CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

ORCA – Online Research @ Cardiff

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository:https://orca.cardiff.ac.uk/id/eprint/172779/

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Corcoran, Padraig and Lewis, Rhydian 2024. Path planning in payment channel networks with multi-party channels. Distributed Ledger Technologies: Research and Practice 10.1145/3702248

Publishers page: https://doi.org/10.1145/3702248

Please note:
Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See http://orca.cf.ac.uk/policies.html for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.

Check for updates

Path Planning in Payment Channel Networks with Multi-Party Channels

PADRAIG CORCORAN, School of Computer Science and Informatics, Cardiff University, Wales, UK.
RHYD LEWIS, School of Mathematics, Cardiff University, Wales, UK.

Payment Channel Networks (PCNs) provide a means to improve the scaling of cryptocurrency payments by allowing peers to make payments between themselves in an efficient manner. To make a payment between two peers, the task of path planning must first be performed to determine a path in the PCN connecting the peers in question before the payment in question is performed using this path. To date, existing research has focused on the problem of performing path planning in PCNs that contain two-party channels. It has been hypothesised that the scaling of PCNs could be further improved by considering the inclusion of multi-party channels that contain more than two peers. However, the problem of performing path planning in PCNs that contain multi-party channels has not yet been considered.

In this article, we address this gap in the research literature and propose a novel path planning method for PCNs containing multi-party channels. This method involves modelling the PCN with multi-party channels as a hypergraph, a type of graph where edges can contain two or more vertices, and using this model to solve the path planning problem in question. We prove that the proposed method is correct and computationally efficient. Furthermore, assuming path planning is performed using this method, we also present theoretical and experimental analyses that demonstrate the scaling benefits of using multi-party channels.

CCS Concepts: • **Mathematics of computing → Graph algorithms**; • **Networks → Network design and planning algorithms**; • **Computing methodologies → Distributed algorithms**.

Additional Key Words and Phrases: payment channel networks; path planning; lightning network

1 INTRODUCTION

A cryptocurrency is a decentralised digital currency that can be transferred between individuals without the involvement of a centralised authority. Some of the most successful cryptocurrencies include Bitcoin, Ethereum and Tether. The majority of cryptocurrencies are built using blockchain technology, which is a decentralised ledger that stores an immutable record of cryptocurrency payments. Due to the necessity of maintaining decentralisation and security, a blockchain will generally have limited scalability, measured in terms of the number of payments processed per unit time and the latency of these payments. Bitcoin, for example, can process approximately seven transactions per second and each transaction requires at least ten minutes for confirmation.

Several potential scaling solutions have been proposed to overcome the above scaling challenge including Payment Channel Networks (PCNs) and sidechains [10]. PCNs are one of the most developed and cited methods and are the focus of this article. A PCN consists of a network of payment channels where each payment channel contains a set of peers. The peers in a given channel can perform payments between themselves by updating their respective balances, where these updates are enforced using a smart contract. These individual payments are not confirmed in the blockchain and in turn, have low latency. When the peers in question have finished

Authors' addresses: Padraig Corcoran, corcoranp@cardiff.ac.uk, School of Computer Science and Informatics, Cardiff University, Cardiff, Wales, UK.; Rhyd Lewis, lewisr9@cardiff.ac.uk, School of Mathematics, Cardiff University, Cardiff, Wales, UK..

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s).
ACM 2769-6480/2024/10-ART
https://doi.org/10.1145/3702248

Just Accepted

transacting, the final mutually agreed balances are confirmed in the blockchain. If two peers do not share a payment channel, payments between these peers can still be performed if a useful payment path connecting the peers can be determined. For example, consider a payment channel that contains Alice and Bob, and a payment channel that contains Bob and Charlie. Alice can make a payment to Charlie by first making the payment to Bob who in turn makes a payment to Charlie. The intermediates in a given payment path charge a fee for forwarding the payment in question. We refer to the problem of determining a useful payment path between two peers in a PCN as the *path planning problem*. In this context, usefulness can be measured along several dimensions including the fees charged by, and the reliability of, the intermediate peers. The most successful implementation of a PCN is the Lightning Network (LN) which is built upon the Bitcoin cryptocurrency [1]. At the date of writing in April 2024, the LN contains over 13 thousand peers and over 52 thousand payment channels.

PCNs with channels containing more than two peers, which we refer to as multi-party channels, are a proposed extension of current PCNs that have the potential to provide further scaling. Irrespective of the corresponding number of peers, each channel typically generates two corresponding blockchain transactions: a channel opening transaction and a channel closing transaction. Therefore, the use of multi-party channels provides a means of increasing the number of peers in the PCN per blockchain transaction. It has also been hypothesised that multi-party channels have the potential to increase the connectivity of a PCN and in turn, improve the efficiency of making payments [6]. Creating and using multi-party channels is a complex problem and an active area of research. This is reflected in the fact that there currently exists no implementation of multi-party channels in the LN. Furthermore, currently, no Bitcoin Lightning Improvement Proposal (bLIP) on the topic of multi-party channels exists. A bLIP is a design document providing information to the LN community, or describing potential new features for the LN [13]. Existing research on the topic of multi-party channels has focused almost exclusively on the problems of creating, updating the state of and closing such channels in a useful manner [3, 6–8, 11, 20, 21]. In this context, usefulness can be measured along several dimensions including security and the complexity of the solution in question. The most cited of these works is the Eltoo (also called LN-Symmetry) protocol which has a significant probability of being implemented in the LN in the near future [6]. A consequence of the above focus is that existing research has only considered the problem of making payments between peers in the same multi-party channel. Indeed, the path planning problem of determining a payment path between peers in distinct multi-party channels has not been considered. In this article, we address this gap in the research literature and propose a solution to this problem. Given that the number of peers in an individual channel will generally be very small relative to the total number of peers in the corresponding PCN, this represents an important research contribution. The solution proposed involves modelling a PCN with multi-party channels as a hypergraph, which is a type of graph where edges can contain two or more vertices. This model is then used to solve the path planning problem in question.

The remainder of this paper is structured as follows. In Section 2 we review related works on the path planning problem. In Section 3 we formulate and present a solution to the problem of path planning in PCNs with multi-party channels. Since no PCN currently implements multi-party channels, this formulation and solution are defined as a generalisation of cases where channels only contain two peers. In Section 4 we present theoretical and experimental analyses that demonstrate the scaling benefits of using multi-party channels. Finally, in Section 5 we draw some conclusions from this work.

Just Accepted

2 RELATED WORKS

In this section, we briefly review existing solutions to the path planning problem. As discussed in the introduction, these works have exclusively focused on the case where the PCNs only contain two-party channels. A more in-depth review of these works can be found in the review article by [16]. Existing solutions to the path planning problem can broadly be categorised as centralised methods and decentralised methods. In centralised methods,

a single peer has knowledge of the current state of the network and uses this to independently compute the entire payment path. A use case, commonly known as source-based routing, for centralised methods is where the payment sender computes a payment path and uses onion routing to forward the payment along this path. Onion routing ensures that each peer along the path only learns the identity of the peer immediately before and the peer immediately after them in the path. They do not learn the payment sender, receiver or amount. Hence, these methods provide relatively good privacy. However, since the path planning task is performed exclusively by the payment sender, these methods require that the sender have significant computational resources and knowledge of the current state of the network. In the decentralised methods, each peer that the payment passes through learns the payment receiver and uses this information to independently decide the next peer to forward the payment to until it is eventually forwarded to the receiver. These methods do not require the payment sender to have significant computational resources or knowledge of the current state of the network. However, since each peer in the payment path learns the identity of the payment receiver, they provide relatively poor privacy. We now review methods in each of these categories.

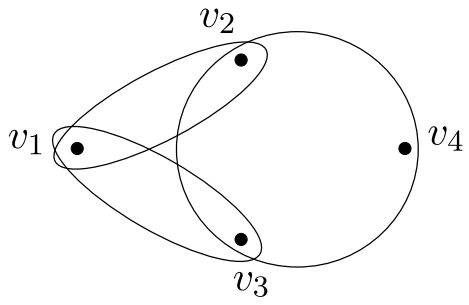
The following are some of the most highly cited centralised path planning methods. The path planning method most commonly used in practice is Dijkstra's algorithm, which is a fundamental path planning method for determining a lowest cost path in a network [1]. Several works have proposed centralised path planning methods that split the original payment into multiple smaller payments that are transferred using distinct payment paths [14, 19]. These methods have several potential benefits including the ability to perform larger payments in the presence of payment channels with less liquidity and improved privacy. Pickhardt and Richter [14] proposed a probabilistic path planning method that selects paths where the probability of payment failure due to insufficient channel liquidity is minimised. Yu et al. [22] proposed a path planning method that initially performs a network probing step to determine and reserve the liquidity of channels before attempting the payment in question.

The following are some of the most highly cited decentralised path planning methods. Roos et al. [18] proposed a path planning method entitled SpeedyMurmurs. This method uses an embedding to map each peer to coordinates in space such that peers that are closer in the original network are closer in the embedding space. Path planning is then performed using a greedy method where peers iteratively forward the payment in question to peers whose corresponding coordinates are closer to the receiver peer. Lin et al. [12] proposed a path planning method that splits the original payment into multiple smaller payments that are transferred using distinct payment paths. Zhang and Yang [23] proposed a path planning method that improves robustness to payment failures. This computes multiple independent payment paths and attempts the payment in question using each of these paths. Grunspan et al. [9] proposed a path planning method that applies an optimisation paradigm known as ant colony optimization. Prihodko et al. [15] proposed a path planning method entitled Flare. In this method, each peer maintains a routing table that contains payment paths to both peers in their local neighbourhood and a small subset of other peers known as beacons. To determine a payment path from a given sender to a given receiver, the routing tables corresponding to both peers are considered. If both routing tables contain a common peer, a payment path is constructed by combining payment paths to and from this peer. Lin et al. [12] proposed a path planning method that partitions the PCN into a set of regions and elects a peer in each region to be a corresponding beacon. Using this method, path planning is performed by forwarding the payment in question to a beacon. This beacon then forwards the payment to the receiver if the receiver is contained in their respective region. Otherwise, the beacon forwards the payment to the beacon corresponding to the region where the receiver is contained.

3 PATH PLANNING WITH MULTI-PARTY CHANNELS

In this section, we present the proposed method for performing path planning in PCNs with multi-party channels. As described above, path planning methods can broadly be categorised as centralised and decentralised methods.

Just Accepted



(a)

$$b(v_1, \{v_1, v_2\}) = 18 \quad b(v_2, \{v_2, v_3, v_4\}) = 11$$
$$b(v_2, \{v_1, v_2\}) = 11 \quad b(v_3, \{v_2, v_3, v_4\}) = 13$$
$$b(v_1, \{v_1, v_3\}) = 15 \quad b(v_4, \{v_2, v_3, v_4\}) = 14$$
$$b(v_3, \{v_1, v_3\}) = 12$$

(b)

$$f(v_1, v_2, \{v_1, v_2\}, a) = 1 \quad f(v_2, v_3, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_2, v_1, \{v_1, v_2\}, a) = 1 \quad f(v_3, v_2, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_1, v_3, \{v_1, v_3\}, a) = 2 \quad f(v_2, v_4, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_3, v_1, \{v_1, v_3\}, a) = 2 \quad f(v_4, v_2, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_3, v_4, \{v_2, v_3, v_4\}, a) = 1$$
$$f(v_4, v_3, \{v_2, v_3, v_4\}, a) = 1$$

(c)

Fig. 1. An PCN hypergraph model $G = (V, E)$ is displayed in (a) where $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3, v_4\}\}$. A corresponding balance map b, which specifies the balance of each peer in each channel, is defined in (b). A corresponding fee map f, which specifies the fees charged by peers for forwarding payments, is defined in (c).

Just Accepted

The proposed method belongs to the former category. The remainder of this section is structured as follows. In Section 3.1 we describe how a PCN with multi-party channels can be modelled as a hypergraph and how the corresponding path planning can be formulated with respect to this hypergraph. In Section 3.2 we describe the proposed path planning method for this model.

3.1 Model & Problem Formulation

In this section, we propose a model of a PCN containing multi-party channels. The LN currently does not implement multi-party channels. Therefore the proposed model is a generalisation of current LN two-party channels. We model a given PCN as a hypergraph $G = (V, E)$ where the vertex set V corresponds to a set of peers and the edge set E corresponds to a set of channels. A hypergraph is a generalisation of a graph where edges can contain two or more vertices [2]. Consequently, a graph is a special case of a hypergraph in which all edges contain exactly two vertices. Figure 1(a) displays an example PCN modelled as a hypergraph where $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3, v_4\}\}$. In the figure, the peers V are represented using filled circles and channels E are represented using closed curves.

Each payment channel in a given PCN has a fixed amount of liquidity. This liquidity is distributed among the peers in the channel where the amount distributed to a given peer is known as the balance of that peer in the channel. Given a PCN $G = (V, E)$, we model the balance of a given peer $v \in V$ in a given channel $e \in E$ as a balance map $b : V \times E \to \mathbb{Z}_{\geq}$. If a given peer is an element of several channels, it will have several corresponding balances. Consider again the PCN displayed in Figure 1(a). Figure 1(b) displays a balance map b corresponding to this PCN.

Let v_1 and v_2 be two peers in a given channel e and let $a \in \mathbb{Z}_{\geq}$ be a payment amount. Provided that the constraint $a \leq b(v_1, e)$ is satisfied, the vertex v_1 can make a payment to v_2 of amount a. This payment corresponds to a redistribution of the channel liquidity. After it has been executed, the balance $b(v_1, e)$ decreases by a while the balance $b(v_2, e)$ increases by a. To generalise a payment between peers in the same channel to a payment between peers in distinct channels we introduce the concept of a payment path. A payment path from v_1 to v_n is an alternating sequence of peers and channels $(v_1, e_1, v_2, e_2, v_3, \ldots, v_{n-1}, e_{n-1}, v_n)$ where $v_1, \ldots, v_n \in V$, $e_1, \ldots, e_{n-1} \in E$ and $\{v_i, v_{i+1}\} \subseteq e_i$. We refer to the number of channels in a given payment path as the length of the payment path. For example, a payment path from v_1 to v_4 in the hypergraph of Figure 1(a) is $(v_1, \{v_1, v_2\}, v_2, \{v_2, v_3, v_4\}, v_4)$.

Given a payment path $(v_1, e_1, v_2, e_2, v_3, \ldots, v_{n-1}, e_{n-1}, v_n)$ of length $n - 1$, if each channel e_i for $i = 1, \ldots, n - 1$ can forward a payment from v_i to v_{i+1} then, in turn, a payment from v_1 to v_n can be made. For example the payment path $(v_1, \{v_1, v_2\}, v_2, \{v_2, v_3, v_4\}, v_4)$ in the hypergraph displayed in Figure 1(a) can potentially be used to make a payment from v_1 to v_4. There may exist several distinct payment paths between a given pair of peers. For example, three distinct payment paths from v_1 to v_4 in the hypergraph displayed in Figure 1(a) are $(v_1, \{v_1, v_2\}, v_2, \{v_2, v_3, v_4\}, v_4)$, $(v_1, \{v_1, v_3\}, v_3, \{v_2, v_3, v_4\}, v_4)$ and $(v_1, \{v_1, v_2\}, v_2, \{v_2, v_3, v_4\}, v_3, \{v_2, v_3, v_4\}, v_4)$.

Peers in a given PCN charge a fee for forwarding a payment where the fee in question is proportional to and subtracted from the payment amount being forwarded. Let $f : V \times V \times E \times \mathbb{Z}_{\geq} \to \mathbb{Z}_{\geq}$ be a mapping from a payment forwarded by a given peer to the corresponding fee charged by that peer. For example, consider again the PCN displayed in Figure 1(a). The fee charged by v_2 for forwarding a payment to v_4 using channel $\{v_2, v_3, v_4\}$ of amount a equals $f(v_2, v_4, \{v_2, v_3, v_4\}, a)$. Since the fee charged for forwarding the payment is subtracted from the payment amount, the actual payment amount received by v_4 equals $a - f(v_2, v_4, \{v_2, v_3, v_4\}, a)$. Therefore, given a payment path $(v_1, e_1, v_2, e_2, v_3, \ldots, v_{n-1}, e_{n-1}, v_n)$, for a payment to be made using this path so that the amount a is received by v_n, a payment amount greater than or equal to a must be forwarded by v_1. Toward defining this amount, let a_i be the payment received by v_i for $i = 1, \ldots, n$. These amounts are defined by the following recurrence relation with the initial condition $a_n = a$:

$$a_{i-1} = a_i + f(v_{i-1}, v_i, e_{i-1}, a_i). \tag{1}$$

The total fees that peers in this payment path charge for forwarding the payment equals $a_1 - a$. We refer to this value as the corresponding payment path fee.

We define the path planning problem as follows. Given a hypergraph $G = (V, E)$ model of a PCN with multi-party channels, find a lowest fee payment path, if one exists, from a given source peer $s \in V$ to a given destination peer $t \in V$ such that a given payment amount $a \in \mathbb{Z}_{\geq}$ is received by t. To illustrate this problem, consider again the PCN displayed in Figure 1(a) and a path planning problem in which the source peer is v_1, the destination peer is v_4 and the payment amount is 1. Let the map f, which specifies the fees charged by vertices for forwarding payments, be that defined in Figure 1(c). Note that, in this example, the fees in question are independent of the payment amount forwarded, which is generally not the case. Furthermore, let us assume that all peers in all channels have a sufficient balance to forward a payment of any value. In this case, the lowest fee payment path equals $(v_1, \{v_1, v_3\}, v_3, \{v_2, v_3, v_4\}, v_4)$ and this payment path has a fee equal to 3 ($f(v_1, v_3, \{v_1, v_3\}, 2) + f(v_3, v_4, \{v_2, v_3, v_4\}, 1) = 2 + 1 = 3$).

In this work, we assume the existence of smart contract mechanisms that allow payments between peers in a PCN with multi-party channels to be made robustly and securely. Defining such mechanisms is beyond

Just Accepted

the scope of this article. However, this could in theory be achieved using a combination of two smart contract mechanisms in a similar manner to how the LN currently combines two mechanisms [1]. To allow payments between peers in the same channel, the mechanism currently used by the LN that involves all peers in the channel exchanging updated signed commitment and punishment transactions. This approach could potentially be extended for channels containing more than two peers. Alternatively, some of the proposed mechanisms described in the introduction to this article, such as Eltoo, could be used. To allow payments between peers in distinct channels, the mechanism currently used by the LN involving the use of a Hashed Timelock Contracts (HTLC). This approach could also potentially be extended. Broadly speaking, as the number of peers in a given channel increases, the corresponding cost and complexity of coordination between these peers also increases. Therefore, useful smart contract mechanisms must be used so that this increase in cost and complexity does not outweigh other benefits of using multi-party channels.

3.2 Path Planning

In this section, we propose a method that provides a solution to the path planning problem formulation presented above. This is a centralised method whereby a single peer computes the entire payment path, and contains the following two steps. In the first step, from a given PCN hypergraph model we derive a corresponding graph model such that lowest fee payment paths in the latter model correspond to low fee payment paths in the former. Recall that a graph model is a hypergraph model where each edge contains two vertices. In the second step, we apply a computationally efficient method for determining a lowest fee payment path in the graph model. We now describe each of these steps in turn.

Let $G = (V, E)$ be a given hypergraph model with a corresponding balance map $b : V \times E \to \mathbb{Z}_{\geq}$ and fee map $f : V \times V \times E \times \mathbb{Z}_{\geq} \to \mathbb{Z}_{\geq}$. We derive a graph $G' = (V', E')$ with corresponding balance map $b' : V' \times E' \to \mathbb{Z}_{\geq}$ and fee map $f' : V' \times V' \times E' \times \mathbb{Z}_{\geq} \to \mathbb{Z}_{\geq}$ using the following approach. Each peer $v \in V$ is first mapped using a bijective map to peer $v' \in V'$. Each hypergraph channel $e \in E$ is then mapped to a subset of channels in E' containing every unordered pair of peers in e. We refer to this mapping as δ and it is formally defined in Equation 2, where $2^{E'}$ is the power set (set of all subsets) of E'. This mapping is commonly referred to as a *clique expansion* in hypergraph research literature [5]. Let $\delta^{-1} : E' \to E$ be the inverse map of δ that maps each channel $e' \in E'$ to the channel $e \in E$ from which it was derived. For each $\{v_i, v_j\} \in E'$ we, in turn, assign $f'(v_i, v_j, \{v_i, v_j\}, a)$ to equal $f(v_i, v_j, \delta^{-1}(\{v_i, v_j\}), a)$, $f'(v_j, v_i, \{v_i, v_j\}, a)$ to equal $f(v_j, v_i, \delta^{-1}(\{v_i, v_j\}), a)$, $b'(v_i, \{v_i, v_j\})$ to equal $b(v_i, \delta^{-1}(\{v_i, v_j\}))$ and $b'(v_j, \{v_i, v_j\})$ to equal $b(v_j, \delta^{-1}(\{v_i, v_j\}))$. That is, each channel in the derived graph inherits the fee and balance properties from the channel in the hypergraph from which it was derived.

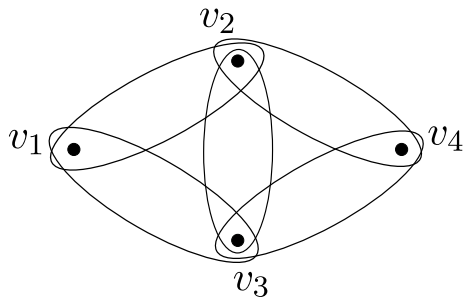
$$\begin{aligned} \delta : E &\to 2^{E'} \\ e &\mapsto \{\{v_i, v_j\} : v_i \in e, v_j \in e\} \end{aligned} \tag{2}$$

To demonstrate the relationship between the hypergraph and graph models, consider again the hypergraph G and corresponding maps b and f displayed in Figure 1. The derived graph G' and corresponding maps b' and f' are displayed in Figure 2. For each hypergraph channel $e \in E$, there will exist $\binom{|e|}{2} = \frac{1}{2}(|e| - 1)|e|$ derived channels in E'. In turn, the total number of channels in G' equals:

$$|E'| = \sum_{e \in E} \frac{1}{2}(|e| - 1)|e|. \tag{3}$$

Since the derived graph is a type of hypergraph, the definitions of a payment path and payment path fee presented above automatically generalise. In Theorem 3.1, we prove that the path planning problem in the hypergraph $G = (V, E)$ reduces to a path planning problem in the graph $G' = (V', E')$. This latter problem can

Just Accepted



(a)

$b'(v_1, \{v_1, v_2\}) = 18$ $b'(v_2, \{v_2, v_3\}) = 11$
$b'(v_2, \{v_1, v_2\}) = 11$ $b'(v_3, \{v_2, v_3\}) = 13$
$b'(v_1, \{v_1, v_3\}) = 15$ $b'(v_3, \{v_2, v_4\}) = 13$
$b'(v_3, \{v_1, v_3\}) = 12$ $b'(v_4, \{v_2, v_4\}) = 14$
$b'(v_2, \{v_3, v_4\}) = 11$
$b'(v_4, \{v_3, v_4\}) = 14$

(b)

$f'(v_1, v_2, \{v_1, v_2\}, a) = 1$ $f'(v_2, v_3, \{v_2, v_3\}, a) = 4$
$f'(v_2, v_1, \{v_1, v_2\}, a) = 1$ $f'(v_3, v_2, \{v_2, v_3\}, a) = 4$
$f'(v_1, v_3, \{v_1, v_3\}, a) = 2$ $f'(v_2, v_4, \{v_2, v_4\}, a) = 4$
$f'(v_3, v_1, \{v_1, v_3\}, a) = 2$ $f'(v_4, v_2, \{v_2, v_4\}, a) = 4$
$f'(v_3, v_4, \{v_3, v_4\}, a) = 1$
$f'(v_4, v_3, \{v_3, v_4\}, a) = 1$

(c)

Fig. 2. A PCN graph model $G' = (V', E')$ derived from the hypergraph model in Figure 1 is displayed in (a) where $V' = \{v_1, v_2, v_3, v_4\}$ and $E' = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}$. A corresponding balance map b', which specifies the balance of each peer in each channel, is defined in (b). A corresponding fee map f', which specifies the fees charged by peers for forwarding payments, is defined in (c).

then be solved in a correct and computationally efficient manner using Dijkstra's algorithm [1]. In this context, correctness means that the algorithm will find a lowest fee payment path if and only if one exists. Combining both of these results, we propose to solve the path planning in the hypergraph G by converting this hypergraph to the graph G' and solving the path planning problem in this graph. This approach is detailed in Algorithm 1. Given a hypergraph G and path planning problem in this graph as input, this approach first computes the corresponding derived graph G' (line 2). Next, a lowest fee payment path p' in G' is computed using the *payment_path* method, which implements Dijkstra's algorithm mentioned above (line 3). Next, the payment path p' in G' is mapped to a payment path p in G by applying the map δ to each edge in p' (line 4). Finally, the payment path p is returned (line 5).

To illustrate the above approach, consider again the hypergraph $G = (V, E)$ displayed in Figure 1 and the path planning problem where the source peer is $v_1 \in V$, the destination peer is $v_4 \in V$ and the payment amount is 1. This reduces to the path planning problem in the graph $G' = (V', E')$ displayed in Figure 2 where the source peer is $v_1 \in V'$, the destination peer is $v_4 \in V'$ and the payment amount is 1. The solution to this latter problem is the lowest fee path $(v_1, \{v_1, v_3\}, v_3\{v_3, v_4\}), v_4)$ in G'. This path has a fee equal to 3 ($f(v_1, v_3, \{v_1, v_3\}, 2) + f(v_3, v_4, \{v_3, v_4\}, 1) = 2 + 1$). By applying the map δ^{-1} to each channel in this path, it maps to the path $(v_1, \{v_1, v_3\}, v_3, \{v_2, v_3, v_4\}, v_4)$ in G. This is a lowest fee path and a solution to the former problem.

Theorem 3.1. *Let $G = (V, E)$ be a PCN hypergraph model with balance map $b : V \times E \to \mathbb{Z}_{\geq}$ and fee map $f : V \times V \times E \times \mathbb{Z}_{\geq} \to \mathbb{Z}_{\geq}$. Let $G' = (V', E')$ be the corresponding derived graph model with balance map $b' : V' \times E' \to \mathbb{Z}_{\geq}$ and fee map $f' : V' \times V' \times E' \times \mathbb{Z}_{\geq} \to \mathbb{Z}_{\geq}$. Consider the path planning problem in G of determining a lowest fee path from $s \in V$ to $t \in V$ such that the amount $a \in \mathbb{Z}_{\geq}$ is transferred to t. This reduces to the path planning problem in G' of determining a lowest fee path from $s \in V'$ to $t \in V'$ such that the amount $a \in \mathbb{Z}_{\geq}$ is*

Just Accepted

transferred to t. Specifically, let the payment path $p' = (v_1, e_1, v_2, e_2, v_3, \ldots, v_{n-1}, e_{n-1}, v_n)$ in G' be a solution to the latter problem. The payment path $p = (v_1, \delta^{-1}(e_1), v_2, \delta^{-1}(e_2), v_3, \ldots, v_{n-1}, \delta^{-1}(e_{n-1}), v_n)$ in G equals a solution to the former problem.

Proof. Let $p' = (v_1, e_1, v_2, e_2, v_3, \ldots, v_{n-1}, e_{n-1}, v_n)$ be a payment path in G'. This payment path maps to a payment path $p = (v_1, \delta^{-1}(e_1), v_2, \delta^{-1}(e_2), v_3, \ldots, v_{n-1}, \delta^{-1}(e_{n-1}), v_n)$ in G where this map is a bijection. That is, each payment path in G' maps to a single payment path in G and vice versa. By definition, $b'(v_i, e) = b(v_i, \delta^{-1}(e_i))$ for $i = 1, \ldots, n$. Furthermore, $f'(v_i, v_{i+1}, e_i, a) = f(v_i, v_{i+1}, \delta^{-1}(e_i), a)$ for $i = 1, \ldots, n-1$ and $a \in \mathbb{Z}_{\geq}$. It follows that the feasibility and fee for making a payment using p' equals the feasibility and fee respectively for making a payment using p. Hence, p' is a lowest fee payment path in G' if and only if p is a lowest fee payment path in G. □

Algorithm 1: Path Planning in PCN Hypergraph

Input: A PCN hypergraph $G = (V, E)$, a payment source $s \in V$, a payment destination $t \in V$ and payment amount $a \in \mathbb{Z}^{>}$ to be received by t.

Output: A lowest fee payment path in G from $s \in V$ to $t \in V$ for the amount $a \in \mathbb{Z}^{>}$ to be received by t. If no such path exists, no path is returned.

begin
  Compute $G' = (V', E')$. `// G' is the graph derived from the hypergraph G.`
  Compute $p' = (v_1, e_1, v_2, e_2, v_3, \ldots, v_{n-1}, e_{n-1}, v_n) = \text{payment\_path}(G', s, t, a)$.
  Compute $p = (v_1, \delta^{-1}(e_1), v_2, \delta^{-1}(e_2), v_3, \ldots, v_{n-1}, \delta^{-1}(e_{n-1}), v_n)$.
  Return p.

We now prove the correctness and computational complexity of Algorithm 1.

Theorem 3.2. *Algorithm 1 for solving the path planning in the hypergraph $G = (V, E)$ is correct and has computational complexity of $O(|V|^2 + |V| \log |V|)$.*

Proof. As proved in Theorem 3.1, the path planning in the hypergraph $G = (V, E)$ can be reduced to a path planning problem in the graph $G' = (V', E')$. This latter problem can, in turn, be solved using Dijkstra's algorithm [1]. This algorithm is correct and has a computational complexity of $O(|E'| + |V'| \log |V'|)$ when a Fibonacci heap data structure is used [4]. Recall that, $|V'| = |V|$. Using Equation 3, $|E'|$ is approximately equal to $|V|^2$ ($\sum_{e \in E} \frac{1}{2}(|e| - 1)|e| \approx \sum_{e \in E} |e|^2 \approx \sum_{e \in E} |V|^2 \approx |V|^2$). Note that, this is a worst-case approximation where edge $e \in E$ contains every element in the set V; that is $|e| = |V|$. Using these facts, the computational complexity of applying Dijkstra's algorithm with respect to the hypergraph $G = (V, E)$ equals $O(|V|^2 + |V| \log |V|)$. □

4 RESULTS & ANALYSIS

In the previous section, we presented a method for performing path planning in PCNs with multi-party channels that can be used as a platform for performing payments. We mentioned in Section 3.1 that, as the number of peers in a given channel increases, the corresponding cost and complexity of performing coordination between these peers also increases. Therefore, a useful smart contract mechanism must be used such that this increase in cost and complexity does not outweigh the benefits of using multi-party channels. In the following two subsections, we present a theoretical analysis followed by an experimental analysis of the benefits of using multi-party channels containing two or more peers in comparsion to using two-party channels containing two peers. The later experimental analysis involves the simulation of payments on a snapshot of the LN.

Just Accepted

4.1 Theoretical Analysis

It was proved in Theorem 3.1 that, path planning in the hypergraph model $G = (V, E)$ of a PCN with multi-party channels is equivalent to path planning in a derived graph model $G' = (V', E')$ with two-party channels. This provides an equivalence, with respect to the ability to perform payments, between the model of a PCN with multi-party channels and the model of a PCN with two-party channels. The analysis presented in this section uses this equivalence to quantify the relative benefits of using multi-party channels.

We first consider the relative number of blockchain transactions generated. Since blockchain transactions are significantly more expensive than payments made using a PCN, it is useful to minimise their number. Let us assume that each payment channel generates two blockchain transactions, corresponding to channel opening and closing transactions. Consider a multi-party channel $e \in E$ in the hypergraph G that contains n peers. This channel generates two blockchain transactions. It maps to the set of two-party channels $\delta(e)$ in the graph G' containing $\binom{n}{2} = \frac{1}{2}(n-1)n$ elements. This set of channels therefore generates a total of $2 \times \frac{1}{2}(n-1)n \approx n^2$ blockchain transactions. To illustrate this result, consider again the PCNs with multi-party and corresponding two-party channels displayed in Figures 1 and 2 respectively. The multi-party channel $\{v_2, v_3, v_4\}$ generates two blockchain transactions. The corresponding three two-party channels $\{v_2, v_3\}$, $\{v_2, v_4\}$ and $\{v_3, v_4\}$ generate six blockchain transactions. In summary, a single multi-party channel generates a constant number of blockchain transactions. This single multi-party channel is equivalent to a set of two-party channels that generate a number of blockchain transactions quadratic in the number of peers.

As described in Section 3.1, a given payment channel will have a fixed amount of liquidity, where this liquidity is distributed among the channel peers. Liquidity is a finite resource that, when locked in a given channel, cannot be simultaneously used for other financial activities. Therefore, it is useful to minimise the amount of liquidity within a PCN. Consider a multi-party channel $e \in E$ in the hypergraph G that contains n peers. The total liquidity in this channel equals

$$\sum_{v \in e} b(v, e). \tag{4}$$

The total liquidity in the corresponding set of two-party channels $\delta(e)$ in the graph G' equals

$$\begin{aligned} &\sum_{e' \in \delta(e)} \sum_{v \in e'} b'(v, e') \\ =&\sum_{e' \in \delta(e)} \sum_{v \in e'} b(v, \delta^{-1}(e')) \\ =&\sum_{e' \in \delta(e)} \sum_{v \in e'} b(v, e) \\ =&(n-1) \sum_{v \in e} b(v, e). \end{aligned} \tag{5}$$

The final equality follows from the fact that each peer $v \in e$ appears in $n-1$ of the set of two-party channels $\delta(e)$. Therefore, a single multi-party channel with a given amount of liquidity is equivalent to a set of two-party channels with $n-1$ times the amount of liquidity. To illustrate this result, consider again the PCNs with multi-party and corresponding two-party channels displayed in Figures 1 and 2 respectively. The multi-party channel $\{v_2, v_3, v_4\}$ has a total liquidity equal to $b(v_2, \{v_2, v_3, v_4\}) + b(v_3, \{v_2, v_3, v_4\}) + b(v_4, \{v_2, v_3, v_4\}) = 11 + 13 + 14 = 38$. On the other hand, the total liquidity of the set of channels $\delta(\{v_2, v_3, v_4\}) = \{\{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}$ equals $11 + 13 + 13 + 14 + 11 + 14 = 76$. In summary, a single multi-party channel is equivalent to a set of two-party channels with significantly more liquidity.

Just Accepted

4.2 Experimental Analysis

In the previous section, we demonstrated the theoretical benefits of using multi-party channels. In this section, we demonstrate empirically that these theoretical benefits result in an improved ability to perform payments. Specifically, we first simulate the expansion of two-party channels to multi-party channels in a snapshot of the LN. We then simulate payment attempts on the original and expanded LNs to empirically evaluate the relative performance. The remainder of this section is structured as follows. In Section 4.2.1 we describe the proposed method for simulating the expansion of two-party channels to multi-party channels. In Section 4.2.2 we describe the LN snapshot used in our evaluation and the method used for generating a set of simulated payment attempts. Finally, in Section 4.2.3 we evaluate the relative performance of these payment attempts on both the original snapshot and its expanded form.

4.2.1 Expanding Payment Channels. Let $G = (V, E)$ be a given hypergraph model of the LN, and let $e \in E$ be a given two-party channel. Algorithm 2 outlines a proposed method for expanding this two-party channel to a multi-party channel containing $n > 2$ peers where the amount of liquidity in the original and expanded channels is equal. This method first assigns $\{v_1, v_2\}$ to be the set of peers in e (line 2). Next, a set β containing $n - 2$ distinct peers is selected uniformly at random from the set $V - e$ (line 3). The channel e is then expanded to be a channel containing n peers by including the set β of peers (line 4). For each peer $v \in \beta$, the corresponding balance is set to equal 0 (lines 5 and 6). Consequently, the original and expanding channels have an equal amount of liquidity. Note that, since each peer $v \in \beta$ initially has a zero balance in the channel, these peers initially can only receive payments from other peers in the channel. These peers can not send payments to other peers in the channel unless they first receive a payment from one of these peers and, in turn, have a non-zero balance. Finally, we extrapolate the fee policy for the original two-party channel to the now multi-party channel as follows. For each peer $v \in \beta$, the fee policy from v_1 to v is set to equal that from v_1 to v_2, and the fee policy from v_2 to v is set to equal that from v_2 to v_1 (lines 7 to 9). For each peer $v \in \beta$, the fee policy from v to each other peer in the channel is uniformly at random set to equal the fee policy from v_1 to v_2 or the fee policy from v_2 to v_1 (lines 10 to 12).

To illustrate the above method consider again the PCN hypergraph model displayed in Figure 1. The result of applying this method to expand the channel $\{v_1, v_3\}$ to a multi-party channel containing three peers is displayed in Figure 3. In this case, the channel in question has been expanded to the multi-party channel $\{v_1, v_3, v_4\}$. To illustrate the benefits of expanding this channel, consider again the path planning problem where the source peer is $v_1 \in V$, the destination peer is $v_4 \in V$ and the payment amount is 1. Recall that, in the original hypergraph, the lowest fee path equals $(v_1, \{v_1, v_3\}, v_3, \{v_2, v_3, v_4\}, v_4)$ and this path has a fee equal to 3. On the other hand, in the hypergraph with the expanded channel, the lowest fee path equals $(v_1, \{v_1, v_3, v_4\}, v_4)$ and this path has a fee equal to 2.

4.2.2 LN Snapshot and Payment Attempts. We obtained a snapshot of the LN on 30 April 2024 using the Lightning Network Daemon (LND) peer implementation by Lightning Labs[1]. Some channels in the snapshot did not have a payment forwarding fee policy and could not be used to forward payments. Therefore, we removed all such channels from the snapshot. The snapshot also only contained information about public channels that are shared amongst peers. Therefore, the snapshot contained a large proportion of isolated peers that may be connected to other peers using only private channels. We also removed all such peers from the snapshot. Following the removal of channels without a fee policy and isolated peers, the LN snapshot contained 3,795 peers and 11,467 channels. On the date the above snapshot was obtained, 1 Satoshi (the atomic unit of Bitcoin currency used by the LN) had an approximate value of 0.0005 Great British Pounds (GBP). The channels in the snapshot contained varying amounts of liquidity. The mean and median channel liquidity were 9,451,547 and 3,000,000 Satoshis respectively. The 10th and 90th percentiles of channel liquidities were 495,000 and 16,700,000 Satoshis respectively. Channel

[1] https://docs.lightning.engineering/

Just Accepted

Algorithm 2: Expanding a two-party channel to a multi-party channel

Input: A PCN hypergraph $G = (V, E)$, a two-party channel $e \in E$ and the number of peers $n > 2$ to expand this channel to.

Output: A PCN hypergraph $G = (V, E)$ where $e \in E$ has been extended to a multi-party channel containing n peers.

begin
  Set $\{v_1, v_2\} = e$.
  Set β to equal $n - 2$ vertices sampled without replacement from $V - e$.
  Set $e = e \cup \beta$.
  forall $v \in \beta$ **do**
    Set $b(v, e) = 0$.
  forall $v \in \beta$ **do**
    Set $f(v_1, v, e, a) = f(v_1, v_2, e, a)$.
    Set $f(v_2, v, e, a) = f(v_2, v_1, e, a)$.
  forall $v \in \beta$ **do**
    forall $v' \in e - v$ **do**
      Set $f(v, v', e, a)$ to equal $f(v_1, v_2, e, a)$ or $f(v_2, v_1, e, a)$ with equal probability.

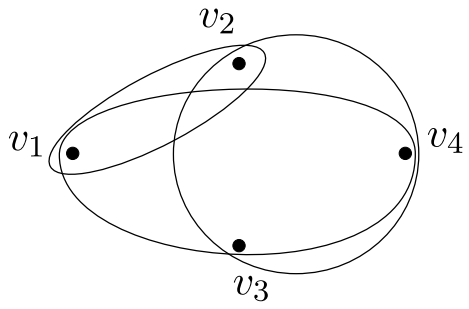
balance information is not shared in the LN for privacy reasons. Therefore, in our simulations we assumed that all channels were balanced such that the liquidity in question was evenly distributed amongst the corresponding peers.

Given the above LN snapshot, we generated a corresponding set of 10, 000 payment attempts using the following approach. A payment attempt is defined by the following two parameters: (1) a payment amount and (2) a payment source and destination pair. For privacy reasons, information regarding payment attempts is not shared by peers in the LN. However, a peer through which a payment passes will learn the payment amount but not the source or destination in question. The payment source and destination are both obfuscated by the use of onion routing, which encrypts this information.

The cryptocurrency company River aggregated payment amount data from several LN peers and published corresponding summary statistics for the period January to August in 2023 [17]. In August 2023, the average payment amount was 44,700 Satoshis with the majority during the entire eight month period being less than 1,000,000 Satoshis. Given this, we assumed that payment amounts are independently sampled uniformly at random from the interval $[1, 1000000]$. Given the lack of data regarding the source and destination of payments, we assumed that payment source and destination pairs are independently sampled uniformly at random from the set of all distinct pairs of peers. Using the above assumptions, we independently sampled uniformly at random 10, 000 payment attempts.

4.2.3 Simulation of Payment Attempts. We simulated the 10, 000 payment attempts described in the previous section on the LN snapshot plus a series of expanded versions of this snapshot. Each expanded version is a function of the following two parameters. The first parameter is the subset of two-party channels that are expanded. We specified this parameter by first defining the size of this subset and in turn selecting the elements in this set uniformly at random. In our analysis, we considered subset sizes of 0 (no channels), 1147 (10% of all channels), 2293 (20% of all channels), 3440 (30% of all channels), 4587 (40% of all channels) and 5734 (50% of all channels). The second parameter is the size of the expanded channels. In our analysis, we considered expanded channel

Just Accepted



(a)

$$b(v_1, \{v_1, v_2\}) = 18 \quad b(v_2, \{v_2, v_3, v_4\}) = 11$$
$$b(v_2, \{v_1, v_2\}) = 11 \quad b(v_3, \{v_2, v_3, v_4\}) = 13$$
$$b(v_1, \{v_1, v_3, v_4\}) = 15 \quad b(v_4, \{v_2, v_3, v_4\}) = 14$$
$$b(v_3, \{v_1, v_3, v_4\}) = 12$$
$$b(v_4, \{v_1, v_3, v_4\}) = 0$$

(b)

$$f(v_1, v_2, \{v_1, v_2\}, a) = 1 \quad f(v_2, v_3, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_2, v_1, \{v_1, v_1\}, a) = 1 \quad f(v_3, v_2, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_1, v_3, \{v_1, v_3, v_4\}, a) = 2 \quad f(v_2, v_4, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_3, v_1, \{v_1, v_3, v_4\}, a) = 2 \quad f(v_4, v_2, \{v_2, v_3, v_4\}, a) = 4$$
$$f(v_1, v_4, \{v_1, v_3, v_4\}, a) = 2 \quad f(v_3, v_4, \{v_2, v_3, v_4\}, a) = 1$$
$$f(v_3, v_4, \{v_1, v_3, v_4\}, a) = 2 \quad f(v_4, v_3, \{v_2, v_3, v_4\}, a) = 1$$
$$f(v_4, v_1, \{v_1, v_3, v_4\}, a) = 2$$
$$f(v_4, v_3, \{v_1, v_3, v_4\}, a) = 2$$

(c)

Fig. 3. An PCN hypergraph model $G = (V, E)$ is displayed in (a) where $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\}\}$. This model corresponds to an expansion of the model displayed in Figure 1 where the channel $\{v_1, v_3\}$ has been expanded into the channel $\{v_1, v_3, v_4\}$.

sizes of 3, 4, 5, 6, 7, 8 and 9. To measure the performance of simulated attempts on the original LN snapshot plus its expanded versions we used the following three metrics. The first metric is the number of the 10,000 payment attempts that were successful. This is a useful metric given that the principal purpose of a PCN is the support payments. The second metric is the median payment path length of the set of successful payment attempts. This is a useful metric given shorter paths contain fewer channels and therefore have a higher probability of being successful. The final metric is the median payment path fee of the set of successful payment attempts. This is a useful metric given that an important aim of PCNs is to provide lower fee payments relative to payments confirmed in the blockchain. In the latter two metrics, the median value was used because it is a robust measure of central tendency. Using a robust measure is important in the context of payment fees because a small percentage of LN channels, and in turn corresponding payments that contain these channels, have extremely high fees. Due to their high fees, in practice, the payments in question may not be performed. For each number of channels and size of channel combination, Table 1 displays the three corresponding metric values. We observed the following properties. Firstly, as the number and/or size of channels increased, both the number of successful payments

Just Accepted

Number / Size	3	4	5	6	7	8	9
0 (0%)	4977, 6, 52	4977, 6, 52	4977, 6, 52	4977, 6, 52	4977, 6, 52	4977, 6, 52	4977, 6, 52
1147 (10%)	5281, 6, 42	5590, 7, 35	5739, 6, 32	5902, 7, 25	6076, 7, 22	6191, 6, 21	6309, 7, 20
2293 (20%)	5547, 7, 33	5970, 7, 22	6219, 7 ,15	6425, 7, 12	6560, 7, 10	6656, 6, 9	6723, 6, 9
3440 (30%)	5804, 7, 25	6222, 7, 13	6529, 7, 10	6691, 7, 8	6770, 7, 6	6808, 6, 6	6841, 6, 5
4587 (40%)	5989, 7, 20	6440, 7, 10	6683, 7, 8	6795, 7, 6	6861, 7, 5	6856, 7, 4	6882, 7, 4
5734 (50%)	6157, 7, 16	6603, 7, 8	6802, 7, 6	6852, 7, 5	6894, 8, 4	6877, 7, 4	6893, 7, 4

Table 1. For each number of channels and size of channel combination, this table displays a comma separated list of the corresponding number of successful payment attempts, the median payment path length of these attempts and the median payment path fee of these attempts.

Just Accepted

increased and the medium payment path fee decreased. Secondly, as the number and/or size of channels increased, the medium payment path length remained approximately the same. This property can be attributed to the fact that the path planning method used optimises payment paths exclusively with respect to fees and ignores lengths. These observed properties empirically demonstrate the benefits of multi-party channels.

5 CONCLUSIONS

It has long been hypothesised that the construction of PCNs with multi-party channels containing more than two peers has the potential to improve payment scaling. Until now the problem of path planning in PCNs with multi-party channels had not been considered, despite the necessity of path planning for performing payments. In this article, we overcome this challenge and proposed a correct and computationally efficient path planning method for PCNs with multi-party channels. We in turn have presented theoretical and experimental analysis that demonstrates the scaling benefits of using multi-party channels.

It is hoped that the above findings will accelerate the development and deployment of PCNs with multi-party channels. However, there exist several other challenges that must be first overcome. Path planning is just one of several methods or algorithms that coordinate within existing implementations of PCNs such as the LN to ensure that the PCN in question functions robustly and securely. Such algorithms include onion routing and channel liquidity management algorithms. These algorithms would need to be generalised to the case of PCNs containing multi-party channels before such PCNs could be used robustly and securely.

ACKNOWLEDGEMENTS

This work was supported by the Security, Crime and Intelligence Innovation Institute at Cardiff University through their Kickstarter Funding scheme.

REFERENCES

[1] Andreas M Antonopoulos, Olaoluwa Osuntokun, and René Pickhardt. 2021. *Mastering the Lightning Network.* O'Reilly Media.
[2] Alain Bretto. 2013. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer* 1 (2013).
[3] Yanjiao Chen, Xuxian Li, Jian Zhang, and Hongliang Bi. 2022. Multi-party payment channel network based on smart contract. *IEEE Transactions on Network and Service Management* 19, 4 (2022), 4847–4857.
[4] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms.* MIT press.
[5] Qionghai Dai and Yue Gao. 2023. Mathematical Foundations of Hypergraph. In *Hypergraph Computation.* Springer, 19–40.

[6] Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. 2018. eltoo: A simple layer2 protocol for bitcoin. *White Paper* (2018).
[7] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. 2019. Perun: Virtual payment hubs over cryptocurrencies. In *IEEE Symposium on Security and Privacy*. 106–123.
[8] Zhonghui Ge, Yi Zhang, Yu Long, and Dawu Gu. 2023. Magma: Robust and flexible multi-party payment channel. *IEEE Transactions on Dependable and Secure Computing* 20, 6 (2023), 5024–5042.
[9] Cyril Grunspan, Gabriel Lehéricy, and Ricardo Pérez-Marco. 2020. Ant routing scalability for the lightning network. *arXiv preprint arXiv:2002.01374* (2020).
[10] Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. 2020. Scaling blockchains: A comprehensive survey. *IEEE Access* 8 (2020), 125244–125262.
[11] Longxia Huang, Hao Lei, and Liangmin Wang. 2024. MPC+: Secure, Compatible and Efficient Off-Blockchain Multi-Node Payment Channel. *IEEE Transactions on Network and Service Management* (2024).
[12] Changting Lin, Ning Ma, Xun Wang, and Jianhai Chen. 2020. Rapido: Scaling blockchain with multi-path payment channels. *Neurocomputing* 406 (2020), 322–332.
[13] Lightning Network. 2024. Bitcoin Lightning Improvement Proposal. https://github.com/lightning/blips. [Online; accessed 5-April-2024].
[14] Rene Pickhardt and Stefan Richter. 2021. Optimally reliable & cheap payment flows on the lightning network. *arXiv preprint arXiv:2107.05322* (2021).
[15] Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy, and Olaoluwa Osuntokun. 2016. Flare: An approach to routing in lightning network. *White Paper* 144 (2016).
[16] Gabriel Antonio F Rebello, Gustavo F Camilo, Lucas Airam C de Souza, Maria Potop-Butucaru, Marcelo Dias de Amorim, Miguel Elias M Campista, and Luís Henrique MK Costa. 2024. A survey on blockchain scalability: From hardware to layer-two protocols. *IEEE Communications Surveys & Tutorials* (2024).
[17] River. 2023. *Research Report: The Lightning Network Grew by 1212% in 2 Years, Why It's important to Pay Attention.* Technical Report. River.
[18] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. 2017. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748* (2017).
[19] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathleen Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Giulia Fanti, and Mohammad Alizadeh. 2020. High throughput cryptocurrency routing in payment channel networks. In *USENIX Symposium on Networked Systems Design and Implementation*.
[20] Ke Xiao, Jiayang Li, Yunhua He, Xu Wang, and Chao Wang. 2024. A secure multi-party payment channel on-chain and off-chain supervisable scheme. *Future Generation Computer Systems* 154 (2024), 330–343.
[21] Yongjie Ye, Zhifeng Ren, Xiapu Luo, Jingjing Zhang, and Weigang Wu. 2021. Garou: An efficient and secure off-blockchain multi-party payment hub. *IEEE Transactions on Network and Service Management* 18, 4 (2021), 4450–4461.
[22] Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari, Dejun Yang, and Jian Tang. 2018. CoinExpress: A fast payment routing mechanism in blockchain-based payment channel networks. In *27th international conference on computer communication and networks*. IEEE, 1–9.
[23] Yuhui Zhang and Dejun Yang. 2021. Robustpay+: Robust payment routing with approximation guarantee in blockchain-based payment channel networks. *IEEE/ACM Transactions on Networking* 29, 4 (2021), 1676–1686.

Received 5 July 2024; revised 5 July 2024; accepted 1 October 2024

Just Accepted