otherwise, they might suffer targeted attacks and be prosecuted by governments and financial institutions. To preserve privacy, PCNs rely on the underlying blockchain and adopt the user's public key as the node identifier in the PCN [234]. Using public keys provides a level of anonymity that is as good as the anonymity on the underlying blockchain. The user is responsible for not linking personal data to the public key in servers of untrusted third parties, e.g., by managing keys via custodial wallets that adopt know-your-customer (KYC) measures. Note, however, that nodes in PCNs often reveal their identities to provide some level of reputation-based trust if the node is owned by a well-known blockchain company such as Bitfinex [235] or if the node is a known retailer.

2) Link Privacy: Link privacy refers to preserving payment channel information other than the channel nodes and the channel capacity, which are published in the blockchain. Kappos et al. [112] identify two privacy requirements for payment channels:

- **Channel privacy**: Two users should, if they wish, be able to hide the existence of the channel they share and preserve any information about the channel from third-party access.
- **Balance privacy**: Although total channel capacities are public, the channel balances, i.e., the distribution of resources in the channel, must remain private. Otherwise, payments could be tracked by monitoring channel balances. Publishing balances would also create a scalability problem because notifying balance updates to all channels in a highly dynamic environment would flood the network with messages.

Link privacy is usually preserved by global specifications that apply systematically to all nodes in a payment channel network. For example, the Lightning Network provides channel privacy by enforcing the advertisement of public channels on the network using the gossip protocol [234]. If a channel is not announced, other users are unaware of its existence. Nodes that only participate in private channels are also considered private since their node identifier is never announced. However, channel announcement messages in the Lightning Network do not provide space for advertising channel balances, and there is no other message type for this purpose. This means users cannot systematically reveal balances, although those can still be inferred in some situations via channel probing [66], [116], [212].

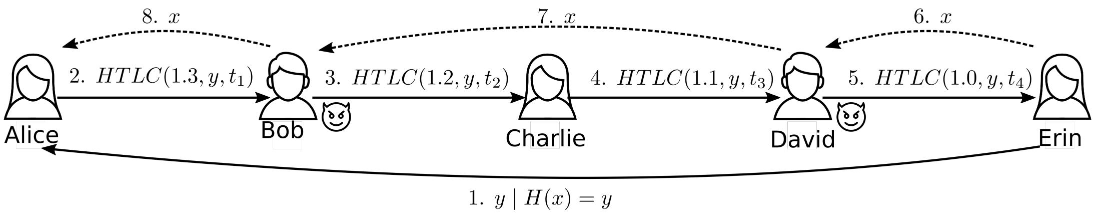

Figure 16. Example of a wormhole attack. Bob and David collude to steal routing fees that Charlie should claim. The solid lines indicate the steps of the payment-establishing phase, while the dotted lines indicate the payment-settling phase.

Kappos et al. [112] try to identify private channels in the Lightning Network using two heuristics to understand LN's privacy limitations. First, the authors use a property heuristic to identify an upper-bound number of private channels. They search for every Bitcoin transaction which occurred after January 12, 2018, when the Lightning mainnet was launched. In particular, the heuristic selects transactions that use a specific type of output address, called payment to witness script hash (P2WSH), which is used to open and close channels in the Lightning Network. Then, the authors filter these transactions with common channel opening and closing transaction features, such as having a 2-of-2 multi-signature address and having at most two outputs. The authors reach an upper bound of 77,245 pairs of transactions that could have been used to open and close private channels in the network. Then, the authors develop another heuristic, called *tracing heuristic*, to identify associated private channels. This heuristic verifies common patterns in opening transactions, such as using the output of a channel-closing transaction to open more channels or using the change in the channel-opening transaction to continue to create channels. The results show that the intersection between the two heuristics indicates 27,183 transactions which likely represent the opening of private channels.

3) Payment privacy: Payments contain sensitive information that can lead to attacks and failed payments if disclosed to unauthorized parties. According to the literature, the following requirements should be guaranteed for payments in PCNs [56], [102], [119]:

- **Balance security**: A PCN must ensure that an honest user in a path does not lose coins, even if all other nodes along the path are malicious or corrupted.
- **On-path relationship anonymity**: Intermediary nodes that route a payment should only know their immediate predecessor and successor in the payment path. Thus, the relationship between the sender and the receiver of a payment is private unless it is a single-hop payment.
- **Off-path payment privacy**: Nodes not involved in the payment path should not be able to obtain any information about a payment routed by honest nodes. This includes the payment value, path, and identifiers of senders and receivers.

While PCNs provide link privacy by design, payment privacy usually depends on the routing protocol. For instance, the Lightning Network's default routing protocol adopts onion source routing to guarantee on-path relationship anonymity and computes paths locally to preserve off-path payment privacy [209]. Some new proposals attempt to provide on-path payment privacy, i.e., to ensure the full payment path is unknown even to the sender and intermediaries via route blinding [101]. Route blinding, however, is yet to be implemented and tested in the Lightning Network. Other protocols, such as AODV-based routing [69] and CheaPay [81], leak payment information due to their collaborative path-finding approach. HTLCs guarantee balance security if nodes are always online (Section VI-E2 gives an example where this is not the case).

4) Privacy-utility trade-off: It is worth mentioning that, as in many computational systems, ensuring privacy in PCNs impacts performance. Not revealing channel balances in the network forces users to guess if a given path has enough balance to support a particular payment via trial-and-error. This trial-and-error approach causes concurrency issues that waste resources, increase average payment latency, and reduce payment goodput. On the other hand, PCNs cannot simply publish channel balances to improve performance since it would compromise user privacy. Tang et al. demonstrate that privacy and utility represent a trade-off for PCNs and that it is unfeasible to provide both simultaneously [103]. In the Lightning Network, central nodes with many resources mitigate this issue by opening large channels that can route payments with high probability at the core of the network. The approach, however, incurs centralization and is not effective for payments that must be routed in the network periphery.

5) Summary of Security and Privacy in PCNs and Discussion: Privacy in payment channel networks can be defined on several levels. First, PCNs rely on the underlying blockchain for node anonymity, which is usually accomplished by adopting public keys as identifiers. The PCN itself (or its channel announcement protocol) guarantees balance secrecy and gives the option of keeping the channel private by forcing users to announce their channels if they want them to be public. The channel opening transaction is inherently private due to the SegWit protocol and only a channel's node can prove that a multisig transaction is used to create a channel. Payment privacy, i.e., the secrecy of the sender-receiver pair, payment value, and payment path, depends on the routing protocol, which must ensure that no information about the payment can be leaked to off-path nodes. Lastly, PCNs present a privacy-utility trade-off that forces payment routing protocols and gossip protocols to either favor privacy or performance, both cannot be achieved simultaneously [103]. Efficient routing protocols in PCNs usually rely on weaker privacy assumptions.

E. Attacks in PCNs

1) Wormhole attacks: The Lightning Network has some security vulnerabilities despite ensuring privacy and security requirements.

Malavolta et al. [71] demonstrate the wormhole attack. In this type of attack, an attacker in the path of a payment colludes with or controls another node in the same path. Although the payment path is confidential, the attacker can easily check if another node under its control is in the payment path since the HTLCs in the Lightning Network have the same hash. Thus, upon receiving two HTLCs at different points with the same hash function, the attacker infers that the payment is on the same path.

Figure 16 illustrates the attack. Alice wants to send a coin to Erin. Each intermediary charges 0.1 coins as a forwarding fee. Bob and David collude to carry out the attack. In the first part, the attacker legitimately establishes HTLCs in the payment path with block hash $y \mid H(x) = y$. The receiver reveals the value of x to David to unlock the chain of payments, who passes it on directly to Bob without redeeming the payment in the channel he has with Charlie. Bob then redeems the payment on his channel with Alice. From Charlie's point of view, who does not get the x value, the HTLC failed. The wormhole attack allows the attacker to receive fees that would otherwise be destined for intermediary nodes. In the example, each intermediary would receive a 0.1 fee if Bob and David acted honestly. However, Bob receives 0.3 coins by performing the wormhole attack, which he can split with David. Despite not losing any coins, the wormhole attack hurts Charlie, who does not receive payment fees and must keep the coins unavailable until the HTLC expires.

Wormhole attacks are possible because relationship anonymity is not guaranteed when nodes on the same path collude [71]. To solve this problem, Malavolta et al. propose a new type of contract called multi-hop HTLC [56]. In this construction, each intermediary receives two hash values y_i and y_{i+1}, where $y_i = H(x_i)$ and $y_{i+1} = H(x_i \oplus x_{i+1})$, where $x_i \oplus x_{i+1}$ represents the logical "exclusive-or" operation between x_i and x_{i+1}. In addition to these values, intermediaries also receive the value x_{i+1} and a proof that $\exists x_i \mid y_i = H(x_i), y_{i+1} = H(x_i \oplus x_{i+1})$. This proof uses zero-knowledge proof (ZKP) techniques, which allow the sender to prove a claim without revealing secret information pertinent to the claim [236]. Thus, as in HTLCs, disclosing x_i is enough to unlock the entire payment chain since x_i is the only information intermediaries omit. This construction guarantees the anonymity of the relationship since each intermediary has a different hash, making it difficult to associate payments in the same chain.

The authors develop a proof-of-concept implementation of their proposal in Python to analyze its feasibility. They verify that a sender takes 309 ms to create each proof for intermediary nodes. Furthermore, each proof has 1.65 MB and each intermediary takes around 130 ms to verify the proof. As an example, the authors use a path with 5 users and verify that the overall computation overhead is 1.32 seconds and the communication overhead is 5 MB.

2) Coin Theft: In any payment channel network, it is possible to steal coins if one of the parties on the channel disconnects for a sufficiently long period. PCNs such as the Lightning Network [54] and the Raiden Network [89] assume that any node transacts in the network remains online as long as the channel is open. Otherwise, one of the channel parties can terminate it by publishing an old transaction[13], which invalidates sent coins and effectively recovers them for the sending party. The system punishes this type of malicious behavior by allowing the victim to spend all the coins in the channel, including those of the malicious party, if they recover from the disconnection during a predefined lock time window. Therefore, the attack is only worth trying if the malicious node can guarantee that the other party will not check the blockchain during the dispute period, which remains until the time window expires.

In networks with fast and reliable connections where all users have a copy of the blockchain, a tiny default value for block time windows allows victims to recover and punish their attackers in time. In such cases, users can detect malicious behavior instantly, without trusting a third party, simply by synchronizing their blockchains and verifying the most recent blocks. However, in heterogeneous scenarios with many users, especially mobile devices, some nodes may disconnect for long periods or indefinitely. Device downtime is especially challenging for use cases where the direction of payments is biased to one side, such as when a seller uses her device to receive transactions from multiple buyers. In this case, the payment channels are expected to be highly imbalanced concerning one of the parties.

Channel imbalance indicates a greater vulnerability to the coin theft problem, as demonstrated in the following formulation. Let two devices b and s, which represent devices of a buyer and a seller, respectively, connected to the routers r_1 and r_2 through payment channels as shown in Figure 17. Each payment channel $u \leftrightarrow v$ has a balance $bal_{u \leftrightarrow v}(t) = (bal_u(t), bal_v(t))$, where $bal_u(t)$ and $bal_v(t))$ are the balances of nodes u and v at time t, respectively. Note that $bal_u(t) + bal_v(t)$ is constant. For payment channels between buyers and routers, for example, $b \leftrightarrow r_1$, the initial balance is $bal_{b \leftrightarrow r_1}(0) = (\alpha, 0)$, where α is the number of coins that the buyer b reserves for payments in the channel. Likewise, the initial balance of payment channels between vendors and routers, for example, $r_2 \leftrightarrow s$ is $bal_{r_2 \leftrightarrow s}(0) = (\beta, 0)$ where β is the number of coins that the router r_2 reserves to forward payments to the seller s. The formulation

[13] Also described as a "revoked state" in the literature [54], [55], [127].

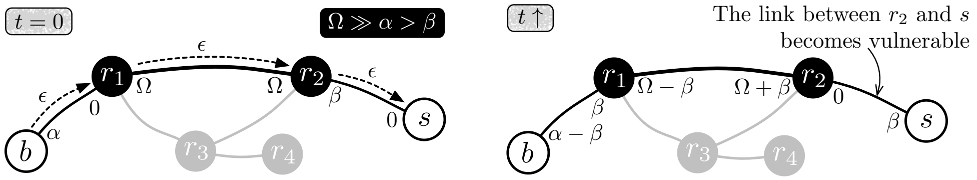

Figure 17. An example of the coin theft vulnerability in payment channel networks [127]. On the left, a continuous amount of ϵ coins flows from a source b to a destination s until the channel capacity between router r_2 and s is exhausted. Then, on the right, s becomes vulnerable if it loses connection because r_2 has nothing to lose if it tries to close the channel with an old balance.

assumes for simplicity and without loss of generality that s and b only participate in one payment channel.

Considering a scenario where a payment of ϵ coins occurs from b to s, r_2 and s sign a commitment transaction $Tx(1)$ containing the new channel balance $bal_{r_2 \leftrightarrow s}(1) = (\beta - \epsilon, \epsilon)$. If s disconnects indefinitely after subscribing, r_2 can close the channel with the previous transaction $Tx(0)$ and recover ϵ coins. Doing so is risky because r_2 would lose all of its coins if s recovers and detects the malicious behavior before the end of the dispute period. However, as s receives more payments, the balance at $r_2 \leftrightarrow s$ converges to $bal_{r_2 \leftrightarrow s}(t) = (0, \beta)$. If that happens, r_2 has little to lose by closing the channel with a previous transaction, even if s recovers in time. This strategy is optimal for any rational r router when its payment channel to a seller runs out. Malicious nodes may also attack in intermediate cases, depending on the risk-benefit ratio. Therefore, the traditional security mechanisms of payment channel networks do not prevent routers from adopting this strategy. The seller s is subject to coin theft even without malicious behavior. Although formulated for an extreme case of buyers and sellers, the problem applies to any situation where a node receives payments and disconnects without properly closing the channel.

The coin theft problem becomes even more significant when the network nodes are mobile devices with intermittent connectivity and disconnections for long periods. Some works propose improvements such as time windows adapted to the device connectivity profile, hiring "watchdog" nodes that constantly check the blockchain to detect channels that were closed improperly, or reputation systems in which nodes would punish the malicious behavior by issuing opinions about routers [114], [127]. However, these solutions still need to be deeply explored, have privacy issues, and can lead to system centralization [146], [237]. Major payment channel network implementations do not currently have an effective solution for these cases [88], [89].

3) Flood and Loot: As the name suggests, the flood and loot attack [104] has two steps: flood, when the attacker attempts to publish multiple transactions in the blockchain simultaneously, and loot, when he steals the coins from a victim. Flood and loot mitigation consists of limiting unresolved HTLCs.

Initially, the attacker creates two different addresses, one to act as a source and the other as the destination of payments. It also creates direct channels between those fake addresses and the victim. In the second step, the attacker sends a payment from the source to the destination using the victim's channels as an intermediate. To execute the payment, the attacker creates an HTLC between the source and the victim, and the victim creates an HTLC between itself and the attacker's destination. Once the destination receives the transaction, it follows the protocol and sends the secret to the victim. Currently, there are no locked coins between the victim and the destination. The victim now tries to redeem the coin between itself and the source. The source, however, does not follow the protocol, forcing the victim to close the channel and claim coins on-chain. The attacker does the same to many victims, creating a flood in the transaction mempool, i.e., the list of all pending transactions in the Bitcoin network. Since the blocks have a limited number of transactions, the victims may need to wait a long time for their transactions to be published, triggering the HTLC's timeout clause.

Once the HTLC reaches its timeout on the blockchain, the source sends a transaction to the network claiming the coins with a higher transaction fee, prioritizing it over the victims' transactions. If the victim cannot confirm the transaction before the attacker, the attacker steals the victim's coins by validating the channel's last state. It is worth mentioning that the attacker does not publish a revoked state; instead, it validates the legitimate HTLC-timeout transaction. Thus, even though the honest user detects malicious behavior, the attacker cannot be punished as he would be in the coin theft attack.

Since the attack is executed based on a race condition triggered by the replace-by-fee policy, the network could reduce the number of conflicting transactions by limiting the number of simultaneous unresolved HTLCs. Another more realistic approach is to implement a reputation system to prevent an honest user from executing payments of attackers.

4) Route Hijacking: The Lightning Network has a topological problem: it is highly centralized in a few nodes [79], [87]. Besides, the channel fees are publicly available. Based on this information, Tochner et al. [84], [105] proposed the route hijack attack, in which the attacker node increases its centrality to be present in victims' routes. To do this, the attacker strategically opens new channels with central nodes to have fewer hops for every possible destination. Moreover, it charges low fees to attract users. Finally, when many users

Table VII
ATTACKS IN PAYMENT CHANNEL NETWORKS AND POSSIBLE COUNTERMEASURES.

Attack	Description	Effect	Threat Level	Countermeasure
Coin theft [54]	Publish old transaction in the blockchain	Loss of coins	medium	Watchtowers [114] and minimum lock time windows [127]
Wormhole [71]	Bypass HTLC secret to steal fees	Loss of routing fees	high	Multi-Hop HTLC [56]
Flood and loot [104]	Flood the network and execute HTLC timeout to retrieve coins	Loss of coins	low	Limiting unresolved HTLCs
Route hijack [84]	Offer lower fees to hijack routes	Loss of routing fees	medium	Introduce randomness to pathfinding [105]
Griefing [85]	Lock channel liquidity with unsolved HTLCs	Denial of service	medium	Griefing-Penalty [238]
Amount jamming [120]	Exhaust the channel capacity with unsolved HTLCs	Denial of service	medium	Unconditional fees and peer reputation [132]
Slot jamming [239]	Exhaust the maximum limit of HTLCs with unsolved HTLCs	Denial of service	high	Unconditional fees and peer reputation [132]
Balance discovery [86], [107], [108], [136]	Estimate balance by sending payments and analyzing error response	Privacy disclosure	high	Generic error responses

choose the attacker as a hop in their path, it starts a denial of service (DoS) attack on the users, refusing to fulfill payments. Then, attacked users must wait until the HTLC timeout to retrieve their locked coins. This attack is similar to the black hole attack in ad hoc networks [240], where the attacker pretends to have the best routes to a destination and then drops all the received packets.

The cause of this attack is the reasonable and predictable behavior of users in the payment channel network: Nonetheless, reasonable users with many resources and high centrality are expected to prefer receiving profits from routing fees instead of delaying payments. They choose the routing path that minimizes the total fees. Hence, introducing random behavior or other path-finding metrics could prevent the attack. Increasing the fees in a path can also improve security, which makes the attacker's channels less attractive. Tochner et al. [84] name this trade-off between fee and security the "price of predictability". The authors also mention that opening 30 channels is enough to hijack 80% of the routes in the Lightning Network.

5) Griefing Attacks: Another possible malicious action is to delay the resolution of HTLCs by executing griefing attacks [85]. The attack is feasible because a user can withhold payments, even if that seems illogical and unprofitable. The attacker needs to be a destination or an intermediate node of payments in the network to execute this attack.

Recall that HTLCs guarantee that each payment can only be claimed either by revealing the payment's secret or by releasing the locked coins after a timeout. In griefing attacks, the attacker withholds the payment secret to the previous hop for as long as possible. By doing this, all nodes between the attacker and the source are forced to wait, possibly until the HTLC timeout, to claim their locked coins. Thus, latency in the network increases, which causes a decrease in the throughput of channels that link the source and the attacker. If the attacker executes this procedure many times, it can prevent the involved channels from forwarding payments since all channel capacity is locked.

Mazumdar et al. [238] highlight that griefing attacks occur because the network does not punish attackers for withholding HTLCs. The authors propose the implementation of a griefing penalty, which consists of compensating the victims of delayed transactions with financial restitution. The authors propose to modify HTLCs to add the compensating fee, which is proportional to the time the victims wait to retrieve their coins. Thus, with Hash TimeLock Contracts with Griefing-Penalty (HTLC-GP), the attacker (specifically, the destination node) would have to pay a compensating fee to all nodes in the attacked path.

6) Channel Jamming: Channel jamming, also known as congestion attack [120], consists of blocking the liquidity of a channel in the Lightning Network by executing false payments [239]. The attacker controls the source and destination of payments, routing them through the victims. Like a griefing attack, the attacker exploits HTLCs by making the victims wait until the expiration to redeem their funds. However, in channel jamming, intermediary nodes are the victims.

Initially, the attacker issues a payment from an address controlled by itself or some colluding node. After establishing HTLCs with each intermediary in the payment route, the destination refuses to reveal the secret. This locks channels in the payment path until the HTLCs time out. Moreover, since the destination refuses the payment, no routing fees are paid, and the attack cost is zero. The attacker only needs enough resources to lock channels temporarily. If the attacker executes this procedure repeatedly, it could cause a denial of service attack to the nodes on the route.

There are two types of channel jamming in PCNs: amount and slot jamming. In the amount jamming, the attacker attempts to exhaust channels. The attacker needs to possess a large number of coins and block them during the attack. It sends a high-value payment and refuses it at the destination, effectively locking up the capacity of channels in the payment path. The attacker must also correctly estimate the channels' balance since channels would refuse payments over their route capacity. Hence, in amount jamming, attackers often probe channel balances before attacking.

On the other hand, slot jamming targets the protocol's maximum limit of simultaneously unresolved HTLCs. Because HTLCs are implemented as if-then-else conditions inside the script area of commitment transactions, establishing a large number of HTLCs increases the size of the commitment transaction. In the Lightning Network, the maximum size of a Bitcoin transaction imposes a limit of 483 simultaneous HTLCs per channel. Otherwise, the channel-closing transaction would be too large to be validated on-chain. Slot jamming exploits this restriction by creating many small payments that occupy the available slots in a payment channel. Eventually, no HTLCs can be served.

The solutions to channel jamming often rely on peer reputation. The core idea is that nodes analyze the HTLC initiator's reputation before taking action. For example, suppose an HTLC initiator has a low reputation. In that case, the HTLC receiver can limit its maximum amount and number of unresolved HTLCs to a small value, charge extra fees for compensating the risk, or even refuse to forward the payments. Shikhelman et al. [132] propose such a peer reputation system with extra fees. Reputation-based systems, however, can compromise user privacy [237]. A new solution called "boomerang payments" proposes to modify HTLCs to include adaptor signatures [106]. Adaptor signatures enable reversible HTLC-type forwarding, i.e., they allow established HTLCs to be canceled before timeout if needed, which would prevent channel jamming. Spear proposes HTLCs with two hash digests to accomplish the same goal [121]. Instead of using a coding technique such as Boomerang, Spear utilizes ARQ (Automatic Repeat Request) to achieve message transmission reliability through unreliable channels. The receiver informs the sender about the partial payments completed. The proposals, however, impose structural modifications to HTLCs that are yet to be tested in large-scale PCNs. So far, channel jamming remains a significant concern and an open issue in payment channel networks.

7) Balance Discovery: The balance discovery attack tries to break channel privacy, estimating the balance between the channel participants [86], [107], [108], [136]. Firstly, the attacker sends a payment to a controlled address with a high-value transaction. Since the path must have sufficient funds, the transaction will likely fail. Then, the attacker repeatedly decreases the transaction value until the payment is completed. This condition determines the bottleneck channel balance from the origin to the destination. Since the channel capacity is publicly available, the attacker can calculate the reverse path balance. Besides, the attacker can execute this attack without cost by refusing the transaction in the destination, since it controls the address. The procedure, however, can only estimate the balance of the channels during a short period, since the network evolution will change the balances over time.

8) Summary of PCN Attacks and Discussion: Table VII summarizes the attacks in payment channel networks. The threat level column indicates on a scale of low, medium, and high what the attack feasibility and the resources compromised if the attacker succeeds. It is possible to identify three significant effects of the attacks: loss of coins, privacy disclosure, and Denial of Service (DoS).

Despite researchers and developers discussing security breaches in PCNs, there is no evidence that they already happened or are feasible in the current network. The attacks that involve the loss of coins are the most difficult to execute since they usually rely on the blockchain to recover the coins, except the wormhole attack that steals routing fees.

Privacy can also be compromised during short periods of time by executing the balance discovery. Although this is a low-cost attack, PCNs are highly dynamic, hence, the balance measurement might be unreliable after minutes or even seconds. Finally, we highlight that DoS by slot jamming is currently a significant threat to payment channel networks because it is easily feasible even for users with low resources. Thus, future research in PCN security should focus on mitigating channel jamming attacks due to their potential harm to the network.

F. Payment Concurrency

Executing a payment in a PCN is similar to circuit-switched networks in that the network must reserve a path with enough resources before sending data. In PCNs, however, users perform path reservations via HTLCs in a decentralized way without a global view of concurrent payments. Nodes in payment channel networks also forward multiple payments simultaneously in a best-effort manner without intelligent scheduling. Such behavior can result in individually feasible payments that block each other when routed or even deadlocks that block both payments. We show an example of this problem in Figure 18. In this section, we briefly present the main proposals in the literature for dealing with payment concurrency.

1) Fulgor and Rayo: Malavolta et al. [56] propose two mechanisms for mitigating payment deadlocks: a blocking protocol that aborts both payments, named Fulgor, and a non-blocking protocol that guarantees at least one payment completes, called Rayo. In Fulgor, two deadlocked payments will stay blocked until the HTLC timeout expires. When this happens, the algorithm sets a random waiting period to reduce the probability of a new deadlock without disclosing payment information. The approach preserves payment privacy at the expense of payment goodput. In Rayo, the authors assume a global payment index that acts as a priority list. Instead of performing best-effort forwarding, nodes verify the global index and abort payments with lower priority. This approach improves payment goodput because one of the payments gets unblocked instantaneously but weakens payment privacy since the transaction identifiers are available to all nodes in the network. It is unclear how this global priority list would be

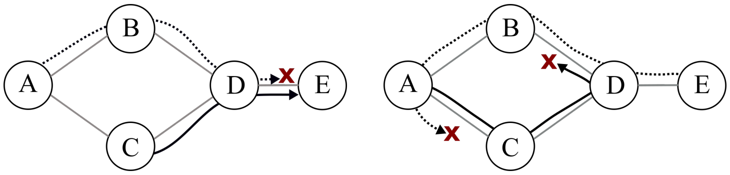

Figure 18. Simultaneous payments occurring in a payment channel network. On the left, a race condition between two payments occurs on channel *(D,E)*, causing one of the payments to fail. On the right, two payments reserve resources from each other's path on channels *(A,C)* and *(D,B)*, creating a deadlock in the network. Race conditions and deadlocks can result in failed payments and wasted resources if untreated.

maintained in a large-scale environment. Nevertheless, the authors do not evaluate their proposal in term of payment concurrency.

2) Pre-locking Channel Liquidity: Werman and Zohar [72] propose to modify Fulgor by introducing a locking phase before the establishment of HTLCs. During this phase, a sender asks nodes in the path to reserve resources for a time T much shorter than an HTLC timeout. Nodes ignore other payment requests during T. This way, the authors claim, the network can ensure only one payment occurs per path and that a rejected payment will be blocked for at most T time units. Rohrer et al. [62] introduce a similar mechanism called capacity locking to improve payment goodput when routing. The experiments show that their proposal provides a higher payment success rate when the number of flows and the transaction volume increase compared to the sequential approach. Their proposal has a success rate above 75% while the sequential approach is near to 0% when the number of flows is 2^{12}. The success rate decreases abruptly in term of the transaction volume for the sequential approach, while their proposal softens this behavior, exhibiting a difference of approximately 50% for a transaction volume of 25. However, sending requests compromises privacy since it discloses payment information and allows attackers to perform denial-of-service attacks via fake requests.

3) Summary of Payment Concurrency and Discussion: Many payment-sending mechanisms in PCNs lock paths based only on the user's view without considering information about other payments [55], [66], [80], [116]. This can create deadlocks which heavily impact throughput in the network. Nevertheless, the topic of payment concurrency still needs to be extensively studied. Besides Fulgor, Rayo, and its improvements, the literature lacks works which explore and propose mechanisms to solve concurrency issues. We speculate that this is due to the difficulty of preserving privacy and scalability in such mechanisms, as obtaining information from other payments incurs extra processing and may reveal sensitive payment data. Besides, payment concurrency issues do not appear frequently in current PCNs as they still lack a massive adoption that would cause heavy loads. Nevertheless, proposing an efficient protocol for dealing with payment concurrency is a fundamental challenge for PCNs as it can dramatically improve their performance in the long run. Also, most payment concurrency proposals lack quantitative evaluation results. An interesting research area is to define experimental scenarios, such as network topologies and client workloads, to compare proposals and quantify their improvements.

G. Load Balancing, Congestion Control, and Payment Scheduling

Besides dealing with deadlocks, PCNs face challenges similar to traditional packet-switched networks. For example, when multiple payments traverse the same hops simultaneously, some channels may become congested or exhausted. Since payments have a deadline to fulfill and channels might run out of capacity under heavy loads, payment scheduling should intelligently control the load balancing of channels to provide network stability and achieve maximum payment goodput. Some authors also propose transport protocols beyond simple forwarding to improve payment delivery [90], [109], [124].

1) Levels of Load Balancing: In their survey on PCNs, Papadis and Tassiulas [109] consider that payment load can be balanced at three levels: channel, path, or node level[14]. Load balancing can be achieved in each level as follows:

- **Channel level.** In channel-level load balancing, the goal is to evenly distribute a load of channels to avoid overloading any particular one. In this case, a node might adjust the rate at which it serves payments in each channel so that the channels preserve enough funds in all directions. This can be achieved with hop-by-hop forwarding decisions and fee adjustments that make channels more or less attractive.
- **Path level.** In path-level load balancing, load between pairs of sources and destinations should be distributed across different paths, similar to the techniques used in the load balancing of Internet traffic. Most multipath payment routing protocols consider per-path balancing since it improves payment goodput [67], [80], [90], [116].
- **Node level.** Finally, in node-level load balancing, users might want to avoid central nodes that process too many payments since these could incur extra delays and pos-

[14] Note that the term "load balancing" here is not to be confused with the channel rebalancing techniques discussed in Section VI-B. In this case, we want to schedule payments optimally so that all payments can be delivered without disrupting the network.

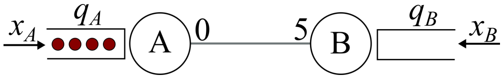

Figure 19. An example of payment queues q_A and q_B in Spider routers (adapted from [90]). Red circles represent unit payments. When the payment arrival rate at A, x_A, is greater than the arrival rate at B, x_B, A eventually runs out of funds, and its payment queue q_A starts to grow. If payments stay in the queue for too long, A will send a message to the payment senders as a warning to reduce their payment rates.

sibly payment failures. Node reputation and monitoring are the main strategies to achieve node-level balancing.

2) Congestion Control with Spider: The Spider routing protocol introduces routers that keep payment queues and adjust payment rates according to the channel load [90]. Figure 19 shows an example of such routers. Their work provides congestion control to PCNs and is inspired by the mechanisms of Multipath TCP for communication networks [241]. The idea is that payments that cannot be forwarded will be stored in a queue instead of failing immediately and will be served as soon as a payment comes in the opposite direction. This way, payments can tolerate ephemeral load changes without reducing payment goodput. Also, if a payment stays too long in a queue, the corresponding router sends a warning message to the sender, which proactively limits the payment rate for paths that traverse that channel. The work also indirectly provides load balancing at the path level, as the payment sender will try a different path when receiving routers' warnings.

3) Payment Scheduling with PMDE: Papadis and Tassiulas formalize and study the payment scheduling problem in depth, paving the way for new scheduling policies [124]. They propose to replace the default scheduling mechanism in a payment channel, in which payments are forwarded or rejected as soon as they arrive, with payment buffers and nodes that schedule payments following some priority. They introduce the Process or Match on Deadline Expiration (PMDE) policy, which attempts to forward a payment only on a given deadline and prove its optimality for fixed arriving payment amounts. The policy also ensures payments are only served if matched with a payment in the opposite direction, preserving channel balances. To our knowledge, their work is the first to propose an optimal payment scheduling policy for PCNs.

The authors build a discrete event simulator of a PCN to evaluate the proposed scheduling policy. They compare their proposed PMDE with three other policies: process at regular intervals with immediate processing (PRI-IP), without immediate processing (PRI-NIP), and process feasible immediately (PFI). The PMDE policy achieves higher normalized throughput under symmetric demand compared to other policies. In particular, while PMDE achieves near 100% normalized throughput, PRI-IP and PRI-NIP stop at around 90% of normalized throughput. When the payment demands are asymmetric, however, PMDE performs similarly to the rest of the evaluated policies, with a normalized throughput of 80%.

4) Summary of Load Balancing, Payment Scheduling, Congestion Control, and Discussion: Payment concurrency, load balancing, congestion control, and payment scheduling are some of the least explored challenges of payment channel networks. One plausible explanation is the difficulty of enforcing systematic modifications in a decentralized network. Out of the discussed topics, congestion control receives more attention as it is essential for payment routing protocols [90], [96], [116]. However, we expect more works on payment scheduling will appear in the following years.

H. PCN Simulation

Payment channel networks are an emerging technology that still needs large-scale deployment on real-world applications. Consequently, most PCNs today run on small networks of tens of nodes [89] or rely on proof-of-concept implementations [57], [63]. Lightning is a large-scale PCN, but obtaining data from it is difficult due to privacy concerns. Therefore, there is a need for PCN simulators that can effectively mimic a real PCN while allowing researchers to experiment with different network conditions that represent potential applications in the future. We present some PCN simulation proposals below.

1) PCNsim: PCNsim [123] is an open-source payment channel network simulator that reproduces the behavior of a PCN on top of the OMNeT++ simulation environment. The simulator follows the specifications of the Basis of Lightning Technology (BOLT) [212] to simulate the messages exchanged between nodes involved in a payment. The simulator also allows users to model channel parameters, such as capacity and fees, based on real data. It includes network topology and workload generator modules that allow researchers to test their proposals under different networking conditions. The main features of PCNsim are the flexibility of its modules and the reproduction of the Lightning Network payment state machine on top of OMNeT++, which supports testing PCNs with several underlying network protocols using the INET framework[15]. However, the current version of the simulator only implements single-path Dijkstra's SPF algorithms for routing and lacks quantitative comparison to the real LN operation.

2) CLoTH: CLoTH is a PCN simulator that produces performance measures such as the probability of payment success and the average payment latency [111]. Like PCNsim, CloTH aims to mimic the behavior of the Lightning Network according to its documented specifications. The current version of the simulator implements LN's default payment state machine, Dijkstra's SPF routing protocols, and multipath payments. The main difference between CLoTH and PCNsim is that CLoTH simulates the Lightning Network in pure C language, whereas PCNsim is built on top of the OMNeT++ environment. Although this difference is insignificant in terms

[15] Available at https://inet.omnetpp.org/Introduction.html

Table VIII
COMPARISON BETWEEN THE EXISTING PCN SIMULATORS IN THE LITERATURE.

Reference	Language	Source code	Main features	Main limitations
CoinExpress [66]	C++/Python	Unpublished	- Based on the ns-3 network simulator - Implements several routing algorithms	- Unpublished source code - Does not support graph inputs - Unclear whether the simulator follows LN specifications
LNSim [65]	C++	https://github.com/gdistasi/LNSim	- Simulates a simplified version of the LN - Generates networks via input or randomly	- Simplifies HTLC functions - Does not support underlying networking protocols
Blyskavka [64]	Java	Unpublished	- Simulates a simplified version of the LN - Uses MASON as a simulation engine	- Unpublished source code - Simplifies HTLC functions - Does not support graph inputs - Limited to 10,000 nodes - Does not support underlying networking protocols
Spider [90]	C++/Python	https://github.com/spider-pcn/spider_omnet	- Based on the OMNeT++ network simulator - Implements routers with congestion control - Implements several routing algorithms - Generates networks via input or randomly	- Simplifies HTLC functions - The implemented routers do not follow LN specifications
LNTrafficSimulator [91]	Python	https://github.com/ferencberes/LNTrafficSimulator	- Simulates a simplified version of the LN - Generates traffic automatically based on LN snapshots - Generates networks via input or randomly	- Limited to fixed-amount payments - Simplifies HTLC functions - Does not support underlying networking protocols
CLoTH [111]	C/Python	https://github.com/marcono/cloth	- Accurately reproduces the LN specifications and code functions - Implements multi-path payments	- Does not support underlying networking protocols
Kappos et al. [112]	Python	Unpublished	- Simulates a simplified version of the LN - Focuses on PCN privacy guarantees - Supports graph inputs from LN snapshots	- Unpublished source code - Simplifies HTLC functions - Does not support underlying networking protocols
Papadis and Tassiulas [124]	Python	https://github.com/npapadis/payment-channel-scheduling	- Implements payment queues/buffers - Supports several single-hop payment scheduling policies	- Restricted to single-hop payments - Does not implement HTLCs - Does not support underlying networking protocols
PCNSim [123]	C++/Python	https://github.com/gfrebello/pcnsim	- Based on the OMNeT++ network simulator - Accurately reproduces the LN specifications and code functions	- Does not support multi-path payments

of the implementation of Lightning Network functions, it means CLoTH's simulation is restricted to the application layer, while PCNsim can simulate the underlying communication network using OMNeT++. Like PCNsim, CLoTH lacks a comparison to the real operation of the LN to assert the simulation accuracy.

3) Spider: Spider [90] develops an event-based simulator for payment channel networks besides proposing a routing protocol. The simulator extends the OMNeT++ simulation framework to model a PCN with Spider routers, providing the congestion control functionalities proposed in the paper. The Spider simulator offers three types of network topology for PCN simulation: the LN topology on July 15, 2019, a Watts-Strogatz small world [242], and a Barabasi-Albert topology [243]. Furthermore, the simulator draws transactions from a credit-card dataset [244] to model transaction size. The code is open-source, and the authors use the simulator to extract statistics and compare their proposal with other routing protocols.

The authors prove their simulation is sound by comparing the transaction goodput of the simulator with a real Lightning Network implementation. The comparison focuses on evaluating the Spider simulator against a modified version of the LND implementation of the Lightning Network. The modified LND implementation queues up HTLCs in intermediary nodes following the Spider design. The results show that the average payment success ratio on the simulator is within a 5% margin of the modified LND. This result is the only quantitative comparison between the Spider simulator and a Lightning Network implementation. The evaluation ignores other important network features, such as payment latency and throughput, and mainly focuses on Spider's routing protocol proposal while simplifying the core functionalities of PCNs, such as payment state control messages and HTLCs. Thus, it is challenging to use Spider as a generic simulator to test other payment strategies, for example.

4) Blyskavka, LNSim, and LNTrafficSimulator: Other proposals implement simplifications of the Lightning Network. Piatkivskyi and Nowostawski [64] develop Blyskavka, a Lightning Network simulator written in Java, to evaluate the impact of payment splitting when routing. Blyskavka simulates the Lightning Network operation rather than the Lightning Network itself, meaning it does not implement its specific messages and states. It also simplifies HTLC simulation by only blocking and releasing payments on the path after a short delay. Stasi et al. [65] develop LNSim, an LN simulator, to evaluate a novel fee definition strategy and a multipath routing heuristic. Their open-source simulator can simulate the network at the LN protocol level but does not implement HTLCs. Beres et al. [91] develop LNTrafficSimulator, a Lightning Network traffic simulator based on LN public data, to analyze the economic and privacy implications of payments. Their work focuses on single-hop payments and simplifies other PCN functionalities.

5) Simulators for Specific PCN Functionalities: Several other proposals implement simple simulators for specific purposes. Kappos et al. [112] develop a PCN simulator in Python to evaluate whether an on-path adversary can successfully identify the payment sender. Their simulator uses publicly available Lightning Network snapshots and information published by central node owners, but the simulation code is yet to be published. CoinExpress [66], [245] develop a PCN simulation tool to test their routing proposal on top of the *ns-3* discrete-event network simulator. The simulator creates a random Watts-Strogatz network with random payments between users. The paper, however, does not clearly describe the simulator functionalities, and no source code is available. Papadis and Tassiulas [124] develop a discrete event simulator of a payment channel with support for transaction buffers. Their simulator aims to evaluate the impact of several single-hop payment forwarding strategies. Consequently, the simulator focuses on scheduling policies instead of providing a complete simulation of PCN functionalities.

6) Summary of PCN Simulation and Discussion: We highlight the main differences between the discussed simulators in Table VIII. Besides CoinExpress, Blyskavka, and Kappos et al., all simulators provide open-source code that can be tested at will. Most simulators implement a version of the Lighting Network, with PCNsim and CLoTH being the only simulators that fully reproduce the phases involved in a payment process [64], [65], [91], [111], [112], [123].

Despite the recent proposals, there are still open research opportunities in PCN simulators. Some simulators, such as PCNsim [123] and CLoTH [111], lack comparisons to the real operation of the Lightning Network to evaluate their accuracy and improve their design. Future simulators could focus on creating PCN simulators for networks other than the Lightning Network [89], [202], [220]. Furthermore, defining simulation parameters and inputs is still challenging. A significant number of PCN simulators uses either the Ripple transaction dataset [68], [220] or a credit-card payment dataset [244] for workload simulation [68], [80], [90], [115], [123]. Using both of these datasets to generate transaction workload, although widely accepted, might result in inaccurate results. The most accurate approach for workload generation uses real payment data collected from a central node that runs on the Lightning Network. Another challenge when designing a PCN simulator is choosing nodes to act as sender-receiver pairs. Collecting real data from a central node becomes useless, given that intermediary nodes know only their predecessor and the following node on the payment path in the onion routing operation. Thus, PCN simulators have few options to select potential sender-receiver pairs. First, simulators could assume a payment distribution and draw nodes from it. This approach has the obvious downside of the uncertainty on how realistic the produced synthetic data is. A second approach is to assign roles based on the available payment data. For example, simulators could identify similarities between the Ripple network and the Lightning Network and use this information to associate nodes on the Ripple payment dataset [220] with nodes on the Lightning Network. The downside of this approach is that it assumes that payments in different PCNs are similar. A third approach could use state-of-the-art traffic analysis from the Lightning Network to select sender-receiver pairs [91]. Nonetheless, all of the above-mentioned approaches present drawbacks derived from PCN's privacy-preserving mechanisms.

I. Support for Light Nodes

Despite having several implementations for computer networks, payment-channel networks still present open challenges related to resource-constrained devices such as mobile phones, smart objects, and sensors. The main problem is that current PCNs assume nodes with high availability, large storage capacity, and high computational power. For example, most routing protocols assume nodes can store and synchronize a copy of the complete topology to find paths [67], [80], [90], [116]. The Lightning Network and the Raiden Network require nodes to store a full copy of the blockchain by default and adopt onion routing to provide payment privacy at the expense of extra processing [54], [89], [119]. Such assumptions create a challenge for wireless devices with limited resources and intermittent connectivity, devices that today account for over half of all the traffic on the Internet [122]. Moreover, as discussed in Section VI-E2, new vulnerabilities appear when light devices are present in the network [127]. This section presents the main proposals that adapt payment channel networks to consider payments with light devices.

1) LNGate: Kurt et al. [113] propose LNGate, a threshold cryptography-based protocol that allows light devices to interact with the Lightning Network via untrusted gateways, store the blockchain, and process payments. In LNGate, any operation, such as opening a channel or sending a payment, only happens if it is signed by both the light device and its corresponding gateway. Payment processing is thus delegated to gateways without compromising security once gateways cannot execute operations without the user's knowledge. In an extended version of the work, the authors use game theory to analyze the security of the proposal and test it with different underlying communication protocols. They also prove that the extra signing step incurs negligible delay to operations [125]. Specifically, using WiFi incurs a 1.07 seconds delay and Bluetooth Low Energy (BLE) takes around 3 seconds while using the usual time, with no IoT device, takes 0.31 seconds. The simulation results also show that the IoT device consumes 3.56 mWh when the user sends 100 payments, while its consumption is usually around 3.03 mWh when idle. Nevertheless, LNGate requires changes to the core of the Lightning Network protocol, which are difficult to enforce in a decentralized environment.

2) Hannon-Jin Protocol: Hannon and Jin [78] propose a watchdog-based protocol and demonstrate its security and fairness using game theory. In their work, light nodes open payment channels with gateways and rely on third parties called watchdogs to monitor the blockchain for possible coin-stealing attacks in the channel. This concept is similar to watchtowers [114] in the Lightning Network. The watchtowers receive financial incentives for the monitoring service, which allows light devices to go offline without losing funds. This approach, however, is vulnerable to collusion attacks between a malicious party and the watchtower and assumes the light nodes send every transaction to the watchtower before disconnecting. The work does not specify how a light node opens a payment channel with the gateway or finds paths without having enough computational resources to store the network topology. It also lacks a quantitative comparison with other proposals.

3) IoTBnB: Robert et al. [92] propose an integration of the Lightning Network with existing large-scale IoT ecosystems called IoTBnB. IoTBnB, which stands for "IoT service for puBlication and Billing", is a digital marketplace where buyers pay for commercialized data using the Lightning Network. They propose to use a Lightning Network gateway module which stores the blockchain and the network topology. The light devices delegate the operations of opening/closing channels and sending payments to the gateways instead of processing them locally. The work, however, assumes gateways are not malicious since they are part of a trusted IoT platform. A similar approach is used in Lightning Service Providers (LSP) [126]. LSPs are companies that provide access to the Lightning Network, analogous to traditional Internet service providers. Users with light nodes trust LSPs to manage their channels and send/receive payments on their behalf, simplifying the user experience and allowing devices to receive payments offline.

4) Other Architectures: Mercan et al. [93] and Rebello et al. [127] present alternative lightweight PCN architectures that focus on reducing computational requirements for mobile devices. In their works, light nodes connect to gateways via payment channels and monitor the blockchain on-demand by downloading blocks from random nodes. Their solution, however, implies that devices have a minimal processing capacity and stay offline only for short periods. Such an assumption does not cover light devices that may disconnect for weeks or months, nor devices that are so restricted in resources that they cannot even establish and verify payment channels. Dealing with such devices is an important issue that needs to be addressed in the current literature.

5) Lessons Learned on Support for Light Nodes: Resource-constrained devices represent new challenges for PCNs, such as dealing with nodes that have intermittent connectivity and limited processing and storage capacities. All proposals leverage gateways to relay payments on the light node's behalf but differ in how they guarantee security. LNGate [113], [125] adds support for threshold cryptography into the HTLC exchange protocol, which prevents gateways from performing actions that do not have the signature of the light node. Instead of enforcing signatures, Hannon and Jin [78] adopt watchdogs to monitor the actions of gateways and punish them if needed. This concept is similar to watchtowers in the Lightning Network [114]. IoTBnB and Lightning Service Providers provide their gateway nodes and assume gateways are trusted because they are controlled by a trusted entity [92], [126]. Other architectures, namely Mercan et al. and Rebello et al., assume light nodes have a minimum capacity to download data from gateways and do operations locally as a normal node. However, this assumption is unrealistic for many light nodes, such as smart objects and IoT devices.

J. Summary of PCN Challenges and Discussion

Overall, designing PCNs that are efficient, secure, and provide support to heterogeneous devices is still an ongoing effort. Nonetheless, some areas seem more advanced than others. In the payment routing field, several works propose alternative protocols that focus primarily on multipath routing to improve payment goodput [66]–[68], [80], [90], [96], [116], [218]. The open challenges in this area include minimizing payment latency, analyzing the efficiency of routing protocols, and supporting multi-metric routing. Channel rebalancing also receives a fair amount of attention from researchers, with active local rebalancing (i.e., rebalancing where a node sends payments to itself to balance its channels) being the most explored technique [60], [61], [94], [115], [128]–[130]. Despite the efforts, it is still unclear how rebalancing methods affect the network as a whole and how much they would cost to users in the long run. Finally, extensive analysis shows that the Lightning Network, the most popular PCN, is centralized [87], [88], [131]. Besides facilitating topological attacks, such centralization can lead to long-term vulnerabilities [79]. Current implementations of the Lightning Network create channels disregarding centralization.

The main underexplored challenges of PCNs lie in security and privacy. In particular, the privacy of channel balances can easily be compromised with cheap channel probing techniques that exploit error messages [86], [107], [108], [136]. Amount jamming and slot jamming, two major denial of service attacks that exhaust channels, also have no systematic solution so far [86], [132]. Works that include resource-constrained devices into the PCN topology must deal with new vulnerabilities, such as untrusted gateways and coin theft [113], [127]. Besides security and privacy, other areas that need more extensive studies include congestion control, payment concurrency, load balancing, and payment scheduling policies.

VII. Scalability in Layer Two: Rollups

Besides payment channel networks, another layer-two solution that emerged from academia and business in recent years is transaction rollups, or simply rollups. Rollups were first proposed as a solution to scale blockchain systems that aggregate and compute transactions off-chain to reduce congestion and transaction overload on layer one. The name comes from the idea that transaction data are *rolled up* in batches before being published in the blockchain [137]. Some works consider rollups as hybrid layer-one/layer-two protocols as they publish information on every single transaction on the blockchain, unlike other pure layer-two protocols that summarize multiple off-chain operations in a single on-chain transaction [160]. Although rollups have been successfully deployed in the Ethereum ecosystem, their implementation is more complex than payment channel networks, as they require deploying smart contracts on a root-of-trust blockchain. Thus, rollups are considered too complex to implement in blockchain systems that do not support Turing-complete smart contracts, such as Bitcoin and its forks [138], [246].

Similarly to payment channel networks, the blockchain is used as a root of trust to solve disputes and initialize the off-chain layer. Users interested in issuing transactions off-chain can allocate funds using the deployed smart contract. A smart contract on the blockchain stores the state root, which is the Merkle root of the current rollup state; for example, it could be the current users' account balances. To update the state of the rollup, an agent, usually called an aggregator, can "*roll-up*" a set of off-chain transactions, calculate its Merkle root, and send it to the smart contract, as depicted in Figure 20. Aggregators send rolled-up transactions to the smart contract in a highly compressed form to reach their goal of reducing congestion. This compressed batch must provide enough information to allow users to compute the state update but can dismiss a significant volume of data, such as 20-byte-long addresses and some signatures. The smart contract verifies the new Merkle root and updates the rollup state.

As the compressed batch misses essential information and anyone can submit a batch, it is necessary to implement methods to attest to the correctness of the published batch. Currently, we can classify rollups according to their new-state verification method: optimistic or zero-knowledge.

A. Optimistic Rollups

In optimistic rollups, an aggregator batches transactions and sends a summary to the on-chain smart contract without providing proof of validity for the new state. Instead, every new state is *optimistically* considered valid, and, as in PCNs, a dispute period begins in the root blockchain. During this dispute period, validators can provide proof that the new state is invalid or incorrectly computed and challenge the update. The smart contract on layer-one can easily verify this proof, and the dishonest party can be punished. Usually, both validators and aggregators stake up part of their funds in a bond, and if any agents act maliciously, the attacker loses its coins at stake [137]. Similarly to PCNs, once the dispute period is finished, the state root is considered valid and cannot be changed or challenged by the validators.

1) Optimism: Optimism is the first optimistic rollup for Ethereum [140]. In Optimism, a single party called *sequencer* acts as a rollup aggregator by managing layer-two block production, transaction confirmations, state updates, and interaction with layer-one. Verifiers monitor the blockchain to challenge fraudulent state updates. If a verifier challenges a state update, the smart contract executes the challenged batch of transactions, starting from the last unchallenged update until the challenged state. This approach presents guarantees that the smart contract will find and correct the fraudulent update but it also requires a step-by-step execution of each instruction, which is inefficient. Furthermore, the sequencer must send enough data to layer-one in each state update. Otherwise, the smart contract cannot verify the state update or challenge. This approach results in high-costs given that smart contracts in Ethereum usually charge by instruction execution and data storage.

2) Arbitrum: Kalodner et al. [73] propose Arbitrum, an optimistic rollup system for the Ethereum blockchain. Like Ethereum, Arbitrum introduces a virtual machine, the Arbitrum Virtual Machine (AVM), that allows users to write codes and deploy applications. Agents called *managers* monitor the progress of the virtual machine and receive incentives to ensure its correct behavior by agreeing on the state updates. Unlike Optimism, Arbitrum does not require a step-by-step execution of the whole transaction batch in case of a dispute. Instead, Arbitrum opts for an off-chain bisection protocol between the disagreeing parties when a dispute occurs. In this bisection protocol, the parties interact with each other to find the disagreement in the state update using a process similar to a binary search. The goal is to reduce the disagreement to a single VM instruction that the validators can easily verify. This procedure significantly reduces the amount of data sent to layer-one, which makes Arbitrum's execution cheaper. In this bisection protocol, the managers must deposit funds in a bond, which they may lose as punishment for a malicious act. If proved correct, the challenger receives half of a manager's deposits and his/her coins back.

3) Cartesi: Similarly, Teixeira and Nehab [74] present the Cartesi machine, a reproducible virtual machine that runs on top of the Ethereum blockchain. The main difference between Cartesi and other optimistic rollups is that Cartesi machines are based on the RISC-V architecture and run a Linux-based system allowing the deployment of multiple applications in general programming languages as long as they are deterministic. Thus, while Optimism and Arbitrum introduce a virtual machine that solely runs programs compatible with the Ethereum Virtual Machine, Cartesi builds its virtual machine following the RISC-V architecture, broadening application possibilities. Cartesi nodes interact with smart contracts implemented in the root chain to update states that other participants can dispute. Like Arbitrum, when nodes challenge a result, they trigger a partition contract that starts a binary search to find the instruction that caused the disagreement. Once the instruction is found, the challenger

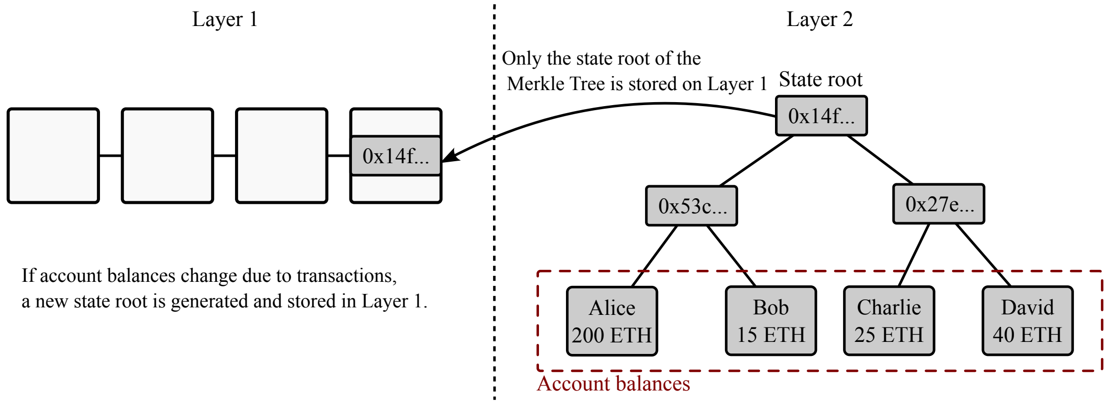
Layer 1
Layer 2
Only the state root of the Merkle Tree is stored on Layer 1
State root
0x14f...
0x14f...
0x53c...
0x27e...
Alice 200 ETH
Bob 15 ETH
Charlie 25 ETH
David 40 ETH
Account balances
If account balances change due to transactions, a new state root is generated and stored in Layer 1.

Figure 20. Instead of sending every transaction to the blockchain, rollups store account balances on a Merkle tree and update transactions off-chain. When an account balance changes, the state root is modified deterministically and can be verified by participants.

provides machine logs to the smart contract that emulates the computation and checks if the challenger is correct. Although Cartesi rollups are promising, they are still under development and lack comparative evaluation with other optimistic rollups.

B. Zero-knowledge Rollups

Unlike optimistic rollups, zero-knowledge (ZK) rollups require aggregators to provide a computational proof, called *validity proof*, to attest the proposed new state root. This validity proof is constructed using a cryptographic technique called zero-knowledge proof (ZKP), which allows users to prove a computation to other parties while keeping the inputs used to perform the computation private. Zero-knowledge systems introduce a setting where a prover (P) wants to prove a statement to a probabilistic polynomial-time verifier (V) while keeping information about the statement private [110]. These proof systems have the following properties [248]:

1) **Completeness**: If the statement is true, an honest verifier V will be convinced of the statement by an honest prover P.
2) **Soundness**: If the statement is false, a malicious prover P is unable to convince an honest verifier V that the statement is true, except for a small probability.
3) **Zero-knowledge**: The verifier V obtains no additional information about the statement other than the truth of the statement. As an example, at the end of the procedure, if the statement is true, the verifier V learns nothing about the statement other than it is true.

Zero-knowledge rollups leverage these properties to provide validity proof for the “rolled-up” batch of transactions. A layer-one smart contract automatically verifies if the update is valid by checking the correctness of the provided mathematical proof. Therefore, zero-knowledge rollups eliminate the requirement for a dispute window and achieve faster transaction finality.

There are two leading technologies used to compute zero-knowledge proofs in blockchain-based systems: Zero-Knowledge Succinct Non-interactive ARgument of Knowledge (ZK-SNARK) [249] and Zero-Knowledge Scalable Transparent ARgument of Knowledge (ZK-STARK) [75]. ZK-SNARKs are non-interactive, provide small proofs and allow users to quickly verify the computation without exchanging messages. Current ZK-SNARKs implementations, however, demand a trusted setup between participants to prevent an attacker from creating false statements. A compromised trusted setup allows malicious parties to provide proof for false statements while remaining undetected to honest parties [110]. On the other hand, ZK-STARKs are transparent, meaning that they don’t rely on any trusted party. Nevertheless, ZK-STARK schemes compile long proofs that require longer processing times compared to ZK-SNARKs.

1) zkSync: zkSync [138] is a zk-rollup system that runs on top of Ethereum. zkSync’s zero-knowledge scheme is based on Plonk [139], a Succinct Non-interactive Oecumenical (Universal) ARgument of Knowledge (SNORK). SNORK schemes are similar to SNARKs with the difference that the trusted setup is universal and can be used for multiple applications. zkSync supports the Solidity programming language, allowing users to deploy their Ethereum Dapps with small or no changes to the rollup systems while paying less for fees and receiving higher throughput. In zkSync, users move funds from the Ethereum mainnet to a designated zkSync account using a smart contract. Then, users can freely issue transactions off-chain to a zkSync operator through an API. The operator will organize the transactions in blocks that are sent to the Ethereum smart contract with a zero-knowledge proof. The proof is sent to the blockchain in a single transaction and achieves finality when the transaction is accepted in a block.

2) StarkEx: StarkEx [246] presents an Ethereum rollup based on ZK-STARK proofs. Like zkSync, StarkEx maintains a contract on the Ethereum mainnet to verify submitted zero-knowledge proofs. However, unlike zkSync, StarkEx establishes a Shared Prover (SHARP), a proving service maintained by the StarkWare company that provides the required infrastructure to generate the ZK proofs. StarkWare applications receive user requests, batch off-chain transactions,

Table IX
COMPARISON BETWEEN THE EXISTING ROLLUP PROPOSALS IN THE LITERATURE.

Rollups	Type	Validity proof	Reported throughput (tps)	Challenge window	Main Feature	EVM compatibility
Optimism [140]	Optimistic	Fraud proof	$2,000$	7 days	First implementation of optimistic rollup	✓
Arbitrum [73]	Optimistic	Fraud proof	$4,500$	7 days	Minimizes on-chain fees by reducing disagreement to a single instruction	✓
Cartesi [74]	Optimistic	Fraud proof	-	-	Implements rollups on a RISC-V architecture allowing for general-purpose Linux applications	✓
zkSync [138]	Zero-knowledge	SNORK	$2,000$	No challenge	Provides small and non-interactive proofs	✓
StarkEx [246]	Zero-knowledge	STARK	$3,000$	No challenge	Eliminates the requirement of a trusted setup for validity proof generation	✗
Loopring [247]	Zero-knowledge	SNARK	$2,025$	No challenge	Deploys an order-ring structure to enable multiple order-matching	✗

and send the batch to the SHARP service. Then, SHARP processes a proof and sends it to the on-chain smart contract, which will deterministically attest the generated proof.

3) Loopring: Loopring [141], [247] is a zk-rollup protocol which runs a Decentralized Exchange (DEX) on top of Ethereum. Instead of creating traditional order books to match exchange orders, Loopring's protocol deploys an *order ring*, which enables the matching of multiple orders in a circular trade. Unlike traditional pair matching in order books, order rings can scale up to 16 orders. A match happens in an order ring if all orders in the ring execute at an exchange rate equal to or better than the original rate set by the user. Loopring employs a specific type of participant called ring miners who receive a fee incentive for creating the order rings. Loopring differs from general zero-knowledge rollups, e.g. zkSync, by focusing on asset exchange instead of general DeFi applications.

C. Summary of Rollups and Discussion

Rollups present a layer-two alternative to payment channel networks that aggregate transactions off-chain to reduce fee costs and enable faster transaction processing. There are two main types of rollups: optimistic and zero-knowledge. In optimistic rollups, users assume that the batch of transactions is legitimate unless a participant disputes its result publicly. In zero-knowledge rollups, the batch of transactions contains a mathematical proof that can be publicly verified, stating that the set of transactions is valid. As zero-knowledge rollups eliminate the requirement of a dispute window, it provides faster transaction finality when compared to optimistic rollups. Table IX compares current rollup proposals in the literature.

Although rollups have a huge potential to scale blockchains, there are still many open challenges. First, their implementation requires deploying a complex smart contract. This requirement prevents the implementation of rollups in blockchains with simple scripting logic, such as Bitcoin [3]. Second, a significant part of optimistic rollups is still highly centralized. As an example, the Optimism Foundation runs the sole transaction aggregator in Optimism [140]. Similarly, Arbitrum presents a transaction aggregator run by Offchain Labs [250]. Thus, current optimistic rollups prioritize scalability and security over decentralization in the blockchain trilemma. This centralization exposes the system to the same vulnerabilities as centralized systems, such as introducing a single point of failure. Finally, while zero-knowledge rollups should provide faster transaction finality in theory, creating and verifying zero-knowledge proofs is still slow. Designing more efficient zero-knowledge proofs is critical to enable the mass adoption of zk-rollups systems. Furthermore, optimistic rollups should also focus on decentralizing aggregators to create more secure solutions.

Compared to PCNs, rollups still have some major disadvantages. PCNs usually allow users to move from layer two to layer one in 24 hours in case of a dispute [251] while optimistic rollup withdrawal time reaches 7 days. Although zero-knowledge rollups eliminate this dispute window, as mentioned before, they still struggle to generate proofs efficiently. Finally, PCNs' throughput is only limited by network bandwidth and computational power while most rollups scale up to only thousands of transactions per second. Nonetheless, rollups present great potential for scaling applications other than issuing payments.

VIII. CONCLUSION

Our survey presented existing techniques to solve the blockchain scalability problem at different layers. We discussed the advantages, disadvantages, and requirements to implement each solution in current blockchain networks. To the best of our knowledge, this work was the first to focus on layer-two solutions in detail, presenting the main challenges of payment channel networks and rollups. We have identified several key findings, outlined below.

A. Lessons Learned

Layers HW, L0, and L1. Scalability enhancements for the Hardware Layer (HW), Layer 0 (L0), and Layer 1 (L1) often need significant structural modifications to blockchain systems which can be challenging to execute. For instance, adopting FPGAs [16], [18] or TEE-backed consensus [14], [15], [77] in layer HW to improve transaction throughput shows significant results but imply users have specific hardware. This restricts the adoption of such solutions to enterprise environments in which nodes are homogeneous and have the capacity to purchase and configure expensive equipment. Modifications in L0 attempt to optimize message formats and communication protocols to maximize the amount of useful data transferred among nodes [20]–[26], [28], [187]. The main issue of this approach is it provides limited throughput gain and often leads to compatibility issues between upgraded nodes and legacy nodes [189]. Lastly, L1 proposals that focus on improving consensus provide significant throughput gains at the expense of security. For instance, DAG-based solutions achieve high throughput by allowing multiple transactions to be validated concurrently, which can lead to double-spend attacks in the short and medium term [30], [32], [46]. Block sharding parallelizes block validation into shards, which may lead to attacks where malicious nodes control a shard. Cross-chain and side-chain solutions delegate the validation of transactions to a secondary consensus protocol with fewer nodes, effectively weakening the security of the system [29], [33]–[35], [43], [51]. Thus, these solutions present good potential for improving scalability if the corresponding security concerns are overcome.

Payment channel networks. Despite representing one of the most promising solutions to improve blockchain scalability, efficient, secure, and heterogeneity-supporting PCNs are still under development, with some areas more mature than others. The payment routing domain seems to concentrate most of the efforts from the community, particularly in the form of multipath protocol proposals that emphasize payment goodput [66]–[68], [80], [90], [96], [116], [218]. However, some challenges remain open in this subject, such as minimizing payment latency and performing multi-metric routing [116]. Channel rebalancing is another significant area of focus, with local rebalancing being a dominant technique for large nodes [94], [115], [129], [230]. Small nodes with few connections still lack a cost-efficient rebalancing solution. Lastly, the Lightning Network, a leading PCN, has been shown to have centralization issues, which poses potential long-term vulnerabilities and topological attack surfaces [79], [87], [131]. This centralization is often overlooked when creating channels in the current implementations.

The critical unaddressed challenges in PCNs revolve around security and privacy. Namely, the privacy of channel balances is at risk due to cheap probing mechanisms that leverage overly-informative error messages [66], [80], [95], [116]. Similar techniques can be used to discover private channels [252]. There are unsolved major denial-of-service attacks, such as amount and slot jamming [86], [132], that can disable channels consistently with low effort. Gateway-related vulnerabilities that arise from the integration of resource-limited devices into PCNs also need to be explored [93], [113], [125]. Thus, we conclude that these topics represent significant opportunities for research that should receive increased focus from the community in the immediate future. Given the growing relevance of PCNs, we anticipate that currently overlooked challenges, including congestion control, payment concurrency, load balancing, and payment scheduling protocols, will ascend in importance due to their great potential to influence the network's operational efficiency.

Rollups. Rollups present a novel and promising layer-two solution for public blockchains with Turing-complete smart contracts that, unlike PCNs, are not restricted to issuing payments. Nevertheless, their nascent stage poses additional challenges. First, the current implementation of rollups demands the deployment of sophisticated smart contracts, which precludes their use in simpler blockchains like Bitcoin. Second, dispute-resolution mechanisms in optimistic rollups are inefficient, causing large delays when a dispute occurs. A notable portion of optimistic rollups circumvents the problem by centralizing transaction aggregation in a trusted entity, which leads to several centralization-related attack surfaces [73], [74], [140]. Third, zero-knowledge rollups, which aim to eliminate the need for disputes, suffer from the current inefficiency of zero-knowledge proofs in general [138], [246], [247]. Therefore, improvements in zero-knowledge proof efficiency and decentralizing aggregators are crucial to foster the widespread adoption of rollups. The literature on this topic is currently scarce and should be developed in the next years. Finally, when contrasted with PCNs, rollups have longer withdrawal times but are not bound by network-related limitations such as bandwidth and pathfinding delays, indicating they will likely serve different purposes.

In light of all the aforementioned takeaways, we conclude that, notwithstanding the inherent challenges they present, layer-two solutions emerge as the most viable technology to enhance the scalability of public blockchains. As layer-two solutions run off-chain, these solutions remove the requirement for a consensus protocol over every transaction, effectively reducing transaction confirmation delays and increasing throughput. Furthermore, they maintain the same level of payment security as guaranteed in other blockchain layers. Our examination encompassed payment channel networks and rollups, both of which are prevalent layer-two solutions in public blockchains. Presently, payment channel networks exhibit a more advanced stage of development compared to rollups, with the Lightning Network standing as the most mature example of a large-scale PCN. Nonetheless, our projection suggests a symbiotic coexistence of PCNs and rollups within the blockchain ecosystem; specifically, PCNs are optimally suited for swift financial transactions, while rollups cater to multifaceted, general-purpose applications.

IX. Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This paper was also funded by CNPq, CAPES, FAPERJ and FAPESP (2018/23292-0, 2015/24494-8, 2015/24514-9, 2015/24485-9, and 2014/50937-1).

References

[1] Blockchain.com, "Blockchain charts," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://www.blockchain.com/charts

[2] World Bank, "GDP (current US$)," 2022, Last access: Nov. 21th 2023.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[4] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," 2014. [Online]. Available: http://bitcoinaffiliatelist.com/wp-content/uploads/ethereum.pdf

[5] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[6] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, and Y. Liu, "A survey of blockchain technology applied to smart cities: Research issues and challenges," *Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2794–2830, 2019.

[7] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[8] L. A. C. de Souza *et al.*, "DFedForest: Decentralized Federated Forest," in *International Conference on Blockchain (Blockchain)*. IEEE, 2020, pp. 90–97.

[9] D. C. Nguyen, M. Ding, P. N. Pathirana, and A. Seneviratne, "Blockchain and ai-based solutions to combat coronavirus (covid-19)-like epidemics: A survey," *Ieee Access*, vol. 9, pp. 95 730–95 753, 2021.

[10] D. Marbouh, T. Abbasi, F. Maasmi, I. A. Omar, M. S. Debe, K. Salah, R. Jayaraman, and S. Ellahham, "Blockchain for covid-19: review, opportunities, and a trusted tracking system," *Arabian Journal for Science and Engineering*, vol. 45, pp. 9895–9911, 2020.

[11] Visa Inc., "Visa annual report," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://s29.q4cdn.com/385744025/files/doc_downloads/2022/Visa-Inc-Fiscal-2022-Annual-Report.pdf

[12] V. Buterin, "Why sharding is great: demystifying the technical properties," 2021, Last access: Nov. 21th 2023. [Online]. Available: https://vitalik.ca/general/2021/04/07/sharding.html

[13] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "SoK: Layer-two Blockchain Protocols," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2020, pp. 201–226.

[14] V. Costan and S. Devadas, "Intel sgx explained," Cryptology ePrint Archive, Report 2016/086, 2016, https://ia.cr/2016/086.

[15] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. Pietzuch, and E. G. Sirer, "Teechain: Scalable Blockchain Payments using Trusted Execution Environments," *arXiv preprint arXiv:1707.05454*, 2017.

[16] Y. Sakakibara, S. Morishima, K. Nakamura, and H. Matsutani, "A Hardware-based Caching System on FPGA NIC for Blockchain," *Transactions on Information and Systems*, vol. 101, no. 5, pp. 1350–1360, 2018.

[17] B. Ampel, M. Patton, and H. Chen, "Performance modeling of hyperledger sawtooth blockchain," in *International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2019, pp. 59–61.

[18] H. Javaid, J. Yang, N. Santoso, M. Upadhyay, S. Mohan, C. Hu, and G. Brebner, "Blockchain Machine: A Network-Attached Hardware Accelerator for Hyperledger Fabric," *arXiv preprint arXiv:2104.06968*, 2021.

[19] The Hyperledger Foundation, "Hyperledger sawtooth," Available at https://sawtooth.hyperledger.org/, 2022, Last access: Nov. 21th 2023.

[20] M. Corallo, "High-speed Bitcoin Relay Network," 2013.

[21] E. Lombrozo, J. Lau, and P. Wuille, "BIP 141: Segregated Witness (Consensus Layer)," Available at https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.b_stds:defact:bitcoin:bips:bip_0141, 2015, Last access: Nov. 21th 2023.

[22] M. Corallo, "BIP 152: compact block relay," 2016, Last access: Nov. 21th 2023. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0152

[23] S. Rüsch, I. Messadi, and R. Kapitza, "Towards Low-Latency Byzantine Agreement Protocols Using RDMA," in *International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2018, pp. 146–151.

[24] D. Ding, X. Jiang, J. Wang, H. Wang, X. Zhang, and Y. Sun, "Txilm: Lossy block compression with salted short hashing," *arXiv preprint arXiv:1906.06500*, 2019.

[25] K. Otsuki, Y. Aoki, R. Banno, and K. Shudo, "Effects of a Simple Relay Network on the Bitcoin Network," in *Proceedings of the Asian Internet Engineering Conference*, 2019, pp. 41–46.

[26] B. Huang, L. Jin, Z. Lu, X. Zhou, J. Wu, Q. Tang, and P. C. Hung, "BoR: Toward High-Performance Permissioned Blockchain in RDMA-enabled Network," *Transactions on Services Computing (TSC)*, vol. 13, no. 2, pp. 301–313, 2019.

[27] K. A. Cheow, "Something on Transaction Structure," 2020, Last access: Nov. 21th 2023. [Online]. Available: https://medium.com/@ackhor/something-on-transaction-structure-1ef60f719f01

[28] M. Xu, S. Liu, D. Yu, X. Cheng, S. Guo, and J. Yu, "CloudChain: a Cloud Blockchain Using Shared Memory Consensus and RDMA," *Transactions on Computers*, 2022.

[29] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling Blockchain Innovations with Pegged Sidechains," *URL: http://www. open-sciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, 2014.

[30] A. Churyumov, "A decentralized system for storage and transfer of value," 2016, "https://obyte.org/Byteball.pdf".

[31] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A Secure Sharding Protocol for Open Blockchains," in *Conference on Computer and Communications Security (SIGSAC)*. ACM, 2016, pp. 17–30.

[32] S. Popov, "The Tangle," *cit. on*, p. 131, 2017, https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf. Last access: Nov. 21th 2023.

[33] J. Poon and V. Buterin, "Plasma: Scalable Autonomous Smart Contracts," *White paper*, pp. 1–47, 2017.

[34] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame, "Towards Scalable and Private Industrial Blockchains," in *Workshop on Blockchain, Cryptocurrencies and Contracts (BCC)*. ACM, 2017, pp. 9–14.

[35] S. Ellis, A. Juels, and S. Nazarov, "Chainlink: A Decentralized Oracle Network," *Retrieved March*, vol. 11, p. 38, 2017, Last access: Nov. 21th 2023. [Online]. Available: https://research.chain.link/whitepaper-v1.pdf?_ga=2.22993531.1352052829.1651659724-1756188614.1651659724

[36] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A Secure, Scale-out, Decentralized Ledger via Sharding," in *Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.

[37] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Conference on Computer and Communications Security (SIGSAC)*. ACM, 2018, pp. 931–948.

[38] Z. Team and P. Barrett, "The Zilliqa Project: A Secure, Scalable Blockchain Platform," *Zilliqa*, pp. 1–18, 2018.

[39] L. Zhao and J. Yu, "Evaluating DAG-based blockchains for IoT," in *International Conference On Trust, Security And Privacy In Computing And Communications / International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 507–513.

[40] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards Scaling Blockchain Systems via Sharding," in *International Conference on Management of Data (SIGMOD)*. ACM, 2019, pp. 123–140.

[41] A. Bugday, A. Ozsoy, and H. Sever, "Securing Blockchain Shards by Using Learning Based Reputation and Verifiable Random Functions," in *International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2019, pp. 1–4.

[42] J. Wang and H. Wang, "Monoxide: Scale out Blockchains with Asynchronous Consensus Zones," in *Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2019, pp. 95–112.

[43] J. Kwon and E. Buchman, "Cosmos Whitepaper," *A Network of Distributed Ledgers*, 2019.

[44] A. Gopalan, A. Sankararaman, A. Walid, and S. Vishwanath, "Stability and Scalability of Blockchain Systems," *Measurement and Analysis of Computing Systems (POMACS)*, vol. 4, no. 2, pp. 1–35, 2020.

[45] S. Popov, H. Moog, D. Camargo, A. Capossele, V. Dimitrov, A. Gal, A. Greve, B. Kusmierz, S. Mueller, A. Penzkofer *et al.*, "The coordicide," 2020, Last access: Nov. 21th 2023. [Online]. Available: https://files.iota.org/papers/20200120_Coordicide_WP.pdf

[46] L. Baird and A. Luykx, "The hashgraph protocol: Efficient asynchronous BFT for high-throughput distributed ledgers," in *International Conference on Omni-layer Intelligent Systems (COINS)*, 2020, pp. 1–7.

[47] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo, "Sidechain Technologies in Blockchain Networks: An

Examination and State-of-the-Art Review," *Journal of Network and Computer Applications (JNCA)*, vol. 149, p. 102471, 2020.

[48] L. Lys, A. Micoulet, and M. Potop-Butucaru, "Atomic cross chain swaps via relays and adapters," in *Cryptocurrencies and Blockchains for Distributed Systems (CryBlock)*. ACM, 2020, pp. 59–64.

[49] I. D. Alvarenga, G. F. Camilo, L. A. De Souza, and O. C. M. Duarte, "DAGSec: A Hybrid Distributed Ledger Architecture for the Secure Management of the Internet of Things," in *International Conference on Blockchain (Blockchain)*. IEEE, 2021, pp. 266–271.

[50] Z. Hong, S. Guo, P. Li, and W. Chen, "Pyramid: A Layered Sharding Blockchain System," in *Internation Conference on Computer Communications (INFOCOM)*. IEEE, 2021, pp. 1–10.

[51] L. Breidenbach *et al.*, "Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks," p. 136, 2021, Last access: Nov. 21th 2023. [Online]. Available: https://research.chain.link/whitepaper-v2.pdf?_ga=2.69000657.1352052829.1651659724-1756188614.1651659724

[52] J. Spilman, "[Bitcoin-development] Anti DoS for tx Replacement," Available at https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html, 2013.

[53] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Stabilization, Safety, and Security of Distributed Systems*, A. Pelc and A. A. Schwarzmann, Eds. Cham: Springer International Publishing, 2015, pp. 3–18.

[54] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," 2016.

[55] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovksiy, and O. Osuntokun, "Flare: An Approach to Routing in Lightning Network," 2016, Last access: Nov. 21th 2023.

[56] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and Privacy with Payment-Channel Networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[57] A. Miller, I. Bentov, R. Kumaresan, C. Cordi, and P. McCorry, "Sprites and state channels: Payment networks that go faster than lightning," 2017.

[58] M. Green and I. Miers, "BOLT: Anonymous Payment Channels for Decentralized Currencies," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 473–489.

[59] E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub," in *Network and Distributed System Security Symposium (NDSS)*, 2017.

[60] "Splicing. [Lightning-dev] Channel top-up," Available at https://lists.linuxfoundation.org/pipermail/lightning-dev/2017-May/000696.html, 2017, Last access: Nov. 21th 2023.

[61] R. Khalil and A. Gervais, "Revive: Rebalancing Off-Blockchain Payment Networks," in *Conference on Computer and Communications Security (CCS)*. New York, NY, USA: ACM, 2017, p. 439–453. [Online]. Available: https://doi.org/10.1145/3133956.3134033

[62] E. Rohrer, J.-F. Laß, and F. Tschorsch, "Towards a Concurrent and Distributed Route Selection for Payment Channel Networks," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology (ESORICS)*, ser. Lecture Notes in Computer Science, J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, and J. Herrera-Joancomartí, Eds. Cham: Springer International Publishing, 2017, pp. 411–419.

[63] Trinity, "Trinity White Paper: Universal Off-chain Scaling Solution," 2018, Last access: Nov. 21th 2023. [Online]. Available: https://www.trinity.tech/#/writepaper

[64] D. Piatkivskyi and M. Nowostawski, "Split Payments in Payment Networks," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology (ESORICS)*, ser. Lecture Notes in Computer Science, J. Garcia-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios, Eds. Cham: Springer, 2018, pp. 67–75.

[65] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre, "Routing Payments on the Lightning Network," in *International Conference on Internet of Things (iThings) and Green Computing and Communications (GreenCom) and Cyber, Physical and Social Computing (CPSCom) and Smart Data (SmartData)*. IEEE, 2018, pp. 1161–1170.

[66] R. Yu, G. Xue, V. T. Kilari, D. Yang, and J. Tang, "CoinExpress: A Fast Payment Routing Mechanism in Blockchain-Based Payment Channel Networks," in *International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2018, pp. 1–9, iSSN: 1095-2055.

[67] O. Osuntokun, "[Lightning-dev] AMP: Atomic Multi-Path Payments over Lightning," Available at https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html, 2018, Last access: Nov. 21th 2023.

[68] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions," in *Proceedings of the 2018 Network and Distributed System Security Symposium*. Internet Society, 2018, arXiv: 1709.05748. [Online]. Available: http://arxiv.org/abs/1709.05748

[69] P. Hoenisch and I. Weber, "AODV–Based Routing for Payment Channel Networks," in *International Conference on Blockchain and Cryptocurrency (ICBC)*, ser. Lecture Notes in Computer Science, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham: Springer, 2018, pp. 107–124.

[70] E. Bergamini, P. Crescenzi, G. D'angelo, H. Meyerhenke, L. Severini, and Y. Velaj, "Improving the betweenness centrality of a node by adding links," *Journal of Experimental Algorithmics (JEA)*, vol. 23, 2018. [Online]. Available: https://doi.org/10.1145/3166071

[71] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability," Cryptology ePrint Archive, Report 2018/472, 2018, https://ia.cr/2018/472.

[72] S. Werman and A. Zohar, "Avoiding Deadlocks in Payment Channel Networks," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology (ESORICS)*, ser. Lecture Notes in Computer Science, J. Garcia-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios, Eds. Cham: Springer, 2018, pp. 175–187.

[73] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Security Symposium*. Baltimore, MD: USENIX, 2018, pp. 1353–1370. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner

[74] A. Teixeira and D. Nehab, "The core of cartesi," *Whitepaper, Cartesi*, 2018.

[75] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," Cryptology ePrint Archive, Paper 2018/046, 2018, https://eprint.iacr.org/2018/046. [Online]. Available: https://eprint.iacr.org/2018/046

[76] P. McCorry, S. Bakshi, I. Bentov, S. Meiklejohn, and A. Miller, "Pisa: Arbitration Outsourcing for State Channels," in *Conference on Advances in Financial Technologies (AFT)*. ACM, 2019, pp. 16–30.

[77] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: A Secure Payment Network with Asynchronous Blockchain Access," in *Symposium on Operating Systems Principles (SOSP)*. ACM, 2019, p. 63–79. [Online]. Available: https://doi.org/10.1145/3341301.3359627

[78] C. Hannon and D. Jin, "Bitcoin Payment-Channels for Resource Limited IoT Devices," in *International Conference on Omni-Layer Intelligent Systems (COINS)*. New York, NY, USA: ACM, 2019, pp. 50–57. [Online]. Available: https://doi.org/10.1145/3312614.3312629

[79] E. Rohrer, J. Malliaris, and F. Tschorsch, "Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks," in *European Symposium on Security and Privacy Workshops (EuroS PW)*. IEEE, 2019, pp. 347–356.

[80] P. Wang, H. Xu, X. Jin, and T. Wang, "Flash: Efficient Dynamic Routing for Off-chain Networks," in *International Conference on Emerging Networking Experiments And Technologies (CoNEXT)*. Orlando Florida: ACM, 2019, pp. 370–381. [Online]. Available: https://dl.acm.org/doi/10.1145/3359989.3365411

[81] Y. Zhang, D. Yang, and G. Xue, "CheaPay: An Optimal Algorithm for Fee Minimization in Blockchain-Based Payment Channel Networks," in *International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6, iSSN: 1938-1883.

[82] M. Conoscenti, A. Vetro, and J. C. De Martin, "Hubs, rebalancing and service providers in the lightning network," *Access*, vol. 7, pp. 132 828–132 840, 2019.

[83] R. Pickhardt, "lightning-network-autopilot," Available at https://github.com/renepickhardt/lightning-network-autopilot, 2019, Last access: Nov. 21th 2023.

[84] S. Tochner, S. Schmid, and A. Zohar, "Hijacking Routes in Payment Channel Networks: A Predictability Tradeoff," *arXiv preprint arXiv:1909.06890*, 2019.

[85] D. Robinson, "HTLCs Considered Harmful," in *Stanford Blockchain Conference (SBC)*, 2019, available at: https://www.youtube.com/watch?v=qUAyW4pdooA&ab_channel=CyberInitiative. Last access: Nov. 21th 2023.

[86] 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 *Asia Conference on Computer and Communications Security*. ACM, 2019, pp. 602–612.

[87] I. A. Seres, L. Gulyás, D. A. Nagy, and P. Burcsi, "Topological analysis of bitcoin's lightning network," in *Mathematical Research for Blockchain Economy*. Springer, 2020, pp. 1–12.

[88] J.-H. Lin, K. Primicerio, T. Squartini, C. Decker, and C. J. Tessone, "Lightning network: a second path towards centralisation of the bitcoin economy," *New Journal of Physics*, vol. 22, no. 8, p. 083022, 2020.

[89] brainbot labs Est., "The Raiden Network: Fast, cheap, scalable token transfers for Ethereum," 2020, available at: https://raiden.network/. Last access: Nov. 21th 2023. [Online]. Available: https://raiden.network/

[90] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High Throughput Cryptocurrency Routing in Payment Channel Networks," in *Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2020, pp. 777–796. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/sivaraman

[91] F. Beres, I. A. Seres, and A. A. Benczur, "A Cryptoeconomic Traffic Analysis of Bitcoin's Lightning Network," arXiv, Tech. Rep. arXiv:1911.09432, 2020, arXiv:1911.09432 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1911.09432

[92] J. Robert, S. Kubler, and S. Ghatpande, "Enhanced Lightning Network (off-chain)-based micropayment in IoT ecosystems," *Future Generation Computer Systems (FGCS)*, vol. 112, pp. 283–296, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19322654

[93] S. Mercan, E. Erdin, and K. Akkaya, "Improving Transaction Success Rate via Smart Gateway Selection in Cryptocurrency Payment Channel Networks," in *International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–3.

[94] R. Pickhardt and M. Nowostawski, "Imbalance measure and proactive channel rebalancing algorithm for the lightning network," in *International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–5.

[95] S. Mazumdar, S. Ruj, R. G. Singh, and A. Pal, "HushRelay: A Privacy-Preserving, Efficient, and Scalable Routing Algorithm for Off-Chain Payments," in *International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–5.

[96] C. Lin, N. Ma, X. Wang, and J. Chen, "Rapido: Scaling blockchain with multi-path payment channels," *Neurocomputing*, vol. 406, pp. 322–332, 2020.

[97] C. Grunspan, G. Lehéricy, and R. Pérez-Marco, "Ant Routing Scalability for the Lightning Network," 2020, arXiv:2002.01374 [cs]. [Online]. Available: http://arxiv.org/abs/2002.01374

[98] P. Li, T. Miyazaki, and W. Zhou, "Secure Balance Planning of Off-Blockchain Payment Channel Networks," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 1728–1737.

[99] Z. Avarikioti, L. Heimbach, Y. Wang, and R. Wattenhofer, "Ride the lightning: The game theory of payment channels," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2020, pp. 264–283.

[100] O. Ersoy, S. Roos, and Z. Erkin, "How to profit from payments channels," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2020, pp. 284–303.

[101] B. Teinturier, "Route Blinding," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://github.com/lightning/bolts/pull/765

[102] E. Rohrer and F. Tschorsch, "Counting Down Thunder: Timing Attacks on Privacy in Payment Channel Networks," in *Conference on Advances in Financial Technologies (AFT)*. New York, NY, USA: ACM, 2020, pp. 214–227. [Online]. Available: https://doi.org/10.1145/3419614.3423262

[103] W. Tang, W. Wang, G. Fanti, and S. Oh, "Privacy-Utility Tradeoffs in Routing Cryptocurrency over Payment Channel Networks," *Measurement and Analysis of Computing Systems*, vol. 4, no. 2, 2020. [Online]. Available: https://doi.org/10.1145/3392147

[104] J. Harris and A. Zohar, "Flood & Loot: A Systemic Attack on the Lightning Network," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT)*, 2020, pp. 202–213.

[105] S. Tochner, A. Zohar, and S. Schmid, "Route Hijacking and DoS in Off-chain Networks," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFC)*, 2020, pp. 228–240.

[106] V. Bagaria, J. Neu, and D. Tse, "Boomerang: Redundancy improves latency and throughput in payment-channel networks," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2020, pp. 304–324.

[107] G. v. Dam, R. A. Kadir, P. N. Nohuddin, and H. B. Zaman, "Improvements of the Balance Discovery Attack on Lightning Network Payment Channels," in *International Conference on ICT Systems Security and Privacy Protection (IFIP SEC)*. Springer, 2020, pp. 313–323.

[108] S. Tikhomirov, R. Pickhardt, A. Biryukov, and M. Nowostawski, "Probing channel balances in the lightning network," *arXiv preprint arXiv:2004.00333*, 2020.

[109] N. Papadis and L. Tassiulas, "Blockchain-Based Payment Channel Networks: Challenges and Recent Advances," *Access*, vol. 8, pp. 227 596–227 609, 2020.

[110] A. Kosba, D. Papadopoulos, C. Papamanthou, and D. Song, "MIRAGE: Succinct arguments for randomized algorithms with applications to universal zk-SNARKs," in *Conference on Security Symposium (SEC)*. USA: USENIX, 2020.

[111] M. Conoscenti, A. Vetrò, and J. C. De Martin, "CLoTH: A Lightning Network Simulator," *SoftwareX*, vol. 15, p. 100717, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711021000613

[112] G. Kappos, H. Yousaf, A. Piotrowska, S. Kanjalkar, S. Delgado-Segura, A. Miller, and S. Meiklejohn, "An Empirical Analysis of Privacy in the Lightning Network," in *Financial Cryptography and Data Security (FC)*, ser. Lecture Notes in Computer Science, N. Borisov and C. Diaz, Eds. Berlin, Heidelberg: Springer, 2021, pp. 167–186.

[113] A. Kurt, S. Mercan, O. Shlomovits, E. Erdin, and K. Akkaya, "LNGate: Powering IoT with Next Generation Lightning Micro-Payments using Threshold Cryptography," in *Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. New York, NY, USA: ACM, 2021, pp. 117–128. [Online]. Available: https://doi.org/10.1145/3448300.3467833

[114] ION Lightning Network Wiki, "Watchtowers," 2021, available at: https://wiki.ion.radar.tech/tech/research/watchtowers. Last access: Nov. 21th 2023.

[115] N. Awathare, Suraj, Akash, V. J. Ribeiro, and U. Bellur, "Rebal: Channel balancing for payment channel networks," in *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2021, pp. 1–8.

[116] R. Pickhardt and S. Richter, "Optimally Reliable & Cheap Payment Flows on the Lightning Network," *arXiv:2107.05322 [cs]*, 2021, arXiv: 2107.05322. [Online]. Available: http://arxiv.org/abs/2107.05322

[117] K. Lange, E. Rohrer, and F. Tschorsch, "On the impact of attachment strategies for payment channel networks," *arXiv preprint arXiv:2102.09256*, 2021.

[118] G. Avarikioti, R. Scheuner, and R. Wattenhofer, "Payment networks as creation games," 2019. [Online]. Available: https://arxiv.org/abs/1908.00436

[119] E. Erdin, S. Mercan, and K. Akkaya, "An Evaluation of Cryptocurrency Payment Channel Networks and Their Privacy Implications," 2021, arXiv:2102.02659 [cs]. [Online]. Available: http://arxiv.org/abs/2102.02659

[120] A. Mizrahi and A. Zohar, "Congestion Attacks in Payment Channel Networks," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2021, pp. 170–188.

[121] S. Rahimpour and M. Khabbazian, "Spear: Fast multi-path payment with redundancy," in *Conference on Advances in Financial Technologies (AFT)*. ACM, 2021, pp. 183–191.

[122] S. Geissler, F. Wamser, W. Bauer, M. Krolikowski, S. Gebert, and T. Hoßfeld, "Signaling Traffic in Internet-of-Things Mobile Net-

works," in *International Symposium on Integrated Network Management (IM)*. IFIP/IEEE, 2021, pp. 452–458, iSSN: 1573-0077.

[123] G. A. F. Rebello, G. F. Camilo, M. Potop-Butucaru, M. E. M. Campista, M. D. de Amorim, and L. H. M. K. Costa, "PCNsim: A Flexible and Modular Simulator for Payment Channel Networks," in *International Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–2.

[124] N. Papadis and L. Tassiulas, "Payment Channel Networks: Single-Hop Scheduling for Throughput Maximization," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2022, pp. 900–909, iSSN: 2641-9874.

[125] A. Kurt, K. Akkaya, S. Yilmaz, S. Mercan, O. Shlomovits, and E. Erdin, "LNGate^2: Secure Bidirectional IoT Micro-payments using Bitcoin's Lightning Network and Threshold Cryptography," 2022, arXiv:2206.02248 [cs]. [Online]. Available: http://arxiv.org/abs/2206.02248

[126] Bitcoin Design, "Lightning Services," Available at https://bitcoin.design/guide/how-it-works/lightning-services, 2022, Last access: Nov. 21th 2023.

[127] G. A. F. Rebello, M. Potop-Butucaru, M. D. de Amorim, and O. C. M. B. Duarte, "Securing Wireless Payment-Channel Networks With Minimum Lock Time Windows," in *International Conference on Communications (ICC)*. IEEE, 2022, pp. 2297–2302, iSSN: 1938-1883.

[128] Z. Avarikioti, K. Pietrzak, I. Salem, S. Schmid, S. Tiwari, and M. Yeo, "HIDE & SEEK: Privacy-Preserving Rebalancing on Payment Channel Networks," Cryptology ePrint Archive, Paper 2021/1401, 2021, https://eprint.iacr.org/2021/1401. [Online]. Available: https://eprint.iacr.org/2021/1401

[129] L. Labs, "Lightning Loop," 2022, Last access: Nov. 21th 2023.

[130] Z. Ge, Y. Zhang, Y. Long, and D. Gu, "Shaduf: Non-cycle Payment Channel Rebalancing," in *Network and Distributed Systems Security Symposium (NDSS)*. The Internet Society, 2022, pp. 1–18.

[131] G. F. Camilo, G. A. F. Rebello, L. A. Souza, M. Potop-Butucaru, M. D. Amorim, M. E. M. Campista, and L. H. M. K. Costa, "Topological evolution analysis of payment channels in the lightning network," in *Latin-American Conference on Communications (LATINCOM)*. IEEE, 2022.

[132] C. Shikhelman and S. Tikhomirov, "Unjamming Lightning: A Systematic Approach," *Cryptology ePrint Archive*, 2022.

[133] Z. Hong, S. Guo, R. Zhang, P. Li, Y. Zhan, and W. Chen, "Cycle: Sustainable Off-Chain Payment Channel Network with Asynchronous Rebalancing," in *International Conference on Dependable Systems and Networks (DSN)*. IEEE/IFIP, 2022, pp. 41–53.

[134] X. Wang, H. Gu, Z. Li, F. Zhou, R. Yu, and D. Yang, "Why Riding the Lightning? Equilibrium Analysis for Payment Hub Pricing," in *International Conference on Communications (ICC)*. IEEE, 2022, pp. 5409–5414.

[135] "lnd-autopilot," Available at https://github.com/lightningnetwork/lnd/tree/master/autopilot, 2022, Last access: Nov. 21th 2023.

[136] A. Biryukov, G. Naumenko, and S. Tikhomirov, "Analysis and probing of parallel channels in the lightning network," in *Financial Cryptography and Data Security (FC)*. Springer, 2022, pp. 337–357.

[137] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *Access*, vol. 10, pp. 93 039–93 054, 2022.

[138] "zkSync Basics," Available at https://v2-docs.zksync.io/dev/fundamentals/zkSync.html, 2022, Last access: Nov. 21th 2023.

[139] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," Cryptology ePrint Archive, Paper 2019/953, 2019, https://eprint.iacr.org/2019/953. [Online]. Available: https://eprint.iacr.org/2019/953

[140] "Protocol specs." [Online]. Available: https://community.optimism.io/docs/protocol/

[141] "protocols/packages/loopring_v3/DESIGN.md at master · Loopring/protocols." [Online]. Available: https://github.com/Loopring/protocols/blob/master/packages/loopring_v3/DESIGN.md

[142] 1ML, "Lightning Network Explorer," Available at https://1ml.com, 2022, Last access: Nov. 21th 2023.

[143] M. Jourenko, K. Kurazami, M. Larangeira, and K. Tanaka, "SoK: A Taxonomy for Layer-2 Scalability Related Protocols for Cryptocurrencies," 2019, report Number: 352. [Online]. Available: https://eprint.iacr.org/2019/352

[144] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Off the chain transactions," *IACR Cryptol. ePrint Arch.*, p. 360, 2019.

[145] A. Hafid, A. S. Hafid, and M. Samih, "Scaling Blockchains: A Comprehensive Survey," *Access*, vol. 8, pp. 125 244–125 262, 2020.

[146] H. Khojasteh and H. Tabatabaei, "A Survey and Taxonomy of Blockchain-based Payment Channel Networks," in *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2021, pp. 1–8, iSSN: 2643-1971.

[147] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 Blockchain Scaling: a Survey," 2021, arXiv:2107.10881 [cs]. [Online]. Available: http://arxiv.org/abs/2107.10881

[148] Z. Zhao, L. Zhou, and C. Su, "Systematic Research on Technology and Challenges of Lightning Network," in *Conference on Dependable and Secure Computing (DSC)*. IEEE, 2021, pp. 1–8.

[149] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804522001801

[150] G.-T. Nguyen and K. Kim, "A Survey about Consensus Algorithms Used in Blockchain," *Journal of Information Processing Systems (JIPS)*, vol. 14, no. 1, pp. 101–128, 2018, publisher: Korea Information Processing Society. [Online]. Available: https://koreascience.kr/article/JAKO201810256452304.page

[151] S. Kim, Y. Kwon, and S. Cho, "A Survey of Scalability Solutions on Blockchain," in *International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2018, pp. 1204–1207.

[152] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A Survey on the Scalability of Blockchain Systems," *Network*, vol. 33, no. 5, pp. 166–173, 2019.

[153] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," *Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[154] D. Yang, C. Long, H. Xu, and S. Peng, "A Review on Scalability of Blockchain," in *International Conference on Blockchain Technology (ICBCT)*. ACM, 2020, pp. 1–6.

[155] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in Blockchains," *Access*, vol. 8, pp. 14 155–14 181, 2020.

[156] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to Scalability of Blockchain: A Survey," *Access*, vol. 8, pp. 16 440–16 455, 2020.

[157] D. Khan, L. T. Jung, and M. A. Hashmani, "Systematic Literature Review of Challenges in Blockchain Scalability," *Applied Sciences*, vol. 11, no. 20, p. 9372, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/20/9372

[158] A. I. Sanka and R. C. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *Journal of Network and Computer Applications (JNCA)*, vol. 195, p. 103232, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521002307

[159] M. H. Nasir, J. Arshad, M. M. Khan, M. Fatima, K. Salah, and R. Jayaraman, "Scalable blockchains — A systematic review," *Future Generation Computer Systems (FGCS)*, vol. 126, pp. 136–162, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21002971

[160] Vitalik Buterin, "An Incomplete Guide to Rollups," Available at https://vitalik.ca/general/2021/01/05/rollup.html, 2021, Last access: Nov. 21th 2023.

[161] G. A. F. Rebello, G. F. Camilo, L. C. Guimarães, L. A. C. de Souza, G. A. Thomaz, and O. C. Duarte, "A security and performance analysis of proof-based consensus protocols," *Annals of Telecommunications*, pp. 1–21, 2021.

[162] J. Chen and S. Micali, "Algorand: A Secure and Efficient Distributed Ledger," *Theoretical Computer Science*, vol. 777, pp. 155–183, 2019.

[163] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "{Bitcoin-NG}: A scalable blockchain protocol," in *Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2016, pp. 45–59.

[164] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Symposium on Operating Systems Design and Implementation (OSDI)*. Berkeley, CA, USA: USENIX, 1999, pp. 173–186.

[165] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Symposium on Principles of Distributed Computing*. ACM, 2019, pp. 347–356.

[166] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain," in *Italian Conference on Cyber Security*, 2018. [Online]. Available: https://eprints.soton.ac.uk/415083/

[167] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols." in *Conference on Computer and Communications Security (CCS)*. ACM, 2016, pp. 31–42.

[168] B. Guo, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo: Faster asynchronous bft protocols," in *Conference on Computer and Communications Security (CCS)*. ACM, 2020, pp. 803–818.

[169] B. Guo, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Speeding dumbo: Pushing asynchronous bft closer to practice," *Cryptology ePrint Archive*, 2022.

[170] Y. Gao, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo-ng: Fast asynchronous bft consensus with throughput-oblivious latency," in *Conference on Computer and Communications Security (CCS)*. ACM, 2022, pp. 1187–1201.

[171] E. Androulaki *et al.*, "Hyperledger Fabric: a distributed operating system for permissioned blockchains," in *13th EuroSys Conference*, 2018, p. 30.

[172] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*. Springer, 2017, pp. 282–297.

[173] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, 2014, https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[174] D. Larimer, "EOS.IO White Paper," 2017, available at https://developers.eos.io/-welcome/latest/protocol/consensus_protocol. Last access: Nov. 21th 2023.

[175] Y. Amoussou-Guenou, A. Del Pozzo, M. Potop-Butucaru, and S. Tucci-Piergiovanni, "Dissecting tendermint," in *International Conference on Networked Systems*. Springer, 2019, pp. 166–182.

[176] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," Ph.D. dissertation, University of Guelph, 2016.

[177] F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong, and M. Zhou, "Delegated Proof of Stake with Downgrade: A Secure and Efficient Blockchain Consensus Algorithm with Downgrade Mechanism," *Access*, vol. 7, pp. 118 541–118 555, 2019.

[178] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "SoK: Sharding on Blockchain," in *Conference on Advances in Financial Technologies (AFT)*. ACM, 2019, pp. 41–61.

[179] Blockchain.com, "Blockchain Size," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://www.blockchain.com/explorer/charts/avg-block-size

[180] Visa, "Visa Acceptance for Retailers," Available at https://usa.visa.com/run-your-business/small-business-tools/retail.html, 2022, Last access: Nov. 21th 2023.

[181] LetsExchange, "What Is Block Confirmation on Ethereum and How Many Confirmations Are Required?" Available at https://letsexchange.io/blog/what-is-block-confirmation-on-ethereum-and-how-many-\\confirmations-are-required/, 2021, Last access: Nov. 21th 2023.

[182] C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," in *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2014, pp. 313–326.

[183] P. Wuille, J. Nick, and A. Towns, "Taproot: SegWit Version 1 Spending Rules," Available at https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki, 2020, Last access: Nov. 21th 2023.

[184] ——, "Validation of Taproot Scripts," Available at https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki, 2020, Last access: Nov. 21th 2023.

[185] C. P. Schnorr, "Method for identifying subscribers and for generating and verifying electronic signatures in a data exchange system," 1991, uS Patent 4,995,082.

[186] J. Lau, "Merkelized Abstract Syntax Tree," 2016, Last access: Nov. 21th 2023. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0114.mediawiki

[187] Cardano, "CARDANO: Making the World Work Better for All," Available at https://cardano.org/, 2022, Last access: Nov. 21th 2023.

[188] B. Cash, "Bitcoin cash," 2023, Last access: Nov. 21th 2023. [Online]. Available: https://bitcoincash.org/

[189] K. Peters, "A history of bitcoin hard forks," 2023, Last access: Nov. 21th 2023. [Online]. Available: https://www.investopedia.com/tech/history-bitcoin-hard-forks/

[190] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications*, vol. 61, no. 7, p. 95–102, 2018. [Online]. Available: https://doi.org/10.1145/3212998

[191] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *Computing Surveys*, 2023.

[192] Q. Wang, J. Yu, S. Chen, and Y. Xiang, "SoK: Diving into DAG-based blockchain systems," 2020, https://arxiv.org/abs/2012.06128v2.

[193] M. Conti, G. Kumar, P. Nerurkar, R. Saha, and L. Vigneri, "A survey on security challenges and solutions in the iota," *Journal of Network and Computer Applications (JNCA)*, p. 103383, 2022.

[194] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, "Scalable and probabilistic leaderless BFT consensus through metastability," *arXiv preprint arXiv:1906.08936*, 2019.

[195] I. Amores-Sesar, C. Cachin, and E. Tedeschi, "When Is Spring Coming? A Security Analysis of Avalanche Consensus," in *International Conference on Principles of Distributed Systems (OPODIS)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[196] J. C. Corbett *et al.*, "Spanner: Google's Globally Distributed Database," *Transactions on Computer Systems*, vol. 31, no. 3, 2013. [Online]. Available: https://doi.org/10.1145/2491245

[197] 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 *Internation Conference on Computer Communications (INFOCOM)*. IEEE, 2022.

[198] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable Bias-Resistant Distributed Randomness," in *Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 444–460.

[199] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *Transactions on Programming Languages and System*, 1982.

[200] F. Vogelsteller and V. Buterin, "Eip-20: Token standard," 2015, Last access: Nov. 21th 2023. [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

[201] M. Hearn, "[ANNOUNCE] Micro-payment channels implementation now in bitcoinj," Bitcoin Forum. Available at https://bitcointalk.org/index.php?topic=244656.0, 2013.

[202] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Conference on Computer and Communications Security (CCS)*. ACM, 2018, p. 949–966. [Online]. Available: https://doi.org/10.1145/3243734.3243856

[203] Bitcoinj.org, "bitcoinj," Available at https://bitcoinj.org/., 2022.

[204] BitcoinWiki, "Script - Bitcoin Wiki," Available at https://en.bitcoin.it/wiki/Script, 2021.

[205] Bitcoin Wiki, "Hash Time Locked Contracts," Available at https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts, 2021, Last access: Nov. 21th 2023.

[206] Lightning Network Developers, "Lightning App Directory," Available at https://dev.lightning.community/lapps/, 2022.

[207] CloudTweaks, "How Bitcoin Brought The Lightning Network To El Salvador," https://cloudtweaks.com/2021/07/how-bitcoin-brought-lightning-network-el-salvador/, 2021, Last access: Nov. 21th 2023.

[208] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation onion router," in *Security Symposium (Security)*. USENIX, 2004. [Online]. Available: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[209] A. Towns *et al.*, "BOLT #0: Introduction and index," https://github.com/lightning/bolts/blob/master/00-introduction.md, 2022, Last access: Nov. 21th 2023.

[210] R. Russell *et al.*, "BOLT #3: Bitcoin transaction and script formats," https://github.com/lightning/bolts/blob/master/03-transactions.md#fees, 2022, Last access: Nov. 21th 2023.

[211] P. Zabka, K.-T. Förster, S. Schmid, and C. Decker, "Node classification and geographical analysis of the lightning cryptocurrency network," in *International Conference on Distributed Computing and Networking (ICDCN)*. New York, NY, USA: ACM, 2021, p. 126–135. [Online]. Available: https://doi.org/10.1145/3427796.3427837

[212] R. Russell *et al.*, "BOLT #2: Peer protocol for channel management," https://github.com/lightningnetwork/lightning-rfc/blob/master/02-peer-protocol.md, 2022, Last access: Nov. 21th 2023.

[213] IntoTheBlock, "Raiden Network Statistics," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://app.intotheblock.com/coin/RDN/deep-dive?group=network&chart=all

[214] Messari, "Raiden Network Market Data," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://messari.io/asset/raiden-network/metrics/all

[215] brainbot labs Est., "Raiden explorer," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://explorer.raiden.network/tokens

[216] Trinity, "Trinity Project," 2018, Last access: Nov. 21th 2023. [Online]. Available: https://github.com/trinity-project/trinity

[217] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. Fanti, and P. Viswanath, "Routing Cryptocurrency with the Spider Network," in *Proceedings of the ACM Workshop on Hot Topics in Networks*, 2018, pp. 29–35.

[218] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "SilentWhispers: Enforcing Security and Privacy in Decentralized Credit Networks," in *Network and Distributed System Security Symposium (NDSS)*. San Diego, CA: Internet Society, 2017. [Online]. Available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/silentwhispers-enforcing-security-and-privacy-decentralized-credit-networks/

[219] J. Edmonds and K. RM Richard, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *Journal of the ACM (JACM)*, vol. 19, no. 2, pp. 248–264, 1972.

[220] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and outlook," in *Trust and Trustworthy Computing*, M. Conti, M. Schunter, and I. Askoxylakis, Eds. Cham: Springer, 2015, pp. 163–180.

[221] R. Pickhardt, "[Lightning-dev] Code for sub second runtime of piecewise linarization to quickly approximate the minimum convex cost flow problem," Available at https://lists.linuxfoundation.org/pipermail/lightning-dev/2022-March/003510.html, 2022, Last access: Nov. 21th 2023.

[222] L. R. Ford and D. R. Fulkerson, "Maximal Flow Through a Network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956, publisher: Cambridge University Press. [Online]. Available: https://www.cambridge.org/core/journals/canadian-journal-of-mathematics/article/maximal-flow-through-a-network/5D6E55D3B06C4F7B1043BC1D82D40764

[223] A. V. Goldberg, "The Partial Augment–Relabel Algorithm for the Maximum Flow Problem," in *Algorithms - ESA*, D. Halperin and K. Mehlhorn, Eds. Berlin, Heidelberg: Springer, 2008, pp. 466–477.

[224] Y. Zhang and D. Yang, "RobustPay+: Robust Payment Routing With Approximation Guarantee in Blockchain-Based Payment Channel Networks," *Transactions on Networking (TON)*, vol. 29, no. 4, pp. 1676–1686, 2021.

[225] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics (Q Appl Math)*, vol. 16, no. 1, pp. 87–90, 1958. [Online]. Available: https://www.ams.org/qam/1958-16-01/S0033-569X-1958-0102435-2/

[226] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230140209. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230140209

[227] R. Pickhardt, "Price of anarchy from selfish routing strategies on the lightning," 2022, Last access: Nov. 21th 2023. [Online]. Available: https://www.bitcoininsider.org/article/167640/price-anarchy-selfish-routing-strategies-lightning-network-rene-pickhardt

[228] B. Teinturier, "Trampoline Routing," 2021, Last access: Nov. 21th 2023. [Online]. Available: https://github.com/lightning/bolts/pull/829

[229] A. Bosworth, "Balance of Satoshis," 2021, available at: https://github.com/alexbosworth/balanceofsatoshis. Last access: Nov. 21th 2023. [Online]. Available: https://github.com/alexbosworth/balanceofsatoshis

[230] C. Otto, "Rebalance-LND," 2022, available at: https://github.com/C-Otto/rebalance-lnd. Last access: Nov. 21th 2023. [Online]. Available: https://github.com/C-Otto/rebalance-lnd

[231] N. Papadis and L. Tassiulas, "Deep reinforcement learning-based rebalancing policies for profit maximization of relay nodes in payment channel networks," Cryptology ePrint Archive, Paper 2022/1385, 2022, https://eprint.iacr.org/2022/1385. [Online]. Available: https://eprint.iacr.org/2022/1385

[232] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, pp. 47–97, 2002. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.74.47

[233] P. Erdős, A. Rényi *et al.*, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[234] R. Russell *et al.*, "BOLT #7: P2p node and channel discovery," https://github.com/lightningnetwork/lightning-rfc/blob/master/https://github.com/lightning/bolts/blob/master/07-routing-gossip.md, 2022, Last access: Nov. 21th 2023.

[235] 1ML, "Lightning Network Explorer: Bitfinex Lightning Node," Available at https://1ml.com/node/033d8656219478701227199cbd6f670335c8d408a92ae88b962c49d4dc0e83e025, 2022, Last access: Nov. 21th 2023.

[236] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Symposium on Theory of Computing (STOC)*. New York, NY, USA: ACM, 1985, p. 291–304. [Online]. Available: https://doi.org/10.1145/22145.22178

[237] G. F. Camilo, G. A. F. Rebello, L. A. C. de Souza, and O. C. M. Duarte, "A Secure Personal-Data Trading System Based on Blockchain, Trust, and Reputation," in *International Conference on Blockchain (Blockchain)*. IEEE, 2020, pp. 379–384.

[238] S. Mazumdar, P. Banerjee, and S. Ruj, "Time is Money: Countering Griefing Attack in Lightning Network," in *International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 1036–1043.

[239] J. Bier, "Preventing Channel Jamming," 2021, available at: https://blog.bitmex.com/preventing-channel-jamming/. Last access: Nov. 21th 2023. [Online]. Available: https://blog.bitmex.com/preventing-channel-jamming/

[240] M. Al-Shurman, S.-M. Yoo, and S. Park, "Black Hole Attack in Mobile Ad Hoc Networks," in *Proceedings of the 42nd annual Southeast regional conference (ACM-SE*, 2004, pp. 96–97.

[241] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," Internet Engineering Task Force, Request for Comments RFC 8684, 2020, num Pages: 68. [Online]. Available: https://datatracker.ietf.org/doc/rfc8684

[242] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998. [Online]. Available: https://www.nature.com/articles/30918

[243] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.286.5439.509

[244] "Credit Card Fraud Detection." [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[245] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer, 2010, pp. 15–34. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_2

[246] "StarkEx Documentation," Available at https://docs.starkware.co/starkex/index.html, Last access: Nov. 21th 2023.

[247] D. Wang, J. Zhou, M. Finestone, and A. Wang, "Loopring: A Decentralized Token Exchange Protocol," 2018.

[248] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994. [Online]. Available: http://link.springer.com/10.1007/BF00195207

[249] 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 (ITCS)*. ACM, 2012, p. 326–349. [Online]. Available: https://doi.org/10.1145/2090236.2090263

[250] "The Sequencer and Censorship Resistance | Arbitrum Docs," 2023. [Online]. Available: https://developer.arbitrum.io/sequencer

[251] A. Poinsot, “Answer to ”How do I set the our_to_self_delay parameter?”,” 2022. [Online]. Available: https://bitcoin.stackexchange.com/a/116433

[252] T. Giorgio, “Preliminary Hidden Lightning Network Analysis,” Available at https://lists.linuxfoundation.org/pipermail/lightning-dev/2022-June/003599.html, 2022, Last access: Nov. 21th 2023.