Demand Matrix Optimization for Offchain Payments in Blockchain

Julia Khamis
Technion

Stefan Schmid
University of Vienna

Ori Rottenstreich
Technion

***Abstract*—Offchain networks are dominant as a solution to the scalability problem of blockchain systems, allowing users to perform transactions without their recording on the chain. Offchain networks consist of channels connecting pairs of users with frequent transactions. Transactions between users without a direct channel are also supported through paths of multiple channels but involve payment of fees to intermediate users. A set of pending transactions (the *demand*) is typically served one by one. This paper proposes a novel approach to reduce the number of offchain transactions and their corresponding fees while preserving the requested impact on user accounts, by performing optimizations directly on the demand matrix. We contribute an analytical characterization of the properties of such demand matrix optimizations, present efficient algorithms and report on an empirical evaluation, demonstrating the effectiveness of the approach.**

I. Introduction

Cryptocurrencies are digital currencies that use cryptographic functions to conduct financial transactions (payments). Payments are recorded on a digital public ledger, also called a transaction blockchain, enabling decentralization, transparency, and immutability [1]. Cryptocurrencies have a common main problem: *scalability* – the ability to cope with the influx of a large number of payments at a time. This refers to the rate limitation of blockchains in processing payments since typically, every payment is recorded into the chain. While Visa and Mastercard can handle thousands of payments per second, Bitcoin's and Ethereum's maximum rates are roughly 7 and 15 payments per second, respectively [2].

Offchain networks are a leading solution for the scalability problem [3]. In recent years offchain networks such as Bitcoin Lightning [4], Ethereum Raiden [5] and XRP Ripple [6] have been implemented and evolved. While there are different types of offchain networks, they typically share the same conception: The ability to take payment operations outside of the blockchain. Participants can use payment channels to perform multiple payments with each other, without the need to commit every payment to the blockchain, as long as the accumulated sum of transferred tokens (units of the cryptocurrency) in each channel is within some predefined bounds. The blockchain is only involved when the participants create and close a payment channel, when they disagree on the results, or upon failing to serve a payment.

Two users u_1 and u_2 in the offchain network can establish a bidirectional payment channel used to serve payments. To open a such a payment channel, u_1 and u_2 deposit the amount of tokens they want to trade, sending a payment on chain. As long as the payment channel is open this amount of tokens is locked on it. The channel is characterized with a pair of non-negative capacities $(c_{(u_1,u_2)}, c_{(u_2,u_1)})$, which specify the channel token values owned by u_1 and u_2, respectively. The distribution of the locked value among the users of a channel changes based on their transactions, but the sum of the values remains constant as long as on-chain payments are not involved.

When modelling users as nodes and payment channels as edges, we obtain the *offchain topology graph*. The offchain network allows two participants without a direct channel between them to send payments via a multi-hop path. For example, assume that user u_1 wants to pay user u_2; while they do not have a direct channel between them, they both have a direct channel with user u_3. Then, u_1 can use the path $u_1 \rightarrow u_3 \rightarrow u_2$ through u_3 to pay u_2 with two offchain transactions. When routing a payment through a channel, the channel's capacity changes according to the tokens transfer. Most of previous works on offchain networks focus on optimizing the routing of the payment, improving the channels capacity balancing [7, 8] or enhancing privacy and security [9]–[11].

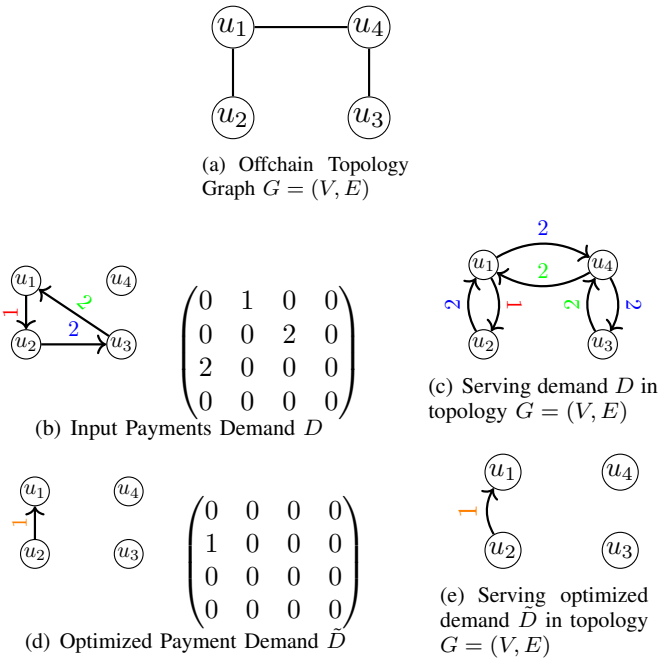
This paper considers a novel and orthogonal approach: rather than optimizing the routing of payments, we consider the optimization of the demand, that is, the payment matrix itself. We will show that such demand matrix optimizations can reduce costs significantly. In this paper, inspired by Bitcoin Lightning offchain network [4] we consider a typical fee structure for the use of a payment channel to serve a transaction: The fee for a transaction of value v is $\Lambda + v \cdot \lambda$, where Λ is a base fee and λ is a relative cost per unit of the crytocurrency, implying a fee proportional to the transaction value.

We study two main questions:

(1) Given the demand matrix describing the to-be-performed payments of the users, how to optimize the matrix to reduce fees while preserving its overall effect?

(2) Given an offchain network topology and the users payments' demand matrix, how to jointly optimize serving the transactions on this particular topology to minimize fees?

An illustration is given in Fig. 1 with four users $u_1, \ldots, u_4$; First, Fig. 1(a) shows a topology graph of an offchain network with three payment channels. Fig. 1(b) describes three payment requests that have to be served. There are a total of $1 + 2 + 2 = 5$ tokens that need to be moved, one token sent by u_1 to u_2, two tokens from u_2 to u_3 and two others from u_3 to u_1; in this scenario u_4 is not paying or receiving any tokens. This is illustrated by the directed graph



(a) Offchain Topology Graph $G = (V, E)$

(b) Input Payments Demand D

(c) Serving demand D in topology $G = (V, E)$

(d) Optimized Payment Demand $\tilde{D}$

(e) Serving optimized demand $\tilde{D}$ in topology $G = (V, E)$

Fig. 1: Payment demand optimization: Offchain channel topology in Fig. 1(a) and input payment demand in Fig. 1(b). Fig. 1(c) illustrates serving the demand in the topology without optimizations with six transactions. Fig. 1(d) shows the optimized equivalent payment demand and Fig. 1(e) its serving with only one transaction.

and corresponding demand matrix. The transactions required to serve all the payments represented in Fig. 1(b) on the topology from Fig. 1(a), appear in Fig. 1(c) such that a total of six transactions are needed. Note that a payment between users without a direct payment channel requires more than a single transaction. Fig. 1(d) describes an equivalent set of payments that has a single payment from u_2 to u_1. It implies the same impact on the accounts balances as the payments from Fig. 1(b). This payment can be served by a single transaction on the offchain network as shown in Fig. 1(e), which can be exploited to lower the cost without affecting the overall outcome (e.g., the balance of the user accounts). This illustrates the potential simplification and cost savings on serving a demand of payments.

Contributions: We make the following contributions:

(i) We initiate the study of a novel optimization opportunity for reducing costs in serving payments in offchain networks, by changing the demand matrix. We show that this optimization problem boils down to finding an equivalent demand matrix of low weight and computing an equivalent demand matrix that can be served in a low cost within a given topology.

(ii) We present an analytical characterization of the properties of demand matrix optimizations.

(iii) We present efficient solutions to these optimization problems for common scenarios, providing formal guarantees.

(iv) We report on an empirical evaluation of the potential benefits of such optimizations, using actual network data.

II. Related Work

Different offchain networks exist in the blockchain world like Lightening [4, 12] for Bitcoin, Raiden [5] for Ethereum, and the TeeChain network [13] that executes payments asynchronously with respect to the underlying blockchain. Perun [14] is an offchain channel system that offers an efficient method for connecting channels, instead of routing payments over multiple channels. Perun uses a technique called virtual payment channels that avoids the involvement of the intermediary for each payment, etc.

Previous works on offchain networks mostly focus on routing schemes for payment transactions. For example, Flash [8] makes a distinction between "mice payments", small frequent payments, and "elephant payments", large rare payment. To obtain a balance between high throughput and probing overhead. Flash routes mice payments by choosing randomly static paths from the routing table. Spider [7] focuses on improving the number and volume of successfully routed payments on offchain, by keeping payment channels balanced. Spider packetizes payments and uses a multi-path congestion control transport protocol to ensure balanced utilization of channels and fairness across flows. Flare [15] is a routing scheme for lightening network where nodes share their knowledge of the network in form of a routing table, that is used to find paths to recipients but when it is impossible, determine beacon nodes that are likely to help in finding these paths. SpeedyMurmurs [11] is a routing scheme that extends VOUTE [16] for decentralized Path-Based Transaction networks using efficient and flexible embedding-based path discovery depending on the presence of landmark nodes. While the algorithm is privacy preserving, it suffers from low payment throughput because it does not consider dynamic channel balance. HushRelay [10] is an efficient privacy-preserving routing scheme, taking into account funds left in each channel while splitting a payment across several paths.

There also exists work on how to optimize the offchain payment topology to reduce fees, in a demand-aware manner. Existing approaches rely on linear programming techniques [17] or focus on particular topologies such as stars, rings and expanders [18]. By their focus on the topology, the works are orthogonal to ours. We also note that demand-aware topologies have also been studied outside the specific context of blockchains, e.g., [19, 20]. We are also not the first to observe the opportunity of optimizing demand matrices to improve network performance, which has for example been studied in the context of content distribution in the Internet [21]. As our approach relies on the detection of cycles, our work is further related to cyclic trading optimizations in barter systems, e.g., [22, 23].

III. Model

A. Offchain Topology

We model an offchain network as an undirected graph where nodes are users and an (undirected) edge $e = \{u, v\}$ illustrates a payment channel between a pair of users u, v. Two users connected through a direct channel can transfer tokens between them, as long as the transfer follows bounds implied by the tokens locked in the channel. In addition, even without such a channel, a payment between two users can be served indirectly through a multi-hop path, a path of payment channels involving intermediate nodes such that its two endpoints are the sender and the recipient. The graph is called the *offchain topology graph* and describes users and payment channels between them.

This graph has similarity to a representation of the physical structure of a communication network, wherein communicating devices appear as nodes and the connections between the devices are described as edges, and data transmissions can be served through a multi-hop path from sender to recipient. The main differences between the two are: *(i)* In communication networks after serving a transmission the bandwidth of the bus does not change, in contrast to the capacity change after serving a transaction in an offchain network. *(ii)* Transmissions of data in the communication networks cannot cancel each other, each transmission is independent to others and contains special data. In contrast, in the context of offchain networks, since the transactions are token transfers, two transactions in opposite directions can cancel or partially cancel each other.

B. Demand Matrix

We are given a demand matrix $D_{N\times N}$ between N users that describes payments pending to be served on the offchain network. This can reflect offchain transactions aggregated in some period of time or periodic payments defined ahead. The diagonal of $D_{N\times N}$ equals zero and element $D_{i,j} \geq 0 \ \forall i, j \in [1:N]$ represents a requested payment of a particular value from user i to j. For a demand matrix D, we denote by $C_D = \sum_{i,j\in[1:N]} I(D_{i,j} > 0)$ the count of distinct payments where $I(\cdot)$ is the indicator function and by $S_D = \sum_{i,j\in[1:N]} D_{i,j}$ the total payments values. $D_{N\times N}$ can also be viewed as a directed demand graph. In such a graph nodes are users and edges are pending payments. A directed edge (s, r) with a positive weight $w_{s,r} > 0$ represents a pending payment of value $w_{s,r}$ from s to r.

We need to serve payments according to the demand matrix $D_{N\times N}$ on the offchain network topology graph $G = (V, E)$. The cost can refer to the number of payments that have to be served or to the number of offchain transactions required to serve them, potentially making payments via multiple channels with more than a single transaction. The cost can also be affected by the value of a transaction.

In order to focus on the benefits of demand matrix optimizations, we assume that the routing policy is given which implies a routing cost $d_G(i, j)$ from user i to user j in topology graph $G = (V, E)$ (e.g., based on the length of the shortest path). In computing such a cost we assume $D_{i,j} \geq 0$ and $d_G(i, j) \geq 0$ for all $i, j \in [1 : N]$. *The cost stands for a transaction fee to intermediate nodes*.

Serving a demand matrix $D_{N\times N}$ results in an update to the account balance of the N users. While for a demand matrix, a user can be involved in multiple payments either as a payer or as a payee, the aggregated impact on its balance is of importance. We can describe this impact on each of the N users by a vector Δ_D of length N named the *delta vector* that describes the outcome (either positive or negative) of each of the account upon serving the demand matrix D.

Definition 1. *(Delta Vector) The impact of serving a demand matrix on N users is described as a vector Δ_D of length N such as*

$$\Delta_D(i) = \sum_{j\in[1:N]} D_{i,j} - \sum_{j\in[1:N]} D_{j,i} \tag{1}$$

In case $\Delta_D(i) > 0$ the user i should send $\Delta_D(i)$ tokens, otherwise, it should receive $-\Delta_D(i)$ tokens.

Since the delta vector of a demand matrix is of importance, in many cases for a given demand matrix, we can refer to an alternative matrix that implies the same delta vector. We would be interested in a demand matrix that implies the same delta vector but would involve a low service cost.

Definition 2. *(Matrix Equivalence) We say that two demand matrices $D_{N\times N}, \tilde{D}_{N\times N}$ are equivalent if they imply the same delta vector $\Delta_D = \Delta_{\tilde{D}}$. We denote such an equivalency by $D \cong \tilde{D}$.*

Serving all payments from two equivalent matrices $D \cong \tilde{D}$ results in the same update to user accounts balance.

IV. Optimization Problem

In an offchain network, each user has its direct neighbors, those she shares payment channels with. To serve a payment in the demand matrix $D_{N\times N}$ between two users without a direct channel, the sender is obliged to send her tokens through a multi-hop path and pay a fee to each user on this path to complete the payment. Assuming a constant fee across all offchain users, which is independent on the transaction value, gives lower fees for shorter paths, in particular zero fee for serving payments to direct neighbors. In our work, we focus on optimizing the demand of the users according to the offchain network topology.

Problem 1. ***Minimal Transaction Problem:*** *Given a demand matrix D, find an equivalent matrix $\tilde{D}$ (satisfying $D \cong \tilde{D}$) of a minimal weight $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0)$.*

While the first problem does not refer to a particular topology G, the second problem optimizes a cost affected by the particular topology.

Problem 2. ***Base Cost Problem:*** *Given a demand matrix D and a network topology G, find an equivalent matrix*

$\tilde{D}$ (satisfying $D \cong \tilde{D}$) and $\tilde{D}$ minimizing the base cost for the transactions to be served $C(\tilde{D}, G) = \sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) \cdot d_G(i,j)$. In $d_G(i,j)$ we refer to the number of required transactions (hop count) within the topology G.

V. How to Optimize a Demand Matrix?

In this section we study properties of a demand matrix $D_{N\times N}$. We identify manipulations that can be applied to the demand matrix to reach an equivalent matrix $\tilde{D}$ such that $D \cong \tilde{D}$. Intuitively, we would like such changes to reduce the costs (as defined in Problems 1-2) in serving the demand matrix in some topology G. For instance, the process results in a small number of payments $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0)$. We will consider both general topologies as well as to particular topologies.

A. Demand Matrix Manipulation for General Topologies

The first simple property allows bounding the number of nonzero values in a demand matrix to at most $\frac{1}{2} \cdot N(N-1)$.

Property 1. *(One-sided Demand Matrix) For any demand matrix $D_{N\times N}$, there exists an equivalent matrix $\tilde{D}_{N\times N}$ (satisfying $\tilde{D} \cong D$) such that for any sender-recipient pair (i,j): $\tilde{D}_{i,j} \cdot \tilde{D}_{j,i} = 0$.*

Proof. We construct $\tilde{D}_{N\times N}$ based on two cases.

1) If $D_{i,j} \cdot D_{j,i} = 0$, we set $\tilde{D}_{i,j} = D_{i,j}$ and $\tilde{D}_{j,i} = D_{j,i}$.
2) If $D_{i,j} \cdot D_{j,i} > 0$ (namely both values are positive), let $D_{min} = \min\{D_{i,j}, D_{j,i}\} > 0$. We set $\tilde{D}_{i,j} = D_{i,j} - D_{min}$ and $\tilde{D}_{j,i} = D_{j,i} - D_{min}$.

It is easy to see that for every pair of indices $\tilde{D}_{i,j} \cdot \tilde{D}_{j,i} = 0$. Moreover, $\tilde{D} \cong D$ since any change of the values for a pair of indices i,j maintains the same delta vector $\Delta_{\tilde{D}} = \Delta_D$. For some index $q \in [1,N]$, clearly $\Delta_{\tilde{D}}(q) = \Delta_D(q)$ if $q \notin i,j$. If $q = i$ for instance, then $\Delta_{\tilde{D}}(q) = \sum_{k\in[1:N]} \tilde{D}_{q,k} - \sum_{k\in[1:N]} \tilde{D}_{k,q} = (\sum_{k\in[1:N]} D_{q,k} - D_{min}) - (\sum_{k\in[1:N]} D_{k,q} - D_{min}) = \Delta_D(q)$. □

The next property generalizes the discussion to matrices with negative values. Intuitively, a demand for a payment of value $D_{i,j} < 0$ is equivalent to a demand for payment of value $-D_{i,j} > 0$ in the other direction.

Property 2. *(Non-Negative Demand Matrix) A demand matrix D, potentially with negative values, has an equivalent demand matrix $\tilde{D}$ with only non-negative values.*

Proof. $\tilde{D}$ is derived from D through first setting $\tilde{D} = D$ followed by elimination of the negative values while keeping the matrices equivalent. For each i,j such that $D_{i,j} < 0$, add $-D_{i,j} > 0$ to both $\tilde{D}_{i,j}$ and $\tilde{D}_{j,i}$. Such an addition does not change the implied delta vector. □

We now refer to properties of a demand matrix that relate to the number of transactions that are required to serve it. In particular for a given demand matrix, we find an equivalent matrix which is more sparse, i.e., has a larger number of zero elements. We explain how this allows serving the matrix at lower cost.

The first property shows that for any demand matrix $D_{N\times N}$ defined for N users, there is an equivalent matrix with at most $N-1$ nonzero values.

Property 3. *(Sparsification) A demand matrix $D_{N\times N}$ has an equivalent demand matrix $\tilde{D}_{N\times N}$ satisfying $\tilde{D} \cong D$ and $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) \leq N-1$.*

Proof. Let Δ_D be the delta vector implied by the demand matrix $D_{N\times N}$. $\tilde{D}$ is constructed as follows. For $i \in [1:N]$ it satisfies $\tilde{D}_{i,j} = 0$ if $j \neq i+1 \pmod N$ and $j+1 \neq i \pmod N$. We first set $\tilde{D}_{1,2}$ as $\Delta_D(1)$, $\tilde{D}_{2,1} = 0$ if $\Delta_D(1) \geq 0$ and $\tilde{D}_{1,2} = 0$, $\tilde{D}_{2,1} = -\Delta_D(1)$ otherwise. For $i \in [2:N-1]$ we set $\tilde{D}_{i,(i+1)} = \Delta_D(i) + \tilde{D}_{(i-1),i} = \sum_{j\in[1:i]} \Delta_D(j)$, where the last equality can be shown by a simple induction. If by the above formula $\tilde{D}_{i,(i+1)}$ is negative we keep $\tilde{D}_{i,(i+1)} = 0$ and instead set $\tilde{D}_{(i+1),i} = -\sum_{j\in[1:i]} \Delta_D(j)$. Clearly, $\tilde{D}$ satisfies $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) \leq N-1$, namely has at most $N-1$ nonzero values. It is also easy to see that $\Delta_D = \Delta_{\tilde{D}}$ since $\Delta_{\tilde{D}}(1) = \tilde{D}_{1,2} = \Delta_D(1)$ and for $i \in [2:N]$ $\Delta_{\tilde{D}}(i) = \tilde{D}_{i,(i+1)} - \tilde{D}_{(i-1),i} = \Delta_D(i)$. □

For every demand matrix D the delta vector satisfies $\sum_{i\in[1:N]} \Delta_D(i) = \sum_{i\in[1:N]} \left(\sum_{j\in[1:N]} D_{i,j} - \sum_{j\in[1:N]} D_{j,i}\right) = \sum_{i,j\in[1:N]} D_{i,j} - \sum_{i,j\in[1:N]} D_{i,j} = 0$. In case there exists multiple disjoint subsets of nodes $\phi_1, \ldots, \phi_k$ with $k \geq 2$ such that for every $i \in [1:k]$ it holds $\sum_{j\in\phi_i} \Delta_D(i) = 0$ then there exists an equivalent demand matrix with less than $N-1$ nonzero values. We refer to ϕ_i for $i = 1 \ldots k$ as a zero-sum subset of nodes. Intuitively, we can have a demand matrix with nonzero values that refer only to pairs of nodes within the same subset such that each subset ϕ_i corresponds to $|\phi_i| - 1$ such nonzero values.

Property 4. *(Sparse Partitions) Consider a demand matrix $D_{N\times N}$ with a delta vector Δ_D. If there is a partition of the N nodes into disjoint subsets $\phi_1, \ldots, \phi_k$ such that $\sum_{j\in\phi_i} \Delta_D(i) = 0$ for $i \in [1:k]$ then an equivalent demand matrix $\tilde{D}_{N\times N}$ satisfying $\tilde{D} \cong D$ and $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) \leq N-k$ exists.*

Proof. This follows from applying Property 3 on each subset ϕ_i and implying the corresponding $|\phi_i|$ values in the delta vector through $|\phi_i| - 1$ payments in the demand matrix. □

By Property 4 in order to find for a demand matrix $D_{N\times N}$ an equivalent demand matrix $\tilde{D} \cong D$ with a minimal number of nonzero values $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0)$ it is required to partition the nodes into a maximal number of disjoint subsets with zero sum of their delta vector values.

Example 1. *Consider for $N = 5$ a demand matrix* $D = \begin{pmatrix} 0 & 7 & 8 & 1 & 0 \\ 2 & 0 & 3 & 0 & 7 \\ 1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 5 \\ 7 & 0 & 0 & 0 & 0 \end{pmatrix}$. *It has a delta vector*

$\Delta_D = (5, 3, -7, 4, -5)$. *The matrix* D *satisfies* $\sum_{i,j\in[1:N]} I(D_{i,j} > 0) = 12$. *By Property 3, an equivalent demand matrix is* $\tilde{D} = \begin{pmatrix} 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ *with only* $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) = N - 1 = 4$ *nonzero values.*

Note that $\Delta_D(1) + \Delta_D(5) = \Delta_D(2) + \Delta_D(3) + \Delta_D(4) = 0$. *By Property 4 an equivalent demand matrix* $\hat{D}$ *can be found with nonzero values for pairs within the subsets of node indices* $\{1, 5\}$ *and* $\{2, 3, 4\}$. *In particular,* $\hat{D} = \begin{pmatrix} 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ *and has only* $\sum_{i,j\in[1:N]} I(\hat{D}_{i,j} > 0) = N - 2 = 3$ *nonzero values.*

By Property 3 for every demand matrix, there exists an equivalent demand matrix with at most $N-1$ nonzero values. In the following we show that this upper bound is tight. Namely, there are demand matrices for which there is no equivalent matrix with less than $N - 1$ nonzero values.

Property 5. *(Lower Bound) There is a demand matrix* $D_{N\times N}$ *for which every equivalent demand matrix* $\tilde{D} \cong D$ *satisfies* $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) \geq N - 1$.

Proof. Consider a demand matrix $D_{N\times N}$ for which $D_{i,N} = 1$ for $i \in [1 : N - 1]$ and $D_{i,j} = 0$ otherwise. It results in a delta vector $\Delta_D = (1, 1, \ldots, 1, 1 - N)$ with $N - 1$ positive values. Any equivalent demand matrix $\tilde{D}$ (of non-negative values) must have at least one nonzero value in each of the top $N-1$ rows since for $i \in [1 : N-1]$ we have $1 = \Delta_D(i) = \sum_{j\in[1:N]} D_{i,j} - \sum_{j\in[1:N]} D_{j,i}$ so that $\sum_{j\in[1:N]} D_{i,j} > 0$. This implies that $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) \geq N - 1$. □

We deduce that finding optimal solutions to the optimization problems is difficult in the general case. The proof relies on a reduction from the partition problem [24] and is omitted due to space constraints.

Theorem 1. *Finding an optimal solution to either problem, Problem 1 and Problem 2, is NP-complete.*

B. Algorithms for Problems 1-2

In this section, we present solutions to Problem 1 and Problem 2. Recall that by Theorem 1 it is hard to find a solution with the minimal cost for each instance. The solution we suggest for Problem 1 produces an equivalent demand matrix with an upper bound of $N - 1$ on its nonzero values. By Property 5 this is the best possible bound that can be suggested in the general case.

Interestingly, for Problem 2 we show that for every connected offchain topology G we can find an equivalent demand matrix with a base cost of $N - 1$. We find this result non trivial since such a cost is affected by the topology and the result relies on the observation that *for any connected topology we can find an equivalent demand matrix for which each payment can be served in a single transaction*. We demonstrate that on the simple *ring* and *star* topologies and then generalize the claim for an arbitrary topology.

	# Edges	Diameter	# Transactions Upper Bound	# Hops Upper Bound
Star	$N-1$	2	$N-1$ (Property 3)	$N-1$ (Property 6)
Ring	N	$\lfloor N/2 \rfloor$		
General	$\geq N-1$	$\leq N-1$		

TABLE I: Topology graph $G = (V, E)$ with $|V| = N$ nodes

Problem 1. The input includes a demand matrix $D_{N\times N}$. The equivalent demand matrix $\tilde{D}_{N\times N}$ is produced as in the proof of Property 3 based on the delta vector Δ_D.

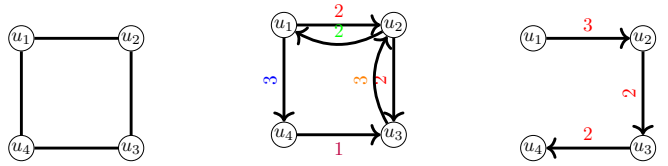
In particular, for $i \in [1 : N - 1]$ we set $\tilde{D}_{i,(i+1)} = I(\sum_{j\in[1:i]} \Delta_D(j) \geq 0) \cdot \sum_{j\in[1:i]} \Delta_D(j)$. Similarly, $\tilde{D}_{(i+1),i} = -I(\sum_{j\in[1:i]} \Delta_D(j) < 0) \cdot \sum_{j\in[1:i]} \Delta_D(j)$. Last, $\tilde{D}_{i,j} = 0$ if $j \neq i + 1 \pmod N$ and $j + 1 \neq i \pmod N$. Clearly, $\tilde{D}$ has a weight of at most $N - 1$.

Next, we refer to Problem 2 that its input includes in addition to the demand matrix $D_{N\times N}$ also an offchain topology G. We firsts study the ring topology and the star topology, and then consider arbitrary graphs. Basic properties of such topologies such as their number of edges and diameter are summarized in Table I.

Problem 2 - The Ring Topology. The users in a *ring* topology create a circular data path. Each user is connected to two adjacent users, like points on a circle. In this topology data travels from sender to recipient through intermediate users, clockwise or anti-clockwise until it reaches the recipient. Consider a ring topology of N users. It has N edges and a diameter of $\lfloor N/2 \rfloor$. Without loss of generality denote the nodes $u_1, u_2, \ldots, u_{N-1}, u_N$ based on a clockwise order in the topology starting from an arbitrary node such the topology has edges $u_1 - u_2 - \ldots - u_{N-1} - u_N - u_1$. Consider the demand matrix $D_{N\times N}$ based on that order of the nodes.

Consider the equivalent demand matrix $\tilde{D}$ from the above solution to Problem 1. It has a weight of at most $N - 1$ and all its payments are in indices of the form i, j for $j = i + 1 \pmod N$ or $j + 1 = i \pmod N$. Namely, each payment of $\tilde{D}$ refers to two nodes connected with a direct channel in the ring topology.

A demand matrix $D = \begin{pmatrix} 0 & 0 & 2 & 3 \\ 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ for $N = 4$ users u_1, u_2, u_3, u_4 has a delta vector $\Delta_D = (3, -1, 0, -2)$. To serve these payments on a ring $u_1 - u_2 - u_3 - u_4 - u_1$ of Fig. 2(a) six transactions are required, see Fig. 2(b). An equivalent demand matrix is $\tilde{D} = \begin{pmatrix} 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ has $N - 1 = 3$ payments, all between nodes connected in the topology and thus requires only three transactions, see Fig. 2(c). In case the ring was connected differently such as $u_1 - u_3 - u_2 - u_4 - u_1$ then a relevant equivalent matrix



(a) Offchain *Ring* topology network.

(b) Serving original demand D on *Ring* topology.

(c) Serving the optimized demand on *Ring* topology.

Fig. 2: Offchain network with four users connected in *Ring* topology Fig. 2(a). Illustration of serving original demand matrix Fig. 2(b) vs. the optimized demand matrix Fig. 2(c)

can be $\hat{D} = \begin{pmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. It again has no more than $N-1=3$ payments and they all take a single transaction in the particular topology. $\hat{D}$ is derived by computing the delta vector based on an order of nodes that follows the topology.

Problem 2 - The Star Topology. In the *star* topology all the network nodes are individually connected to a central point of communication. A payment between two nodes other than the central node can be performed through two payments involving the central node. Previous work [3, 25] motivated the use of the star topology for payment channels systems with the option of the central node in the topology to serve as a payment channel hub. The star topology has $N-1$ edges connecting the N users as well as a diameter of 2. In a star topology, we can optimize the demand matrix $D_{N\times N}$ (with a delta vector Δ_D) to an equivalent one $\tilde{D}_{N\times N}$ that contains at most $N-1$ nonzero entries, where each payment is to or from the central node. Without loss of generality, let N be the index of the user located at the central node. A user with a positive value in the delta vector sends a transaction to the central user and receives a transaction if its value is negative. Namely, for $i \in [1:N-1]$ we set $\tilde{D}_{i,N} = \Delta_D(i) \cdot I(\Delta_D(i) > 0)$ and $\tilde{D}_{N,i} = -\Delta_D(i) \cdot I(\Delta_D(i) < 0)$. Likewise $i, j \in [1:N-1]$ we set $\tilde{D}_{i,j} = 0$. Only a single transaction to or from the central user is needed to each other user and accordingly at most $N-1$ transactions are required on the offchain star topology.

Problem 2 - A General Topology. We now generalize the above results of a ring and a star and show that for an *arbitrary connected topology* G and a demand matrix $D_{N\times N}$, we can find an equivalent demand matrix that has at most $N-1$ payments such that each is served by a single payment in the topology.

Given the topology G, we first select an arbitrary spanning tree T. Any such spanning tree T has at least two leaves, vertices that have only one edge of T incident to them. We begin with a demand matrix $\tilde{D}$ without payments. We repeatedly select such a leaf i and add to the matrix a payment between the leaf to its adjacent node j. The direction of the payment and its value is determined by $\Delta_D(i)$ such that $\tilde{D}_{i,j} = \Delta_D(i) \cdot I(\Delta_D(i) > 0)$ and $\tilde{D}_{j,i} = -\Delta_D(i) \cdot I(\Delta_D(i) < 0)$. No payment is added if $\Delta_D(i) = 0$. We remove the leaf from the tree and remain with a spanning tree for other nodes. Likewise, we update the delta vector by adding to $\Delta_D(j)$ the value of the payment if the payment is to node j or subtracting the value if the transaction is from node j. Since the payment is between two nodes adjacent in the tree $T \subseteq G$ it can be implemented by a single transaction in the original topology G. The process continues till two nodes remain in the spanning tree and if their payment is added, we get a total of at most $N-1$ transactions.

We conclude this description with the following property.

Property 6. *(Single Transaction) For every connected topology G, a demand matrix $D_{N\times N}$ with an arbitrary number of payments, has an equivalent demand matrix with at most $N-1$ payments that can all be served by a single transaction in the topology.*

VI. Proportional Cost

We generalize the cost function of Problems 1-2 and define an new optimization problem that refers to a cost function for which the fee for sending a transaction is proportional to the value of the transaction.

A. The Proportional Cost Problem and its Properties

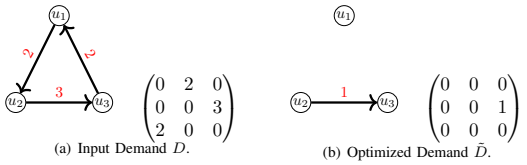
Problem 3. ***Proportional Cost Problem:*** *Given a demand matrix D and a topology G, find an equivalent matrix $\tilde{D}$ (satisfying $D \cong \tilde{D}$) such that $\tilde{D}$ minimizes within the topology G the cost proportional to the transactions size $\Psi(\tilde{D}, G) = \sum_{i,j\in[1:N]} \tilde{D}_{i,j} \cdot d_G(i,j)$.*

The following property allows changing the demand matrix along a list of pairs that imply a cycle.

Property 7. *(Cyclic Elimination) Consider a demand matrix $D_{N\times N}$ and a cycle of nodes $C = (u_0, u_1, u_2, \ldots, u_{\ell-1}, u_\ell)$ with $u_0 = u_\ell$. A demand matrix $\tilde{D}_{N\times N}$ given by subtracting a constant w from all cycle edges $u_k \to u_{k+1}$, $k \in [0, \ell-1]$ satisfies $\tilde{D} \cong D$.*

Proof. To demonstrate the matrix equivalence, we show that the delta vectors satisfy $\Delta_{\tilde{D}} = \Delta_D$. First, if $q \notin C$, clearly $\tilde{D}_{q,k} = D_{q,k}$ and $\tilde{D}_{k,q} = D_{k,q}$ for every k. Thus $\Delta_{\tilde{D}}(q) = \sum_{k\in[1:N]} \tilde{D}_{q,k} - \sum_{k\in[1:N]} \tilde{D}_{k,q} = \sum_{k\in[1:N]} D_{q,k} - \sum_{k\in[1:N]} D_{k,q} = \Delta_D(q)$. On the other hand, if $q \in C$ then $\tilde{D}_{q,k} = D_{q,k} - w$ if k immediately follows q in the cycle C and $\tilde{D}_{q,k} = D_{q,k}$ otherwise. Here, $\Delta_{\tilde{D}}(q) = \sum_{k\in[1:N]} \tilde{D}_{q,k} - \sum_{k\in[1:N]} \tilde{D}_{k,q} = (\sum_{k\in[1:N]} D_{q,k} - w) - (\sum_{k\in[1:N]} D_{k,q} - w) = \Delta_D(q)$.

In particular, by selecting w as the minimal nonzero value in the demand matrix $D_{N\times N}$, the equivalent matrix $\tilde{D}$ has a smaller number of nonzero values $\sum_{i,j\in[1:N]} I(\tilde{D}_{i,j} > 0) \leq \sum_{i,j\in[1:N]} I(D_{i,j} > 0) - 1$. Moreover, in any topology G and for any $w > 0$ the change reduces the base cost $C(D, G)$ (in Problem 2) and the proportional cost $\Psi(D, G)$ (in Problem 3) of serving the original demand matrix $D_{N\times N}$.



(a) Input Demand D. (b) Optimized Demand $\tilde{D}$.

Fig. 3: Illustration of a payment demand in Fig. 3(a), and demand after cycle elimination in Fig. 3(b).

Note that the ability to refer to matrices with negative values allows us to make use of Property 7 by reducing values smaller the minimal among the edges of the cycle. □

Fig. 3 shows an example for an payment demand D and an equivalent demand $\tilde{D}$ obtained after a cycle elimination.

In the following we refer to Problem 3 of minimizing the proportional cost in the case of a symmetrical topology G such that the cost $d_G(i,j)$ has a fixed value for all pairs i,j. We point out a particular equivalent matrix $\tilde{D}$ that minimizes the proportional cost among those obtained by subtracting a fixed value from a given cycle.

Property 8. *(Median Subtraction) Consider a demand matrix $D_{N\times N}$ and a symmetrical topology G that implies a fixed cost $d_G(i,j) = d > 0$ for all pairs. Let $C = (u_0, u_1, u_2, \ldots, u_{\ell-1}, u_\ell)$ with $u_0 = u_\ell$ be a cycle of nodes. Among all equivalent demand matrices $\tilde{D}_{N\times N}$ given by subtracting a constant w from the cycle edges, the demand matrix that minimizes the proportional cost $\Psi(D,G)$ is given for the constant $w = w_{med}$ where w_{med} is the median value among the ℓ edge weights of the cycle C.*

Proof. Payments between nodes which are not edges in the cycle have a fixed contribution to the proportional cost $\Psi(D,G)$. Consider a payment (from the cycle) of value x changed to the value of $x - w$. The contribution of the payment to the cost is $|x - w| \cdot d$ where $d > 0$. Thus the cost is monotonically decreasing in the range $w \in [0, x]$, reaches a minimal value of 0 for $w = x$ and is monotonically increasing for $w > x$. Within each such range, a change of w by ϵ implies the same change on the contribution to the cost. For the cost $\Psi(D,G)$ as a whole, if w is smaller than the median value among the ℓ edge weights, then increasing w would reduce the contribution of more than half of the payments (and would increase the contribution of less than a half). On the other hand, if w is larger than the median decreasing its value would reduce the contribution of more than half of the payments. Thus the minimal cost is obtained for $w = w_{med}$. □

In case the number of the edges in odd, the median is well defined. Otherwise, the median can be one of the two middle elements or their average.

The following example illustrates the above properties for a particular demand matrix and equivalent matrices:

Example 2. *Assume a symmetrical topology G that implies a fixed cost $d_G(i,j) = d$ for all pairs of nodes. Consider a demand matrix $D = \begin{pmatrix} 0 & 8 & 0 & 1 \\ 0 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. This demand matrix demonstrates a cycle with three users $u_1 \to u_2 \to u_3 \to u_1$.*

We generate four equivalent demand matrices as follows:

(i) Matrix after reducing minimal cycle value $w_{min} = 1$, $\tilde{D}_{min} = \begin{pmatrix} 0 & 7 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ implying a cost $\Psi(D,G) = 10 \cdot d$.

(ii) Matrix after reducing maximal cycle value $w_{max} = 8$, $\tilde{D}_{max} = \begin{pmatrix} 0 & 0 & 7 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ implying a cost $\Psi(D,G) = 13 \cdot d$.

(iii) Matrix after reducing average cycle value $w_{avg} = 4$, $\tilde{D}_{avg} = \begin{pmatrix} 0 & 4 & 3 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ implying a cost $\Psi(D,G) = 9 \cdot d$.

(iv) Matrix after reducing median cycle value $w_{med} = 3$, $\tilde{D}_{med} = \begin{pmatrix} 0 & 5 & 2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ implying a cost $\Psi(D,G) = 8 \cdot d$.

As claimed by Property 8, the optimal result is achieved upon reducing from the cycle the value of its median.

B. Algorithm for Problem 3

We present a heuristic for optimizing a demand matrix $D_{N\times N}$ for an arbitrary symmetrical graph topology. The optimization has two stages. First we use Property 1 and generate a one-sided demand matrix (removing demand cycles between two users). Second, we detect cycles with positive payment values iteratively. We detect cycles by running the DFS algorithm that produces a tree for a connected graph. A cycle can be detected in a graph only if there is a back edge present in the DFS tree. A back edge is an edge that is from a node to itself (self-loop) or one of its ancestors in the tree produced by the DFS. For each detected cycle we perform Property 8 and subtract the median value among the values of the detected cycle. We run this processing iteratively as long as the demand graph contains cycles.

VII. Ethereum-based Experimental Evaluation

To complement our analytical insights and in order to shed light on the potential benefits of our approach in practice, we conducted an empirical evaluation on Ethereum data [2]. The demand refers to Ethereum payments between May 1 and October 31, 2019. We categorized the demand into disjoint equal-length epochs of 1 second, 10 seconds, 1 minute, 10 minutes, 30 minutes, 1 hour, 3 hours, 6 hours and 12 hours. Specifically, we collected data on Ethereum, expecting a similar demand on an offchain network.

Fig. 4 represents the average ratio between number of distinct users and number of payments in each of the time epochs. We observe that for longer epochs the ratio decreases meaning that each user is performing more payments, which

implies a denser demand matrix allowing potential for larger savings in optimizing.

Next we evaluate the optimization achieved on two specific topologies: *star* and *ring*, when finding an equivalent demand matrix as explained in Section V-B for each topology. We present the average cost (number of needed transactions to serve all payments) on the topology in each epoch, before and after the processing of the demand matrix.

The results can be found in Fig. 5 and Fig. 6. As we can see in both topologies, the obtained cost from demand matrix processing reflects optimization. In addition, as the length of the epoch increases the obtained savings increases too, which is caused by the increased density of the demand matrix. For *ring*, we observe large savings (caused by the large diameter $\lfloor N/2 \rfloor$) for longer epochs with a large number of users; this makes serving all payments expensive, while with demand matrix pre-processing, the number of transactions reduces rapidly as a result of aggregating transactions. For example, for epochs of 1-minute and 6-hours we detect a saving of 98.58% and 99.9%, respectively. For *star* the savings is typically much smaller. Since the *star* diameter is 2, then its savings are limited by the average distance between sender and recipient which is between 1 and 2, and by amount of repeated transactions, which is more probable in longer epochs. For instance, for time epochs of 1-second and 3-hours, the observed savings is 96.3% vs. 100% for ring. and equals 20% and 51.7% for star.

For the evaluation of the general case of Problem 2, we generate a connected random graph with different values of p (connectivity probability) and show the average ratio between the original and optimized cost (number of needed transaction to serve all payments) on different random graphs as a function of p. The evaluation is on epochs of 10-minutes.

Fig. 7 shows the results. The connectivity probability p reflects the number of offchain channels in the topology graph: as p increases the probability for a direct channel between sender and recipient increases. This reduces the demand processing, especially for a sparse demand, as in 10-minutes epochs. For example, for $p = 0.1$ the ratio between the original cost and optimized cost is 1.53, indicating savings of 36.6% while for $p = 0.5$ the savings are 17.5%.

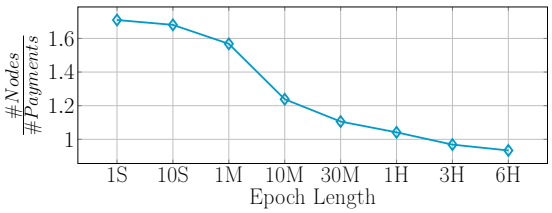


Fig. 4: Ethereum data: Average ratio of the distinct users per payment as a function of epoch length. Epochs lengths in range 1 second - 6 hours.

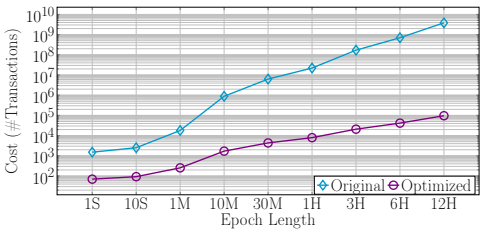


Fig. 5: *Ring* topology: Average number of transactions before and after the optimization as a function of epoch length.

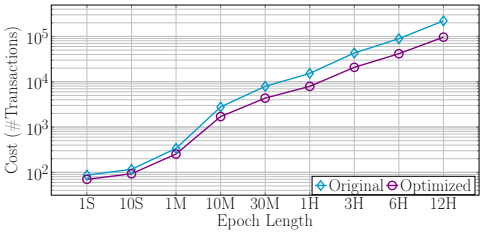


Fig. 6: *Star* topology: Average number of transactions before and after the optimization as a function of time length.

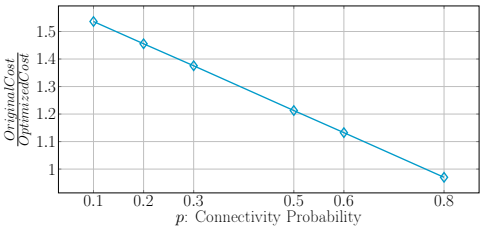


Fig. 7: *Random graph* topology: Average cost ratio (before and after the optimization) as a function of connectivity probability per edge p, in time epoch of 10-minutes.

VIII. Conclusion and Future Work

We initiated the study of demand matrix optimizations for offchain payment networks in Blockchain. This approach benefits from the interaction among requested payments and allows a meaningful savings in the service cost. We studied the optimization of a demand matrix independently as well as with respect to its service cost in a particular topology.

As a future work, we would like to combine the matrix optimization with other critical degrees of freedom in the offchain network design that we assumed are given. These include the routing policy of payments or alternatively the topology construction assuming it contains a restricted number of direct channels.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Bitcoin white paper*, 2008.

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014. [Online]. Available: https://gavwood.com/paper.pdf

[3] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "SoK: Layer-two blockchain protocols," in *Financial Cryptography and Data Security (FC)*, 2020.

[4] J. Poon and T. Dryja, "The Bitcoin Lightning network: Scalable off-chain instant payments," 2016.

[5] "Raiden network," 2017. [Online]. Available: http://raiden.network/

[6] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and outlook," in *International Conference on Trust and Trustworthy Computing (TRUST)*, 2015.

[7] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. C. Fanti, and P. Viswanath, "Routing cryptocurrency with the Spider network," in *ACM HotNets*, 2018.

[8] P. Wang, H. Xu, X. Jin, and T. Wang, "Flash: Efficient dynamic routing for offchain networks," in *ACM CoNEXT*, 2019.

[9] U. Nisslmueller, K. Foerster, S. Schmid, and C. Decker, "Toward active and passive confidentiality attacks on cryptocurrency off-chain networks," *arXiv preprint 2003.00003*, 2020.

[10] S. Mazumdar, S. Ruj, R. G. Singh, and A. Pal, "HushRelay: A privacy-preserving, efficient, and scalable routing algorithm for off-chain payments," *arXiv preprint 2002.05071*, 2020.

[11] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling payments fast and private: Efficient decentralized routing for path-based transactions," *arXiv preprint 1709.05748*, 2017.

[12] "Lightning RFC: Lightning network specifications," 2019. [Online]. Available: https://github.com/lightningnetwork/lightning-rfc

[13] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. Pietzuch, and E. G. Sirer, "Teechain: Scalable blockchain payments using trusted execution environments," *arXiv preprint 1707.05454*, 2017.

[14] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *IEEE Symposium on Security and Privacy (SP)*, 2019.

[15] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An approach to routing in Lightning network," *White Paper*, 2016.

[16] S. Roos, M. Beck, and T. Strufe, "VOUTE-virtual overlays using tree embeddings," *arXiv preprint 1601.06119*, 2016.

[17] L. Aumayr, E. Ceylan, M. Maffei, P. Moreno-Sanchez, I. Salem, and S. Schmid, "Demand-aware payment channel networks," *arXiv preprint 2011.14341*, 2020.

[18] J. Khamis and O. Rottenstreich, "Demand-aware channel topologies for off-chain payments," in *International Conference on Communication Systems and Networks (COMSNETS)*, 2021.

[19] C. Avin, K. Mondal, and S. Schmid, "Demand-aware network designs of bounded degree," *Distributed Computing*, 2019.

[20] C. Avin, A. Hercules, A. Loukas, and S. Schmid, "rDAN: Toward robust demand-aware network designs," *Information Processing Letters*, 2018.

[21] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber, "Pushing CDN-ISP collaboration to the limit," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 34–44, 2013.

[22] R. Eidenbenz, T. Locher, S. Schmid, and R. Wattenhofer, "Boosting market liquidity of peer-to-peer systems through cyclic trading," in *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2012.

[23] C. Hajaj, J. P. Dickerson, A. Hassidim, T. Sandholm, and D. Sarne, "Strategy-proof and efficient kidney exchange using a credit mechanism," in *Conference on Artificial Intelligence (AAAI)*, 2015.

[24] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2006.

[25] Y. W. Georgia Avarikioti, Gerrit Janssen and R. Wattenhofer, "Payment network design with fees," in *Springer Int. Workshop on Data Privacy Management, Cryptocurrencies and Blockchain Technology*, 2018.