Blockchain-Based Payment Channel Networks: Challenges and Recent Advances

NIKOLAOS PAPADIS (Member, IEEE), AND LEANDROS TASSIULAS, (Fellow, IEEE)

Department of Electrical Engineering, Yale University, New Haven, CT 06511, USA
Yale Institute for Network Science, Yale University, New Haven, CT 06511, USA

Corresponding author: Nikolaos Papadis (nikolaos.papadis@yale.edu)

This work was supported in part by the National Science Foundation (NSF) under Grant 1815676, in part by the Army Research Office (ARO) under Grant W911NF1810378, in part by a grant from the JP Morgan Chase, and in part by the Onassis Foundation under Grant F ZP 041-1/2019-2020.

ABSTRACT Blockchain technology has been developed with the vision to enable trusted collaboration between untrusted parties, without the need for a central authority. Despite its many promising applications, the technology suffers from a scalability problem. In order to increase transaction throughput and decrease transaction confirmation latency, payment channel networks have been proposed. Payment channel networks introduce a layer on top of the main chain, in which transactions can happen in a safe manner between only the transacting parties without burdening the entire network. In this article, we present and highlight the many interesting research aspects this new type of network introduces. We first provide background on the mechanics of the operation of payment channel networks, and then proceed to present a plethora of research problems of networking and/or economics flavor arising in this context, including routing, scheduling, rebalancing, network design and topology analysis, and fee optimization. This work is within the scope of both the networking and network economics communities.

INDEX TERMS Blockchain, complex networks, cryptocurrencies, cryptoeconomics, networking, payment channel networks (PCNs), peer to peer computing, routing.

I. INTRODUCTION

Blockchain technology has emerged in the recent years as an enabler of trusted collaboration among untrusted parties. The application that introduced blockchain, Bitcoin, was developed and published in late 2008 [1] by a pseudonymous figure under the name Satoshi Nakamoto, accompanied by open-source code. Bitcoin introduces a distributed currency, without a central entity (e.g., a bank or a government) controlling the distribution of money or censoring transactions. Instead, the different stakeholders are at the same time the ones maintaining the network’s credibility and the value of the currency.

Bitcoin was the first to achieve this goal of distributed trusted collaboration after about three decades of distributed systems research and many ambitious though incomplete attempts, especially regarding the implementation of a digital and distributed currency. Predecessors – academic or not – include David Chaum’s “Blind signatures for untraceable payments” paper in 1983, Digicash, Hashcash, Bit Gold, and B-money. The innovation of Bitcoin was that it managed to solve the double-spending problem, i.e. make sure digital money cannot be spent twice, just as fiat money cannot be spent twice, in a distributed environment, via an ingenious orchestration of various already-existing tools from cryptography and distributed systems. For a thorough walkthrough of Bitcoin’s “academic pedigree” and more details on the aforementioned attempts, the reader is referred to [2].

Bitcoin’s emergence as a public, permissionless blockchain was followed by many other blockchain projects, public or private, permissionless or permissioned. The distinction between public and private refers to who is allowed to read the blockchain’s contents, while the distinction between permissionless and permissioned refers to who is allowed to join the network as a stakeholder or validator. This article is concerned with blockchains of Bitcoin’s type, namely public and permissionless, and their scalability, as private and/or permissioned systems leverage on the extra assumptions on some of the participants being trusted and thus achieve better scalability properties. We will describe the cryptographic primitives or implementation details of the various mechanisms only to the extent it serves our goal of presenting...
interesting networking problems in the area, and cite relevant sources for the keen reader who wants to delve deeper.

A. BLOCKCHAIN SCALABILITY

Despite the popularity blockchain has gained in the business and entrepreneurship world, its actual usefulness and fulfillment of its purpose as a distributed trusted collaboration technology stumbles upon its lack of scalability [3], [4]. Figures vary slightly, but Bitcoin’s throughput is generally cited to be in the range of 3-7 transactions per second, and Ethereum (the second most popular cryptocurrency) can handle roughly double that amount. When talking about a global payment network, with payment loads like those supported by PayPal, Visa, or other centralized payment service providers and which can reach thousands of transactions per second, one can easily understand that, for this technology to succeed in its vision, it is necessary that we invent our way past this scalability barrier.

![Architecture of a blockchain-based ecosystem.](image)

1) THE LAYER 1 SCALABILITY PROBLEM

In order to better explain blockchain’s scalability problem, we will introduce an architecture of a blockchain ecosystem and then identify the main pain points. Although not officially standardized, a widely accepted hierarchy is the following, shown also in Fig. 1 (adapted from [5]): The base layer is the hardware layer, which comprises of either regular computers, Application-Specific Integrated Circuits (ASICs) or trusted hardware. On top of the hardware operate the nodes, which are connected in a global peer-to-peer (P2P) communication network; this is layer 0. The nodes (called miners in Bitcoin terminology, from the process of “mining” - generating new blocks) maintain the blockchain, a distributed database recording transactions, where the inclusion of a transaction in what is perceived as the truth by everyone is determined through consensus. This distributed ledger of transactions comprises layer 1. Layer 1 is perfectly operational on its own, and applications can be built on top of it. However, as mentioned, it suffers from bad scalability properties. This is the reason another layer, layer 2, has been introduced.

To delve deeper into what makes the consensus layer non-scalable, let us make clear what it wants to achieve. Remember that the blockchain aims to be a distributed ledger of transactions, or, in other words, a record of transaction history. Transactions are organized in batches called blocks, which are linked to one another forming a chain (whence the name blockchain). Initially, all nodes start with a copy of the same block, called the “genesis block.” The consensus is needed to ensure that, after some transactions are generated by some node, they will be eventually included in everyone’s ledger, as long as they do not violate some rules that each node checks before including the transaction in their own record of history. The rules are simple: the coins a node spends actually do belong to the node, and the coins spent have not been already spent (otherwise we end up in a problematic “double-spending” situation).

Therefore, a transaction’s lifecycle is the following. First, a transaction is generated by one node and gossiped to the entire network in order to be validated. Each node checks if the aforementioned rules are satisfied and, if yes, adds the transaction to its “mining pool,” a pool of pending transactions the node maintains. The node then takes the role of a miner: it selects some transactions from the memory pool, adds them to a prospective block, and tries to solve a computationally hard puzzle (essentially invert a hash function) in order to publish the block. Actually solving the puzzle correctly is essential, as only then will other nodes add the block to their chain. In such a case, we say that the block contains a “Proof of Work” (of work done to produce the block). Once the miner solves the puzzle, it publishes the block via a similar gossip process. Normally, the block is the only one at that level of the chain, and everybody adds it to their ledger. There is a slight possibility of a “fork” though: two or more blocks extend the same parent block, resulting in the chain being forked. This can happen either maliciously, or simply because of network delay: a node generated a block at a certain level, but it took long enough for this block to reach all other nodes that in the meantime someone else generated another block at the same level. The protocol parameters (that is the block size and the block mining rate) are tuned to accommodate for the network delay in such a way that this forking event is a rare occurrence, so most of the time the chain grows without this happening.

This block generation process is followed to ensure the safety of the ledger against adversaries with small percentages of the computational power of the network, but it is also what limits scalability. Precisely this repeated gossiping of everything to everyone and validation of everything by everyone is the main factor that makes blockchain non-scalable. Moreover, the highly energy-consuming mining process, for which Bitcoin is often blamed, is in place in order to defend the network against attacks (in particular, the so-called “Sybil

---

1 Satoshi Nakamoto in the original whitepaper claimed resilience of the protocol against adversaries with up to 50% of the network’s computational power. It was later proved though that adversaries with even less power can launch other types of attacks, like the famous “selfish mining” attack [6].
attack”). On the contrary, in a centralized database, all communications and validations happen in one trusted place, resulting in orders of magnitude better performance and without that energy expenditure. Considering also that the rule of thumb for considering a transaction confirmed is to wait for it to be 6 blocks deep in the chain and that on average a block is produced every 10 minutes, the induced confirmation latency of about 1 hour for any transaction is too slow for many applications. These factors demonstrate the blockchain scalability problem and naturally lead to and motivate the layer 2 solution presented in the sequel.

2) PAYMENT CHANNELS AS A SCALABILITY SOLUTION
A natural idea for the decongestion of the main network is to bring the settlement of some of the transactions off the chain. This means that routine validation can only be performed by the transacting parties, and only in case of a dispute would the parties resort to the global slow conflict resolution procedure. This leads us to the idea of a payment channel: the two interested parties open a channel between themselves by depositing some initial funds, and subsequently transact by updating the number of coins belonging to each of them. This happens only inside the channel, so the global network is not burdened with many – possibly thousands – of small transactions, except in case of a dispute. When the parties are done transacting, they can close the channel. The complete lifecycle of a channel is shown in Fig. 2.

Channels were proposed as a blockchain scalability solution and are indeed very promising in this direction. In addition, they offer some level of privacy to the participants, as the frequency and volume of their transaction activity is not revealed to everyone. Channels construct a layer 2, as they are built on top of the blockchain, and therefore do not require any changes to the core protocol. What layer 2 assumes from layer 1 is integrity of transactions, i.e. only valid transactions will be included in the blockchain, and eventual synchronicity with an upper time bound, i.e. every valid transaction will eventually be added to the ledger [5].

The layer 2 protocol accompanying Bitcoin is called Lightning [7], and the one for Ethereum is called Raiden [8]. Lightning consists of more than 6,000 nodes and 32,000 channels at the time of writing, with the total funds in the channels exceeding 1,000 BTC [9]. Currently, there exist three interoperable implementations: lnd, eclair and c-lightning. The cryptocurrency Ripple [10] also features payment channel functionality. Layer 2 protocols other than channels also exist, for example commit-chains and protocols for refereed delegation. The interested reader is referred to [5] for more details. In this article, we focus on payment channels, since they introduce many interesting research problems of a networking flavor. We will now review their functionality, with a focus on Lightning in particular.

II. BACKGROUND IN PAYMENT CHANNELS
Payment channels work particularly well for transactions happening between the same entities, and with high frequency and small amounts. This type of payments is sometimes called micropayments or “streaming” payments. Their characteristics make them “not worth the wait” of the main chain and desirable to be sped up.

A. PAYMENT CHANNEL OPERATION
The channel is established via a so-called MULTISIG Bitcoin transaction, namely a transaction output that requires signatures from multiple parties in order to be spent. Alice and Bob, who want to open a channel, create a transaction that they both sign and where they deposit some initial balances. The sum of these balances is the channel capacity, and cannot change for the lifetime of the channel (the channel has a “conservation of capacity” property). The transaction is released to the Bitcoin network and Alice and Bob have to wait for it to be confirmed via the regular slow procedure. Once it is confirmed, they only exchange updates on the balances on the two sides, without interacting with the main chain (whence the term “off-chain transactions”). Both nodes sign the updates, which are called “commitment transactions,” and are containing a timelock: they cannot be redeemed before the timelock expires. If Bob tries to cheat, Alice can take the last update signed by both nodes, which is a valid

2https://github.com/lightningnetwork/lnd
3https://github.com/ACINQ/eclair
4https://github.com/ElementsProject/lightning
5In practice, Lightning channels are usually single-funded initially [11], as setting up a dual-funding channel requires some trust between the two nodes or is more cumbersome if they do not trust each other [12].
Bitcoin transaction, and publish it to the main chain. Later commitment transactions have shorter timelocks than earlier ones. Thus, a node participating in a channel can be sure that it will not lose any funds even if the counterparty tries to submit a previous commitment transaction to the blockchain. Thus, the channel satisfies the “trust-free” principle of the blockchain. When the nodes do not need the channel anymore, they close it by publishing one more on-chain transaction and collect the final channel balances by adding them to their on-chain balances.

It is important to note that the funds a node has in a channel are locked in the channel until the channel closes. This means that the node cannot use them to pay for another on-chain transaction or to establish a new channel. Also, funds cannot leave the channel they are on and move to another channel.

Visually, a payment channel resembles a row of an abacus, in which the total number of beads in the row is constant, but the beads can be distributed in any way on the two sides of the row.

B. NETWORKS OF PAYMENT CHANNELS

The network formed with the nodes being coin owners and the links being payment channels is called a Payment Channel Network (PCN). Unlike the main network, where everyone can pay everyone else as long as they know their address (their public key), the PCN is not a complete graph. Reasons for this include the finiteness of the amounts the nodes have on the blockchain, the fact that the funds in a channel are locked in that channel for its lifetime, and the rarity or even inexistence of a need for transacting between certain nodes. These factors render the establishment of channels between every pair of nodes inefficient and not actually necessary. Whenever the need arises for a transaction to happen between two nodes that are not connected via a channel, two possibilities exist: either they open a channel, or they try to route the payment via multiple existing channels, utilizing the funds other nodes already have in them.

An example of a PCN is shown in Fig. 3. Suppose Alice wants to pay 3 coins to Carol. Since she does not have a direct connection to Carol, Alice has to resort to a multihop path. She could, for example, use Bob’s channel with Carol: Alice will pay 3 coins to Bob in the Alice-Bob channel, and then Bob will pay 3 coins to Carol in the Bob-Carol channel. The updated balances will be (0, 7) in the Alice-Bob channel and (0, 10) in the Bob-Carol channel.

Multihop payments introduce several new aspects of PCNs. First of all, a multihop payment has to happen atomically: either all steps will complete successfully or none of them will complete. In Lightning channels, this is guaranteed through the cryptographic construct of Hashed Time-Lock Contracts (HTLCs) [7]. HTLCs allow for chaining of payments, thus avoiding problematic situations like one node paying first and then another node in the payment path not cooperating and stealing funds. Second, some route discovery work is necessary. Suppose that, in the PCN of Fig. 3, Alice wants to send 4 coins to Carol. Now the Alice-Bob link cannot support this payment, as Alice has only 3 coins in it. So, if the payment is to be sent in one chunk, this has to be done via the Evelyn-David-Carol route. This demonstrates the need for payment routing in PCNs. At a later stage in the development of Lightning, the possibility for Atomic Multipath Payments (AMP) was added. So Alice could also process the previous payment by sending 3 coins through Bob and 1 coin through Evelyn. Still though, the need for routing remains.

Before we introduce several interesting problems that arise in a PCN, we need to introduce some factors involved in a successful payment.

1) INFORMATION AVAILABLE TO NODES

First, let us describe what knowledge the nodes have in the network. Remember that a channel is established through a regular Bitcoin transaction that lives on the main chain, and hence is visible to everyone. So the connection topology of the PCN, with the initial (but not the current) balances, is known to everyone, as it can be derived by simply parsing the blockchain. The knowledge of the initial balances also reveals the capacity of each channel, which is the sum of the initial balances and, as mentioned, remains the same throughout the channel’s lifetime. What is not known to the nodes, though, are the current channel balances for channels they are not participating in. Of course, Carol knows the current balances in her channels with Bob and David, but she has no idea about the current balances for instance in the Alice-Bob channel, other than that it started with initial balances of (3, 4). Relaxations of the secrecy around balances are possible and will be discussed in the sequel, but Lightning specifically aims to operate in a highly privacy-focused manner, leading its developers to such a decision. So the graph of Fig. 3 depicts an all-seeing (and actually inexistent) entity external to the network rather than the view of any individual node. We should note that in practice the network is a multigraph, as two nodes might have more than one channel between them. Other information that is necessary

---

6This dispute resolution mechanism is followed in the implementation of payment channels according to [13]. Meanwhile, in the implementation according to [7], the misbehaving party is punished by losing all the channel funds to the other party.
for routing, like fees, is propagated via a gossip process between the peers.

2) PERFORMANCE METRICS
The lack of knowledge of the current balances for most channels in a PCN inevitably leads to some payments failing by taking a route which includes a step with insufficient balance (less than the payment’s value plus remaining fees). Thus, the metric usually used for quantifying the performance of a PCN is the overall payment success rate: how many payments succeed end-to-end out of all payments attempted (either per originating node, or for the entire network). Other useful metrics include the successful payment volume (throughput) – normalized or not, the (average) payment path length, the total payment duration and the collateral cost [14].

3) FEES AND INCENTIVIZATION
The next element necessary for multihop payments in PCNs is an incentivization mechanism for the intermediate nodes in a payment path. Note that the completion of a multihop payment involves several steps, and all involved nodes lock some funds in this process until it completes. Although they cannot lose their funds, still they cannot use them for their own benefit either inside or outside the channel until they are released when the end-to-end payment either succeeds or fails. So why should an intermediate node relay payments by offering its liquidity and locking its funds for the benefit of others? The answer is relay fees. In the words of the Lightning whitepaper [7], “the time-value of fees pays for consuming time (e.g., 3 days) and is conceptually equivalent to a gold lease rate without custodial risk; it is the time-value for using up the access to money for a very short duration.” Thus, multihop payments operate as follows: The original payer actually routes a slightly higher amount via the payment path to the final recipient, and the extra amount is taken as fees by the relays. As a result, the total balance of a node in all its channels will slightly increase. The presence of fees introduces many problems related to incentives and network design.

4) PAYMENT DEADLINES
Another important element of multihop payments are deadlines. Because funds are locked end-to-end in a payment path until the payment is completed, a malicious node could take advantage of this to have other nodes lock their funds forever by commencing a multihop payment, establishing a path, and never completing it. Or one of the intermediate nodes in the path could crash, and then everyone would have their funds locked forever. To prevent these situations, every multihop payment comes with a deadline for its completion, expressed absolutely or relatively to the current time either as a concrete time in the future or in terms of block height on the blockchain. If the payment is not completed end-to-end by the deadline, all the funds locked by all the nodes in this path are released. The deadline decreases at each payment hop.

5) A COMPLETE EXAMPLE
A complete description of a multihop payment is shown in Fig. 4. In this example, Alice wants to pay 4 coins to David. Note that Alice and David can communicate directly or indirectly (e.g., for the exchange of necessary keys and for route discovery), but do not have a channel between them and thus cannot exchange funds directly in a trusted way. The path discovery has already happened and Alice has decided to use the path Alice → Bob → Carol → David for the payment. First, Alice asks David to think of a secret R7 and send the hash H8 of R to her. Then Alice constructs a message with the payment containing this hash H, a deadline (starting at 5 time units) and the amount, in this case 4, plus some fees that the intermediate nodes will take (hence 4.02 coins in this case), and sends it to the next hop, in this case Bob. Bob, takes off his fee portion (0.01 coins), decreases the deadline by 1, and forwards the updated message to the next step, Carol. Carol follows the same process, until the message reaches the final recipient, David. Note that during this process the balances of all nodes on the path remain unaltered. The changes will only be applied if the payment completes successfully end-to-end, atomically. For this to happen, David needs to share the secret number R with everyone else in reverse order of the path. By sharing R with Carol, David can get his 4 coins. Carol then shares R with Bob and gets her 4.01 coins. Finally, Bob shares R with Alice and gets his 4.02 coins. These amounts are now reflected in an update in all the channel balances.

In the unfortunate event that, say, Carol is uncooperative, the expiration of the deadline will allow everyone else to withdraw their funds without losses. Also, Carol cannot cheat.

7In Lightning, it is the receiver of the payment who generates the secret, while in Raiden the secret is generated by the sender [8].
8A hash function is a function that maps data of arbitrary length to an output of fixed length. The data can be viewed as a single long number that is provided as input to the function. Lightning uses the SHA-256 hash function.
and get her 4.01 from Bob without paying 4.00 coins to David, as the latter is required in order for David to share the necessary secret R with her. For a more thorough visual explanation of the HTLC mechanism, the reader is referred to [15] and for the actual cryptography behind it, as well as other practical constraints and implementation details of various aspects of the system to the original Lightning whitepaper [7] and the Lightning Network specification [16].

Armed with the knowledge of the specifics of the functionality of a payment channel and a PCN, we can now proceed with examining some interesting research problems arising in the area.

III. NETWORKING AND ECONOMICS ASPECTS OF PAYMENT CHANNEL NETWORKS

A. ROUTING

Payment routing is of paramount importance in PCNs, as an operation needed every time a payer and a payee are not directly connected. Routing can follow many variations – which we describe below – depending on the efficiency, speed and privacy levels required. Analogies and inspiration can be drawn from routing algorithms applied in other types of networks. Fundamental factors that play a role in choosing a path are the balances on the route and whether these are known to the nodes, the relay fees, and the payment deadlines. Desiderata when designing a routing protocol include autonomy of each node to act independently, flexibility when changes in the network occur, trustlessness, and resilience against nodes with arbitrary or malicious behavior [17].

1) DIFFERENT APPROACHES TO ROUTING

A fixed path for all payments between two nodes resembles static routing from traditional networking, while a determination of the path for each individual payment can be viewed as the analog of dynamic routing.

Routing between certain source and destination nodes can be performed entirely by the source, which will determine the entire path (as in link state routing), or at each step, with each node forwarding the payment to an appropriate next step (as in distance vector routing). Although currently Lightning follows source routing, the more dynamic and decentralized approach is being studied as an alternative. Issues of malicious behavior might arise in such a scheme, since an adversarial relay node could send a payment along a longer route to collect more fees or make it fail (this is not possible in source routing, as the path is predetermined). This approach aims to achieve throughput optimality and take full advantage of the available funds (for example, [18]). Implementation of such schemes might require modifications in the HTLC establishment mechanism for multihop payments due to the per-step nature of the routing decisions, as well as introduction of per-node queues of in-flight transactions.

In another effort to increase the payment success rate, fragmentation of payments is possible with the introduction of Atomic Multipath Payments [20], through which a single big payment can be split into multiple small ones, each of them possibly following a different path and joining the others at the final destination. This fragmentation is analogous to the one happening in data networks, where information is split into packets, and can improve the success rate of large payments that would require going through a big central node or would otherwise fail due to insufficient funds of the intermediate links. This results to higher decentralization, increased reliability of payments, and better privacy, as a result of obfuscated total payment amounts. Moreover, it improves the profitability of nodes, as their costs from locking capital remain the same, but they are able to use it to a greater extent and earn forwarding fees [21]. An interesting research question arising is how to find the optimal split given the payment demand and the network conditions. An idea currently being used is “recursive halvening” [21], however, an optimization that takes into account all the factors involved in the splitting decision could lead to higher success rates and lower costs.

When a node determines the path or the next step for a payment, it needs to choose a metric according to which different options will be ranked. The cheapest alternative in terms of fees is an option, while the shortest in number of hops (for the path planning case) is another. Lightning currently employs a combination of both according to a certain formula embedded in the code.9 This choice, though, leads to underutilization of the available resources [22]; different combinations might perform better. Another way to determine a route, assuming payment flows across the network are known, is to calculate

9https://github.com/lightningnetwork/lnd/blob/master/routing/pathfind.go
the max-flow path from the source to the destination, where the flow allowed via a link is either the balance on the side considered (if known), or some estimation of the balance if only the capacity is known (e.g., a random number in the interval [0, capacity]). Multiple flow-based approaches have been published [18], [22]–[26].

Other approaches to routing include beacon-based routing with proactive aggregation of information [27], landmark-based routing [28], embedding-based routing [29], ant routing [30], and Ad-hoc On-demand Distance Vector (AODV)-based routing [17].

2) PRIVACY
Privacy issues are taken heavily into account in the Lightning community. For this reason, onion routing is being used for multihop payments (namely encryption of payment information in a nested fashion for consecutive steps), so that relay nodes that receive a payment only know who they received it from and who they should send it to, without revealing the original sender and the final recipient. Different types of privacy to be preserved can be defined [28]. Sender (receiver) privacy is achieved in a payment if an adversary cannot determine the sender (receiver) in a transaction between non-compromised nodes. Value privacy is achieved if an adversary cannot determine the total value of a transaction between non-compromised nodes. Finally, link privacy is achieved if an adversary cannot determine the balances on a channel between non-compromised nodes. The routing used by Lightning currently achieves link and – to some extent – sender privacy. Increasing the payment success rate is constrained by the desired degree of privacy. This leads to privacy-utility tradeoffs: revealing more information (e.g., the exact channel balances) would lead to higher success rates, but less privacy [31].

3) CONCURRENCY
PCNs have to face concurrency issues as well, as simultaneously scheduling multiple transactions can result in deadlocks [32]. Nodes are allowed to serve and forward multiple payments at the same time, as long as they have enough balance in their channels. However, a deadlock occurs when two or more individually feasible multihop payments share edges in their paths in a way that none of them can progress. This happens when an edge needs to be used by multiple payments but is reserved by one of them, thus blocking some other payment, which however is in turn blocking an edge needed by the first payment (see Fig. 6 for an example adapted from [32], [33]). In this worst case, a deadlock occurs with no payment being able to execute, although all of them might be individually feasible [33]. Of course, the expiration of the deadline will solve the problem, but only after some wasted time. Deadlocks can only happen if there is a cycle in the channel graph. Routing mechanisms should be designed keeping in mind that deadlocks need to be avoided. Alternatively, non-blocking mechanisms for concurrent payments have to be introduced [32].

![A deadlock situation in a PCN.](image_url)

**FIGURE 6.** A deadlock situation in a PCN. Suppose every channel has a balance of 1 coin in the direction of the arrows and 0 in the opposite, and two payments of 1 coin each are attempted concurrently: one via the path $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$, and one via the path $F \rightarrow D \rightarrow E \rightarrow B \rightarrow C$. The colored arrows indicate up to what point each of the payments reached in its path before getting blocked. A deadlock occurs because the 1 coin of the channel $D \rightarrow E$ needed to complete the first payment is reserved by the second, and at the same time, the 1 coin of the channel $B \rightarrow C$ needed to complete the second payment is reserved by the first.

B. SCHEDULING, DEADLINES, LOAD BALANCING AND CONGESTION CONTROL
Routing, as described, only considers one payment in the network. However, when there exist multiple simultaneous payments, as happens in reality, if many of them follow the same path, some paths might get congested. Considering that payments have a certain deadline to meet in order to complete successfully, it is important that routing and scheduling of payments happens in a way that controls the load of channels to achieve maximum successful completion of payments.

The load can be balanced at the channel level, the path level, or the node level:
- Load going through particular channels can be distributed so that the channels are not overloaded (which could be harmful for the network, e.g., if the channel is a central one).
- Load between proximal sources and destinations can be distributed across different paths, in a way similar to what happens in the load balancing of Internet traffic.
- A node might not want to route too many payments in certain directions in its channels, so that the channels do not get depleted and have funds for some higher priority transactions.

Congestion control can be also explored in a similar fashion, in the sense that the routing algorithm proactively limits the load imposed on a channel, path or node. Techniques applied in classical networks might be adaptable in this setting, as is done for example in [22] drawing inspiration from Multipath TCP (MPTCP). It should be noted that, unlike communication networks, achieving low latency is not always necessary depending on how crucial the transaction is, since the alternative of processing it on-chain is by orders of magnitude slower anyway [22].

C. REBALANCING
Lightning channels are characterized by their capacity throughout their lifetime. This can prove to be problematic in situations where one of two sides of the channel is more active than the other in terms of transaction volume. This would lead to a drainage of funds on the one side and an
accumulation of funds on the other. This, in turn, would result in subsequent transactions incoming at the drained side being rejected, as there would exist no available funds to process them. A solution to this problem would require rebalancing the channel. An interesting quantity to be studied for each rebalancing method is the routing capacity between the source node and a target destination node, namely the maximum total amount that can be sent to the destination via all the possible paths [34].

Rebalancing can be done in the following ways:

1) OFF-CHAIN: CIRCULAR SELF-PAYMENTS
A node with more than one channels that has a drained channel on its side can discover a circular path in the PCN back to itself, with the drained channel as the last step, and send a payment to itself via this path [11], [35]. This would lead to a redistribution of the node’s total off-chain funds between its channels, adding liquidity to the drained side. Of course, as long as this cycle involves other nodes’ channels (which is most likely), the rebalancing will cost some amount in relay fees. Nodes need to be able to discover such circular routes in their neighborhood (e.g., via a pathfinding algorithm applied to the same source and destination). A visual illustration is given in Fig. 7. A constraint of this method is that the maximum rebalancing amount is constrained by the balances on the circular route.

![Figure 7. Circular rebalancing performed by Alice (fees are omitted). Left: Channel balances before the rebalancing. Right: Channel balances after Alice performs a circular payment of 4 coins to herself via Carol and Bob.](image)

2) FEE MANAGEMENT AND NEGATIVE FEES
A node can balance its channels by making it favorable to other nodes to use them for their payments via setting low, zero, or even negative fees [7]. Offering negative fees means that the node will pay others to route payments through its channels, and might be worth the cost if rebalancing certain channels this way enables higher gains (e.g., by enabling routing once again at the before-depleted side of the channel).

![Figure 8. Rebalancing a channel via Loop (fees are omitted). Top: configuration before the rebalancing. Middle: On/off chain rebalancing. Bottom: Configuration after the rebalancing.](image)

3) ON/OFF-CHAIN: SUBMARINE SWAP (LOOP)
Another rebalancing technique that was developed later as an addition to the Lightning protocol as a hybrid on- and off-chain solution is called “submarine swap,” is implemented as “Loop” [36] and is based on the following idea: Suppose a channel between nodes A and B is being depleted on A’s side, and A wants to add funds of value \( v \) on her side. Instead of closing the channel and launching a new one, A can pay B an amount of \( v \) via an on-chain transaction, and B will give this amount \( v \) back to A off-chain: the amount will be moved from B’s side of the channel to A’s side. The cost of this process consists of the on-chain transaction fee (plus a fee that the swap server B might request from A). Fig. 8 provides a graphical explanation.

The fact that Loop rebalancing involves an on-chain transaction introduces some delay in the process, greater than in the circular route rebalancing. Since the on-chain delay is orders of magnitude higher than the off-chain delay, a node considering Loop rebalancing needs to start planning well in advance of its channel depletion in order to not run completely out of funds. Questions regarding how this planning should be made and which of the two forms of rebalancing is more appropriate in a given situation arise.

4) SPlicing
Splicing [37] performs rebalancing by changing the capacity directly and not via the balances. The term refers to the on-chain rebalancing that happens when two nodes that have a channel close and reopen the channel with altered capacity in a single on-chain transaction. If the new capacity is higher than the old one, the operation is called “splice in,” otherwise it is called “splice out.” Splicing has the disadvantage of having to pay and wait for an on-chain transaction, but on the other hand gives complete freedom to the node to deposit or withdraw funds to/from the channel. The Lightning developer community has been investigating ways to accomplish the same functionality (a change in the channel capacity) while keeping the channel open and fully active (for example, by having the two nodes updating both the old and
new channels until the new channel generation transaction is confirmed on-chain).

**D. NETWORK AND FEE DESIGN AND INCENTIVIZATION**

1) CHANNEL CREATION AND FEE OPTIMIZATION

Another interesting family of problems arises around the choices nodes can make on how to use their funds, which channels to open and how to determine their own relay fees [38]. Suppose a Payment Service Provider (PSP) exists, namely an entity that controls a lot of liquidity and possibly multiple nodes/channels. The decisions of such a PSP can be formalized as optimization problems. For example, if the PSP knows the payment demands along certain directions, it can determine a payment admission policy that will guarantee that its funds will not be depleted in some channels and that the fees it receives are maximized. Also, given the topology, the problem of when the nodes have incentive (when is it profitable for them) to open a new channel versus using the network arises. A variation to consider is the following: given some amount a node is willing to “invest” into opening new channels, which channels should it open based on payment demands and fees? How to gain a central position in the network? For what transaction amounts is it better to use the blockchain, Lightning, or neither, based on each one’s cost in fees [40]? Optimal solutions to such problems are relevant to the implementation of the Lightning “Autopilot”, a feature that helps users by recommending the best channels to open (e.g., when they first join the network) and currently operates on heuristics and centrality measures. One can also consider network creation games [41] or game theoretic problems with strategic agents trying to game others and manipulate the routes via appropriate fee selection in order to maximize their profit or to increase their benefits above their fair share in the network. Interesting tradeoffs arise, as higher fees might lead to higher profits but at the same time to lower probability of being selected as a relay. In any case, fee optimization [42], [43] is an open area of research, especially as current fee structures might be economically irrational for many routing nodes, meaning that, for the current actual traffic volumes, fees have to rise to make the network economically viable [44].

Layer 2 fees are determined as a function of the payment amount, as happens in traditional finance. In particular, a relay node charges a base fee for others using his channel, plus a proportional fee rate times the payment amount. In contrast, in Bitcoin (layer 1) miners charge fees according to the transaction size in bytes, regardless of the amount. This means that the exploration of layer 1 economics done so far is not directly transferable to layer 2 economics. The latter has to be done from scratch and can possibly benefit from the traditional economics domain.

[10] At least one Lightning implementation, c-lightning, allows opening multiple channels with a single commitment transaction [39].

[11] https://github.com/lightningnetwork/lnd/tree/master/autopilot

[12] According to [45], the current median base fee is 1 satoshi ($=10^{-8}$ bitcoins – 1 satoshi is the smallest unit recorded on the Bitcoin blockchain) and the current median fee rate is 0.000001 ppm (parts per million).

2) WATCHTOWERS

An issue faced by PCNs is that in order for nodes to be able to transact, they need to be online at the same time, and even all the time if they want to function as relays. This is difficult and sometimes impossible, and has led to the introduction of “watchtowers” [46]: nodes that function as beacons monitoring other nodes’ channels while the nodes are offline, for a compensation of course. With the introduction of watchtowers, concerns around decentralization arise, as a watchtower aggregating the overlooking control over several channels might be able to proceed to malicious actions. How to properly incentivize honest watchtower behavior is an interesting problem that needs further investigation [47].

3) CHANNEL FACTORIES

Another interesting proposal is the one of channel factories [48]. The paper proposes a new layer on top of Lightning that can enable trustless off-chain channel funding, instead of on-chain channel funding as described before (and which means that the initial funding transaction of the channel has to be confirmed on-chain). With channel factories, a group of $n$ nodes that want to pay each other, instead of opening $m(n-1)$ channels among themselves (which would create $m(n-1)$ on-chain funding transactions and $m(n-1)$ more for the eventual closure), they invest some initial amounts and create a channel factory with only one pair of open-close transactions by using an $n$-of-$n$ MULTISIG transaction. Then, within the group, channels can be opened and closed without announcing anything to the main chain, and thereby not suffering the associated costs and confirmation delays. Rebalancing can also be done more easily in this microcosm, e.g., by closing and reopening all depleted channel as balanced ones with internal redistribution of funds and without on-chain announcements (see Fig. 9 for an example). Higher order structures of this kind are also possible.

4) VIRTUAL CHANNELS

Another proposed concept is the one of virtual channels [49], which aims to be an alternative to multihop payment routing. When two nodes want to pay each other, they face the decision of whether to open a channel between themselves or utilize a multihop path. Apart from the cost of each option, a tradeoff arises between the amount they will block in a new channel and the interaction needed with other intermediary nodes. Interactions, although safe thanks to the protocol’s design, can be problematic as they introduce reliability issues (for example if one of the intermediate nodes is not online), fees, and delays [50].

Virtual channels attempt to minimize the need for interaction with intermediaries by creating a bridge between the two nodes, and limit that interaction to only the setup and the closing of the channel. The intermediary functions as a payment hub: it is connected with both transacting parties

[13] Virtual channels as described in [49] require smart contract capabilities (a Turing-complete scripting language, like in Ethereum), but recently efforts have been made to realize virtual channels in settings with more limited scripting capabilities [51].
A and B with normal “ledger” channels (backed by the blockchain) and initially offers some liquidity in both sides, which constitutes the balances of the virtual channel. From then on, A and B transact and update the balances without the involvement of the intermediary. When they are done, they update the ledger channel balances and close the virtual channel. Incentives for the intermediaries are of course important and can be materialized in the forms of fees [52]. The concept can be extended to multiparty virtual state channels [53]. Virtual channels are appropriate for applications like selling microservices and data especially in ad-hoc network or Internet of Things settings, where connectivity can be intermittent and reliance on any intermediary introduces delay and a point of failure.

E. TOPOLOGY

PCNs, as a type of complex networks, can also be studied from a network science point of view, with regards to their topological properties. Such analyses involving for example connectivity, centrality, and assortativity metrics can offer insights about the resilience of the network to partitioning, to random failures, to attacks against specific hub nodes, and to deanonymization attacks. They can also reveal to what extent the vision of the network to be decentralized is achieved in practice, as the need for efficiency necessitates that small nodes connect to wealthy nodes that act as payment hubs and thus centralize the system. Data analysis can also reveal different correlations between quantities that are not obvious. Finally, the study of the capacities in the network can reveal an upper bound on the maximum possible cash flow between different parts of the network, information that is useful for assessing the optimality of any possible design of a routing scheme.

Some research has already been done in this direction of studying the topology. The work in [54] shows that Lightning’s topology exhibits scale-free properties and calculates metrics such as node degrees, number of bridges, connected components, assortativity, clustering coefficients and more. For example, they calculate the length of the average shortest path to be 2.8, meaning that on average there are roughly 1-2 intermediaries for every payment. Another interesting result relates to the way that new nodes connect to existing ones: if done by trying to maximize centrality, this leads to a preferential attachment pattern and subsequently to a hub-and-spoke network. A similar behavior is exhibited by the allocation of capital in the network. The study also shows how easily the biggest connected component gets disconnected if a few central nodes are eliminated. Thus, although Lightning is shown to be quite robust in the face of random failures, it is vulnerable in the face of targeted attacks. Related issues of centralization, robustness, synchronization, and anonymity properties of the topology are also studied in [55]. A percolation point of view of Lightning topology is taken in [56]. However, as the topology evolves and the dynamics change, questions such as the ones stated above will need to be addressed or revisited.

As a last note on the topology, we should mention private and zombie channels. A significant portion of the total Lightning channels are so-called “private” channels, namely channels not announced to the rest of the network to be used for routing [57]. This is rather a misnomer, as their initial commitment transaction is publicly visible on the blockchain. A more appropriate term would be “unannounced” channels. Moreover, certain channels have a channel partner which has gone permanently offline, therefore rendering the channel unusable. These are called “zombie” channels [57] and cannot be easily identified, as the counterparty might come back offline, but are harming the online party as it is forced to keep liquidity locked without being able to use it. Knowledge of the existence of these types of channels is important whenever designing algorithms or deriving results based only on the “public”-announced channels, or whenever making claims about the entire topology of the network.

F. CHALLENGES AND EVALUATION DATA

Assuming operation in full privacy, as in Lightning, the main challenge in studying the network aspects of PCNs is the lack of knowledge of the channel balances. This is a significant difference from other types of network modeling (e.g., communication networks), where the link capacity (equivalent to the balances in PCNs) is known.

Another challenge is related to the metrics used for evaluating the performance of a PCN. The success rate is a valid metric, however it is impossible to measure in a real PCN, as what happens to transactions is not transparent to the entire network, but rather only to the involved nodes in the transaction path in case of either a success or a failure. Some studies have followed the approach widely also used for studying Bitcoin’s topology, which is to establish one or more regular Lightning nodes and monitor the transaction traffic. However, this method again provides a limited view of the network.

Finally, an important aspect of any work seeking to improve a system is its evaluation on real data. In the case of PCNs, these data include the graph topology, the channel capacities and balances, transacting pairs and transaction
amounts and timestamps. Topology data, namely connections and channel capacities, are available either by parsing the blockchain, where this information is public, or through online aggregators and explorers. Unfortunately, though, beyond that, the privacy-first stance of Lightning renders the evaluation task very difficult: not only are the instantaneous channel balances hidden from everyone except the nodes maintaining the channel, but also onion routing prevents any centralized data collection on the amounts of transactions happening in the system. Therefore, the options for evaluation of a routing/rebalancing scheme or any proposed modification are constrained to the following:

- Use synthetic data: Define a distribution for picking nodes to transact (for example, uniform over nodes), and a distribution for the amounts, e.g., constant [44], uniform [58], normal [59], Pareto [31], Poisson [60], power-law [40]. Also, define the frequency of interactions and the initial distribution of balances in the channel. Using synthetic data is a frequent approach, but has the drawback that the data are artificial.
- Use a transaction dataset from another system, like Ripple [26], [29], [61], [62] or Venmo [63], debit/credit card data [58], or even cross-border payment data. The main issue with this approach is the assumption that Lightning transactions will behave in a similar manner to another system’s transactions.
- Use data privately collected by a big and central Lightning node, which is expected to be routing a large amount of traffic and therefore can at least give a distribution for the transacted amounts. However, still the two endpoints of the payment are not known to the node.
- Set up a node and collect the data. Drawbacks are that this requires investing funds (unless done in the testnet, but then the transactions are not real), and that it is unlikely that the node will be central enough to be routing enough traffic that will be representative of the overall network traffic. Endpoints of payments again would be unknown.

All of the above approaches have drawbacks with respect to how realistic and representative of the network’s traffic the dataset is. However, in the absence of an appropriate dataset, an ideal alternative does not seem to exist.

IV. OTHER PAYMENT NETWORKS

A. STATE CHANNELS

Bitcoin’s purpose is limited to being a decentralized digital currency. Ethereum was later created to support more decentralized applications via smart contracts, i.e. contracts written in software and enforced through consensus. Smart contracts enable more elaborate business logic (supported by a Turing-complete scripting language) and can be simply described as transactions that happen conditionally. An entire state can be stored in and modified by smart contracts.

In the same way Ethereum extends Bitcoin with state processing capabilities, state channels extend payment channels to support arbitrary state operations. For example, we can imagine two nodes playing chess. The rationale is the same as for introducing payment channels: the entire network does not need to monitor the events between two parties. An initial smart contract creates the state channel and functions as the “judge.” From then on, the channel participants only exchange state updates between themselves, with the main chain invoked only in case of a dispute. State channels can support multiple participants and are essentially a generalization of payment channels: what is being tracked is the state of an arbitrary program – payments are just one possibility. State channels can form networks and similar issues as in PCNs can be studied. Some examples of state channel projects are CelerX [64], “State Channels” [65], Perun [50], and Connext Network [66]. For a more comprehensive list, the reader is referred to [67].

B. CREDIT NETWORKS

The described payment channel networks such as Lightning resemble, in some sense, debit networks: nodes have some liquidity, and can only spend up to the amount they own. Credit networks also exist [68]–[70], with their existence actually preceding the one of blockchain-based PCNs. In a credit network (or IOU – “I owe you” network), nodes can actually spend more than they own. Such an action results in a due credit, or in other words, a negative balance. Additionally, in such networks, the constraint of a constant link capacity is not valid anymore. Instead, nodes might trust each other up to a maximum value of credit. In some sense, credit networks are a generalization of PCNs. Other similar networks exist in monetary economics, for example in interbank cross-border settlements.

C. CROSS-CHAIN TRANSACTIONS

HTLCs, the enabler of multihop payments in a PCN, can also be used to enable cross-chain swaps of different cryptocurrencies. Instead, for example, for one to rely on a cryptocurrency exchange to sell their Bitcoin to buy Ethereum, they can do so via an atomic swap of funds between the two chains, with the correct funds atomically recorded to either both or none of the Bitcoin and Ethereum blockchains. Multistep payments are also possible (e.g., Bitcoin to Ethereum to Litecoin), and such swaps are becoming more relevant as the area of Decentralized Finance (DeFi) is emerging. Similar problems as the ones described for PCNs might be encountered in cross-chain transactions. Some indicative projects in this area include Interledger [71], Polkadot [72], and Cosmos [73]. For a survey on cross-chain transactions, the reader is referred to [74].

D. CENTRALIZED PAYMENT SYSTEMS

The vision to enable easy (micro)payments over the Internet with short delays and low fees has existed outside the cryptocurrency world as well for quite some time. Several payment platforms exist and aspire to draw customers to use their improved service as opposed to the slower and more expensive process of going through a bank to make a payment or a cross-border transfer.

Venmo, currently only operational in the United States, offers easy payments mainly between people who know each
other, in a user-friendly way integrated with a messaging application. Venmo made the default choice for all transactions through its platform being public (the user can opt out, but most users do not) [63]. This is in stark contrast to Lightning, where privacy is safeguarded by default, even at the expense of lower payment success rates.

PayPal is more friendly for payments towards businesses and allows anyone having an email address to send money globally. Both of the transacting sides provide their bank account or credit card information to the company, allowing it to function as a trusted intermediary for the transfer. This way, the sensitive user information stays within PayPal’s systems instead of being transmitted over multiple steps over the Internet [75]. PayPal forwards the requests for money transfers between its own and the transacting parties’ banks and withholds some fees during the process. For more details, refer to [76], [77].

TransferWise aims to provide transparent cross-border payments without hidden fees and cheaper than going through the traditional banking system. The company achieves this by transforming cross-border payments to domestic ones, by maintaining bank accounts in multiple countries. When a European customer wants to pay someone in the United States, they can send euros to TransferWise’s account. TransferWise will charge some fee and then use the current exchange rate for euro-to-dollar and pay the final recipient from the company’s US bank account in US dollars. This way, the money never actually crosses any borders and the company avoids fees of converting between different currencies [78].

All of the above services make profit from flat or proportional fees associated with the transfers, which might apply to all payments or only for payments where the client wants expedited service. PayPal in addition earns interest by depositing the money consumers have in their PayPal accounts to a traditional bank [75].

Conceptually, the role of such an centralized intermediary can be viewed as a network involving the consumer, the merchant and the intermediary connected as follows: each of the consumer and the merchant maintains a payment channel with the intermediary. A payment from a consumer with some positive balance to the merchant through the intermediary is similar to a multihop payment in a PCN: the payment amount is transferred from the consumer’s side to the intermediary’s side of the channel, and in the next step the same amount – reduced by some fees – is transferred on the second channel from the intermediary’s side to the merchant’s side. If the consumer does not have enough balance in the beginning of the process (which is allowed by the terms of some services), the situation gets more complicated, as the consumer’s bank needs to enter the loop. Of course, chances that a payment will fail are much lower in these systems compared to PCNs.

Another similarity of TransferWise in particular with the PCNs described before is related to keeping the flow of money between currencies balanced. As TransferWise maintains accounts in multiple currencies, the balance in each of one of them needs to suffice in order to cover the needs of the local market. If, however, the flow of money is unbalanced in some direction (for example, from a developed to a developing country), the company will need to purchase and inject additional funds in the depleted side (similarly to the payment channel rebalancing case). This incurs a certain cost, but the low cost and high trading volume on balanced routes compensates for it [78].

Finally, payment channels exhibit similarities to traditional payment infrastructures in the following sense: a payment channel is like a safe where both parties lock funds, and then update the state of the safe like an ever-shifting balance sheet. This is similar to the concept of real-time gross settlement (RTGS) and continuous linked settlement (CLS) from traditional banking [79]. A significant difference though is that payment channels operate in a trustless manner.

Centralized models like the above are subject to the single point of failure problem, which is made visible when hackers access sensitive databases or a government asks for access to payment records. Their operation, however, might lend itself to analysis similar to the one performed in their distributed counterparts – PCNs – and lead to interesting conclusions.

V. CONCLUSION

In this article, we reviewed payment channels networks and outlined several interesting research directions in this area. Starting from detailing the blockchain scalability problem and the need for a scaling solution, we presented an off-chain transaction mechanism – the payment channel – as well as the networks formed by payment channels. This new type of networks resembles traditional networks to some extent, but also calls for PCN-specific research to improve their scalability and performance given their special characteristics, including the requirement for privacy. Our hope is that the present article will provide a starting point and inspire more researchers to enter and contribute to this emerging field.

REFERENCES

[1] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Accessed: 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] A. Narayanan and J. Clark. “Bitcoin’s academic pedigree.” Commun. ACM, vol. 60, no. 12, p. 36–45, Nov. 2017. [Online]. Available: https://doi.org/10.1145/3132259
[3] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, “On scaling decentralized blockchains,” in Financial Cryptography and Data Security, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and R. Rohloff, Eds. Berlin, Germany: Springer, 2016, pp. 106–125.
[4] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, “Stochastic models and wide-area network measurements for blockchain design and analysis,” in Proc. IEEE Conf. Comput. Commun. (INFOCOM), Apr. 2018, pp. 2546–2554.
[5] L. Gjildeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, “SoK: Layer-two blockchain protocols,” in Financial Cryptography and Data Security, J. Bonneau and N. Heninger, Eds. Cham, Switzerland: Springer, 2020, pp. 201–226.
[6] I. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” Commun. ACM, vol. 61, no. 7, p. 95–102, Jun. 2018. [Online]. Available: https://doi.org/10.1145/3121998
[7] J. Poorn and T. Dryja. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. Accessed: 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf
[8] Raiden Network. Accessed: Oct. 30, 2020. [Online]. Available: https://raiden.network/101.html
S. M. Varma and S. T. Maguluri, "Throughput optimal routing"

R. Pickhardt and M. Nowostawski, "Imbalance measure and proactive channel rebalancing for the Lightning Network," in Proc. IEEE Conf. Blockchain Secur. Privac., May 2020, pp. 1–5.

How to Fund a Bidirectional Payment Channel? Accessed: Oct. 30, 2020.

[Online]. Available: https://www.idealists.illinois.edu/handle/2142/99319

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, Switzerland: Springer, 2015, pp. 3–18.

C. N. Cordi, "Simulating high-throughput cryptocurrency payment channel networks," M.S. thesis, 2017. [Online]. Available: https://www.ideals.illinois.edu/handle/2142/99319

[Online]. Available: https://medium.com/peter_t_visualizing-htlcs-and-the-lightning-networks-dirty-little-secret-eb9f6773a0

Lightning Network Specification. Accessed: Oct. 30, 2020. [Online]. Available: https://github.com/lightningnetwork/lightning-rfc

P. Hoenisch and I. Weber, "AODV–based routing for payment channel networks," in Blockchain–ICBC, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham, Switzerland: Springer, 2018, pp. 107–124.

S. M. Varma and S. T. Maguluri, "Throughput optimal routing in blockchain based payment systems," CoRR, vol. abs/2001.05299, pp. 1–16, Dec. 2020. [Online]. Available: https://arxiv.org/abs/2001.05299

Lightning Network Explorer. Accessed: Oct. 30, 2020. [Online]. Available: https://explorer.acinq.co

Atomic Multithreaded Payments. Accessed: Oct. 30, 2020. [Online]. Available: https://lists.linuxfoundation.org/pipermail/btc-law/2018-February/0015708.html

J. Jager. Multi-Path Payments in LND: Making Channel Balances Atomic Multipath Payments. Accessed: Oct. 30, 2020. [Online]. Available: https://lists.linuxfoundation.org/pipermail/btc-law/2018-February/0015708.html

[Online]. Available: https://github.com/lightninglabs/loop

New on Bitcoin’s Lightning Network: LND Adds Accounting Feature, C-Lightning Gets an Upgrade. Accessed: Oct. 30, 2020. [Online]. Available: https://www.coindesk.com/bitcoin-lightning-network-lnd-accounting-c-lightning

S. Bränzei, E. Segal-Halevi, and A. Zohar, "How to charge Lightning," CoRR, vol. abs/1712.0222, pp. 1–28, Dec. 2017. [Online]. Available: http://arxiv.org/abs/1712.0222

G. Avarikioti, R. Scheuner, and R. Wattenhofer, “Payment networks as creation games,” in Data Privacy Management, Cryptocurrencies and Blockchain Technology, C. Pérez-Soló, G. Navarro-Arribas, A. Biryukov, and J. García-Alfaro, Eds. Cham, Switzerland: Springer, 2018, pp. 195–210.

G. Di Stasi, S. Avallone, R. Canonico, and G. Vente, “Routing payments on the lightning network,” in Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Com Comput. Commun. (GreenCom), IEEE Cyber, Phys. Soc. Comput. (CPSCom), IEEE Smart Data (SmartData), Jul. 2018, pp. 1161–1170.

F. Engeln, H. Kopp, F. Kargl, and C. Weinhardt, “Towards an economic analysis of routing in payment channel networks,” in Proc. 1st Workshop Scalable Resilient Infrastructures Distrib. Ledgers (SERIAL). New York, NY, USA: Association for Computing Machinery, 2017, pp. 1–6, doi: 10.1145/3158243.3158286.

F. Béres, I. A. Seres, and A. A. Beniczú, “A cryptographic traffic analysis of Bitcoins lightning network,” Cryptocurrencies and Financial Markets, vol. 6, pp. 1–24, Nov. 2020. [Online]. Available: https://cryptocoinsystems.pub/pvub8bbyown

Real-Time Lightning Network Statistics. Accessed: Oct. 30, 2020. [Online]. Available: https://1ml.com/statistics

Watchtower Proposed Protocol. Accessed: Oct. 30, 2020. [Online]. Available: https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-December/0002350.html

G. Avarikioti, F. Laufenberg, J. Sliwiński, Y. Wang, and R. Wattenhofer, “Towards secure and efficient payment channels,” CoRR, vol. abs/1811.12740, pp. 1–18, Nov. 2018. [Online]. Available: http://arxiv.org/abs/1811.12740

C. Burchett, C. Want, and R. Wattenhofer, “Scalable funding of bitcoin micropayment channel networks,” Roy. Soc. Open Sci., vol. 5, no. 8, 2018, Art. no. 180089. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsos.180089

S. Dziembowski, L. Ecke, S. Faust, and D. Malinowski, “Perun: Virtual payment hubs over cryptocurrencies,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2019, pp. 106–123.
NIKOLAOS PAPADIS (Member, IEEE) received the Diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 2016, and the M.Sc. and M.Phil. degrees in electrical engineering from Yale University, New Haven, CT, USA, in 2017 and 2019, respectively. He is currently pursing the Ph.D. degree with Yale University and is a member of the Yale Institute for Network Science (YINS).

In the summer of 2019, he was a Research Intern with IBM Research. His research interests include blockchain and distributed ledgers, their networking, scalability, security and financial aspects, as well as payment networks and their efficient design and performance optimization. In the past, he has also performed research in the areas of recommendation systems and digital identity.

Papadis’s awards and honors include the IBM Ph.D. Fellowship, the Onassis Foundation Scholarship, the A. G. Leventis Foundation Scholarship, the Gerondellis Foundation Scholarship, the Limmat Stiftung Undergraduate Award, and the Christos Papakyriakopoulos Award.

LEANDROS TASSIULAS (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 1991.

He held faculty positions at Polytechnic University, New York, NY, USA, the University of Maryland, and the University of Thessaly, Greece. He is currently the John C. Malone Professor of Electrical Engineering with Yale University. His research interests include computer and communication networks, with an emphasis on fundamental mathematical models and algorithms of complex networks, architectures and protocols of wireless systems, sensor networks, novel internet architectures, and experimental platforms for network research. His most notable contributions include the max-weight scheduling algorithm and the back-pressure network control policy, opportunistic scheduling in wireless, the maximum lifetime approach for wireless network energy management, and the consideration of joint access control and antenna transmission management in multiple antenna wireless systems.

Dr. Tassiulas’s research has been recognized by several awards, including the IEEE Koji Kobayashi Computer and Communications Award, the Inaugural INFOCOM 2007 Achievement Award for fundamental contributions to resource allocation in communication networks, the INFOCOM 1994 and 2017 Best Paper Awards, the National Science Foundation (NSF) Research Initiation Award in 1992, the NSF CAREER Award in 1995, the Office of Naval Research Young Investigator Award in 1997, and the Bodoskasi Foundation Award in 1999.