Noname manuscript No.
(will be inserted by the editor)

Demand-aware Channel Topologies for Off-chain Blockchain Payments

Julia Khamis · Ori Rottenstreich

Received: December 1, 2020

Abstract Off-chain is a common approach to deal with the scalability problem of blockchain networks. It enables users to execute multiple payments without committing each of them to the blockchain by relying on predefined payment channels. A pair of users can employ a payment even without a direct channel between them, via routing the payment through off-chain channels involving other intermediate users. Users together with the off-chain channels form a graph, known as the off-chain network topology. The off-chain topology and the payment characteristics affect network performance such as the average number of intermediate users a payment is routed through, the amount of fees, or channel capacities needed to successfully route payments. In this paper, we study two basic problems in off-chain network design. First, efficiently mapping users to an off-chain topology with a known structure. Second, constructing a topology of a bounded number of channels that can serve well users with associated payments. We design algorithms for both problems and evaluate them based on real data from Raiden, the off-chain extension for Ethereum.

Keywords Blockchain · Off-chain · Topologies

1 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 7 and 15 payments per second, respectively [2].

Off-chain networks are a leading solution for the scalability problem [3] [9] [10]. While there are different types of off-chain networks, all share the same conception. An off-chain network enables taking payment operations outside of the blockchain. Participants can use payment channels to make 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 when failing to serve a payment.

Two users u_1 and u_2 in the off-chain network can establish a bidirectional payment channel used to serve payments. To open a bidirectional 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 $(c_{(u_1,u_2)}, c_{(u_2,u_1)})$ satisfying $c_{(u_1,u_2)}, c_{(u_2,u_1)} \geq 0$, which specify the amount of tokens on channel owned by u_1 and u_2, respectively. The distribution of the locked value among the users of a channel changes based on their transfers, but the sum of the values remains constant as long as on-chain payments are not involved.

Users as nodes and payment channels as edges imply a graph, known as the *off-chain topology graph*. The off-chain 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, given that they do not have a direct channel between them but both have a direct channel to 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. When routing a payment through

A preliminary version of this article was accepted to the International Conference on Communication Systems and Networks (COMSNETS) '21, January 2021. This submission includes a mapping algorithm to the HyperX topology that is first presented here together with new real data experimental results (Fig. 5(b), Fig. 6(b), Fig. 7, Fig. 8(b), Fig. 9(b), Fig. 10 and Fig. 11(a)).

J. Khamis and O. Rottenstreich
Technion
Haifa, Israel
E-mail: sjul14@campus.technion.ac.il, or@technion.ac.il

(a) Input Pending Payments Directed Graph

(b) Topology Graph, User Mapping 1.

(c) Same Topology Graph, User Mapping 2.

(d) An alternative Topology Graph

Fig. 1: Illustration of pending payments Fig. 1(a), and channel topologies with mappings to serve them (Fig. 1(b)-1(d)). Fig. 1(b)-1(c) have an identical topology but with different mappings while the topology in Fig. 1(d) is different. An evaluation metric can be the average hop count using shortest path routing.

a channel, the channel's capacity changes according to the tokens transfer. Most of the off-chain previous works focus on optimizing the routing of the payment, improving the channels capacity balancing, and enhancing privacy and security [11] [12] [13]. Our approach relates the off-chain network design to traditional graph embedding methods that have been studied for communication networks [27] [28].

One of the limits of off-chain networks is their bounded number of channels, since each action of opening or closing a channel necessitates an on-chain transaction, therefore affecting the scalability issue of the on-chain that the off-chain is trying to solve [14]. In this sense, off-chain networks should be sparse but still improve the performance of payments. In this paper, we learn *the impact of off-chain network structure on its performance*. The topology structure of the network as well as the location of each user directly affects the performance. For example, as the number of hops in a routing path from sender to recipient can affect the *latency* in implying a payment, the existence of a direct channel between a pair of users making frequent payments, would certainly allow good performance. The average number of hops is one of many matrices that can be used to measure performance, in addition to total expected fees, maximum needed channel capacity, etc.

We study two main questions:

(1) Given an off-chain network topology and demand for particular payments, where to place each user in this topology?

(2) Given the payments demand, how to construct a matching network topology?

An illustration is given in Fig. 1 with four users $u_1, \ldots, u_4$; First, Fig. 1(a) describes the payment requests that have to be served. There are a total of $1 + 7 + 2 = 10$ payments, sent by u_1 and u_2 such that 7 of the payments are sent from u_1 to u_3. Next, Fig. 1(b)-1(d) show topology graphs for serving the payment requests. Fig. 1(b) and Fig. 1(c) show two options of mapping users to nodes in the same off-chain topology. In Fig. 1(b) users u_1 and u_3 do not have a direct channel, so each time user u_1 wants to pay u_3 it has to send its payment through u_4, involving two payment channels. On the other hand, in an alternative mapping of nodes to the same topology in Fig. 1(c), there exists a direct channel between u_1 and u_3 allowing to reduce the number of involved channels in serving the common payments in comparison to Fig. 1(b).

In addition, an alternative off-chain topology with the same number of channels can correlate better with the payment requests for further improving average path length (number of hops). In Fig. 1(d) user u_1 has a direct payment channel to each of the other users, allowing all payment requests to be served efficiently through a single channel. Indeed, this topology and mapping are ideal upon trying to reduce payment path lengths in serving the demand of Fig. 1(a). This illustrates the impact of the topology selection and the user mapping to the topology on the network efficiency.

Contributions: We make the following contributions:

(i) We define two fundamental optimization problems in the design of off-chain networks (Section 3.2).

(ii) We overview basic topologies and refer to fundamental properties of the problems (Section 4).

(iii) We present efficient solutions for the problems for common scenarios (Section 5).

(iv) We analyze real off-chain Raiden data and evaluate the algorithms for such real payment information (Section 6).

2 Related Work

Different off-chain networks exist in the blockchain world like Lightening [9] [15] for Bitcoin, Raiden [10] for Ethereum, and the TeeChain network [16] that executes payments asynchronously with respect to the underlying blockchain. Perun [17] is an off-chain 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.

Most previous works on off-chain networks suggest improvements for routing schemes for payment transactions. For example, Flash [12] makes a distinction between Mice, small frequent payments, and Elephant, 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. While it splits elephant payments to maximum k paths and routes them by a modified max-flow algorithm, which finds the maximum amount of flow that the off-chain topology allows from sender to recipient considering the capacity changes of the channels. Spider [11] focuses on improving the number and volume of successfully routed payments on off-chain, 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 [18] 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 [19] is a routing scheme that extends VOUTE [20] for decentralized Path-Based Transaction networks using efficient and flexible embedding-based path discovery depending on the presence of landmark nodes. It is one of the most privacy-preserving algorithms, but it suffers from low payment throughput because it does not consider dynamic channel balance. HushRelay [21] is an efficient privacy-preserving routing scheme, taking into account funds left in each channel while splitting a payment across several paths.

Several works on off-chain networks relate to security and privacy as of [22] which studies inherent limitations of Plasma [23] (an example of an off-chain protocol). [24] identifies and analyzes a type of Denial-of-Service attack based on attracting routes. It exploits the way transactions are routed and executed along the channels of the network in order to attract nodes to route through the attacker. [25] provides an analysis of the trade-offs of fee cost efficiency and route discovery, which should be confidential, in large-scale off-chain networks in which nodes behave strategically. [13] observes that path discovery may harm privacy. In particular, they identify two threats of (i) probing attack (ii) timing attack, related to an active and a passive adversary. Last, [26] proposes a simple but effective, multi-hop, anonymous, and privacy-preserving PCN (MAPPCN).

Beyond blockchain, our work relates to virtual network embedding in communication networks, which deals with the efficient mapping of virtual nodes and virtual links onto the substrate network resources [27] and [28], when to our work only the node mapping can be applicable. Similarly, recent works studied demand aware designs of communication networks [29] and [30] where they suggest how to build a demand aware tree and continuous-discrete network designs, respectively.

3 Fundamentals of Off-chain Topologies

3.1 Model

We model an off-chain network as an undirected graph when nodes are users and an (undirected) edge $e = e_{\{u,v\}}$ illustrate a payment channel between the pair of users connected by the edge, where the edge capacity $c_e = (c_{(u,v)}, c_{(v,u)}) \geq 0$ reflects the amount of tokens locked on that payment channel on both directions: Locked by node u for potential payments to v and by node v for potential payments to u. We refer to the locked tokens on a channel as channel capacity. Two users connected through a direct channel can transfer money between them. 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, and each of its payment channels has enough capacity to serve the payment. For example, if u_1 can transfer x tokens to u_2, when they both have direct channel to u_3 such that $e_{\{u_1,u_3\}}$ is direct channel between u_1 and u_3, $e_{\{u_3,u_2\}}$ is a direct channel between u_3 and u_2, and $c_{(u_1,u_3)}, c_{(u_3,u_2)} \geq x$. The payment can be implemented through a payment of u_1 to u_3 along with a payment of u_3 to u_2. Such a payment path is described as $u_1 \rightarrow u_3 \rightarrow u_2$ and u_3 can be incentivized with a fee for its participation. After serving the payment from u_1 to u_2 the capacity of the payment channels $e_{\{u_1,u_3\}}$ and $e_{\{u_3,u_2\}}$ changes from $(c_{(u_1,u_3)}, c_{(u_3,u_1)})$ and $(c_{(u_2,u_3)}, c_{(u_3,u_2)})$ to $(c_{(u_1,u_3)} - x, c_{(u_3,u_1)} + x)$ and $(c_{(u_2,u_3)} - x, c_{(u_3,u_2)} + x)$.

The graph is called the off-chain topology graph since it describes the users and the 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. We also define a demand matrix $D_{N\times N}$ between N users, that describes the payments pending to be served on the off-chain network. The diagonal of $D_{N\times N}$ equals zero and element $D_{i,j} \geq 0 \ \forall i,j \in [1:N]$ represents the payments from user i to j. For a demand matrix D, we denote by $S_D = \sum_{i,j\in[1:N]} D_{ij}$ the total payments values and by $C_D = \sum_{i,j\in[1:N]} I(D_{ij} > 0)$ the count of distinct payments where $I(\cdot)$ is the indicator function. We need to route payments according to the demand matrix D on the off-chain network topology graph. $D_{N\times N}$ can also be viewed as a directed pending payment graph. In such graph the nodes are users and edges are pending payments. An edge $e_{s,r}$ with a positive weight represents a pending payment of that value from s to r. For instance, the graph in Fig. 1(a) corresponds to a demand matrix D with four rows and columns such that $D_{2,1} = 1, D_{1,3} = 7, D_{1,4} = 2$ and $D_{i,j} = 0$ for other indices, implying $S_D = 10$ and $C_D = 3$.

3.2 Optimization Problems

In an off-chain network, each user has its direct neighbors, those he shares his payment channels with. In our work, we focus on off-chain network topology selection and user mapping. While network topology describes precisely the connection between nodes, nodes mapping demonstrates users' assignments to nodes of the topology. We define two fundamental optimization problems for the off-chain network assuming some given routing scheme for payments. In the definitions a function $\psi(v_1, v_2) \rightarrow \mathbb{R}$ denotes the routing cost between two nodes $v_1, v_2 \in V$ in a topology $G = (V, E)$ following some known routing scheme. Along the paper, we refer to a mapping of users to nodes in the topology which is an injective

function, namely two users are necessarily mapped to different nodes of the topology. In particular, if the number of nodes in the topology equals the number of users ($|V| = N$) the mapping is a bijective function.

Problem 1 Static Mapping Given an off-chain network topology $G = (V, E)$ with $|V| = N$ nodes, a demand matrix $D_{N\times N}$ which describes the pending payments and an evaluation metric $\Psi(D, \psi) \to \mathbb{R}$. The aim is to find a mapping $\phi : [1 : N] \to V$ for each user to a node in the topology graph minimizing the evaluation metric $\Psi(D, \psi)$.

In the context of Problem 1 the evaluation metric refers to the weighted average of the routing cost:

$$\Psi(D, \psi) = \sum_{\forall(i,j)\in[1:N]} D_{ij} \cdot \psi(\phi(i), \phi(j)) \tag{1}$$

In the equation above, $\phi(i)$ is a mapping function, that maps user i to a node in the off-chain topology graph $v \in V$.

Problem 2 Dynamic Topology Given an off-chain topology graph $G = (V, E)$ with $|V| = N$ nodes, a demand matrix $D_{N\times N}$ describing pending payments between N users and an evaluation metric $\Psi(D, \psi) \to \mathbb{R}$. The aim is to dynamically construct an off-chain network through the addition of payment channels to the topology of the graph G. The updated topology should optimally route the C_D payments of D, minimizing the evaluation metric $\Psi(D, \psi)$.

The evaluation metric function for Problem 2 satisfies:

$$\Psi(D, \psi) = \Theta(M) + \sum_{\forall(i,j)\in[1:N]} D_{ij} \cdot \psi_M(i, j). \tag{2}$$

The metric refers to the total cost of opening the new channels and routing the required payments. $\Theta(M)$ refers to the cost to open $M \geq 0$ new channels. The function $\psi_M(v_1, v_2)$ returns the cost of routing a single token from sender node v_1 to recipient node v_2 in the *updated topology graph* (obtained from G after the addition of the M new channels). As a new established channel might be available for a bounded time till its capacity is saturated, we refer to its establishment cost although it can be helpful also in serving additional future payments.

3.3 Evaluation Metrics

In both problems the evaluation metrics of a solution take into account the routing cost function $\psi(v_1, v_2) \to \mathbb{R}$ between two nodes $v_1, v_2 \in V$ in a topology $G = (V, E)$. Thus solving the problems with different cost functions can lead to different off-chain networks even for the same demand matrix. Some examples of routing cost functions can be:

1. Hop count (distance between sender and recipient pairs). Here the function $\psi(v_1, v_2)$ returns the number of edges in a shortest path between nodes v_1, v_2.
2. Transmission fees. Here $\psi(v_1, v_2)$ returns the amount of fees paid to intermediate nodes along a path between nodes v_1, v_2.

For both Problems 1 and 2, if there is no path connecting two nodes v_1, v_2 (they appear in different connecting components) then the functions $\psi(v_1, v_2)$ and $\psi_M(v_1, v_2)$, respectively, take the value of ∞. For Problem 2, the evaluation metric takes into account the cost of opening the additional payment channels. Typically, establishing a payment channel requires the deposit of its two capacities.

For Problem 1 we focus on hop count, trying to find a mapping minimizing the average distance between the sender and recipient pairs. Therefore we aim to minimize the evaluation metric Ψ defined in Eq. (1):

$$\min \Psi(D, \psi) \tag{3}$$

In this case function $\psi(v_1, v_2)$ returns the distance (of the shortest path) between sender node v_1 and recipient node v_2 in topology graph G, and entry D_{ij} represents the number of pending payments from user i to user j.

For example in Fig. 1 when comparing the mapping ϕ_1 in Fig. 1(b) with the mapping ϕ_2 in Fig. 1(c):

$$\Psi_1(D, \psi) = 1 \cdot 1 + 7 \cdot 2 + 2 \cdot 1 = 17$$
$$\Psi_2(D, \psi) = 1 \cdot 2 + 7 \cdot 1 + 2 \cdot 1 = 11$$

The second mapping ϕ_2 implies a shorter average distance.

Problem 1 can be viewed as an instance of the Quadratic Assignment Problem (QAP) problem [34] from the field of facilities location problems.

Definition 1 Quadratic Assignment Problem (QAP) Given are N users and a demand matrix $D_{N\times N}$ where $D_{i,j}$ describes the amount of traffic sent from user i to user j. A graph $G = (V, E)$ with $|V| = N$ is given with a known distance between any pair of nodes. The goal is to map users to nodes such that the weighted average distance is minimized.

(a) Initial State of off-chain topology graph $G = (V, E)$ (b) Adding channel $e_{\{u_1,u_3\}}$ (c) Adding channel $e_{\{u_2,u_3\}}$ (d) Adding channels $e_{\{u_1,u_3\}}$ $e_{\{u_2,u_3\}}$

Fig. 2: Dynamic topology: An initial state of an off-chain network $G = (V, E)$ in Fig. 2(a), and its possible extensions to three different topologies Fig. 2(b)-2(d) to serve the payments from $D_{N\times N}$

	Star	**Ring**	**Expander**	**HyperX**
Parameters			Regularity k	Dimensions $S_1, \ldots, S_L$
Diameter	2	$\frac{1}{2} \cdot N$	$O(\log(N))$	L
Number Of Edges	$N-1$	N	$\frac{1}{2} \cdot N \cdot k$	$\frac{1}{2} \cdot N \cdot \sum_{i=1}^{L}(S_i - 1)$

Table 1: Topology graph $G = (V, E)$ with $|V| = N$ nodes

QAP is an NP-hard problem and moreover it was shown that there are no approximation algorithms for any constant factor [35]. A common assumption is that finding optimal solutions to QAP problems of size greater than 15 is impossible. In the case of evaluating the distance between two nodes based on hop count, all distances are integers and are affected by the location of the nodes unlike the allowed distances in the QAP problem that can be arbitrary. Besides, we focus on particular topologies rather than general graphs. This allows developing solutions carefully designed for particular topologies and despite the hardness of the general problem we can derive guarantees on the solutions for some topologies.

We optimize Problem 2 while referring to the cost $\Theta(M)$ of opening M new channels, which can be expensive, since opening a new channel requires an on-chain transaction with higher fees, in addition to the routing cost of payments. Here, we measure the routing cost based on the second evaluation metric considering the routing fees in the obtained topology following the addition of the new channels. Function $\Theta(M)$ refers to the fee required to open $M \geq 0$ new channels and function $\psi_M(v_1, v_2)$ to fees for routing a single token from sender node v_1 to recipient node v_2 in the obtained topology.

We look at the example shown in Fig. 2 for Problem 2. Given the initial state Fig. 2(a) of the topology with three nodes (users) $\{v_1, v_2, v_3\} \in V$, one single payment channel $e_{\{v_1,v_2\}} \in E$, and two pending payments one from v_1 to v_3 of 5 tokens, the other of single token from v_2 to v_3, $D = \begin{pmatrix} 0 & 0 & 5 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$. Assuming routing fee of 0.05 tokens through channel and 2 token fee for opening a new channel. We compare three topologies from Fig. 2(b)-2(d):

$$\tilde{\Psi}_1(D, \psi) = 2 + 5 \cdot 0.05 + 1 \cdot 0.05 \cdot 2 = 2.35$$
$$\tilde{\Psi}_2(D, \psi) = 2 + 5 \cdot 0.05 \cdot 2 + 1 \cdot 0.05 = 2.55$$
$$\tilde{\Psi}_3(D, \psi) = 2 \cdot 2 + 5 \cdot 0.05 + 1 \cdot 0.05 = 4.3$$

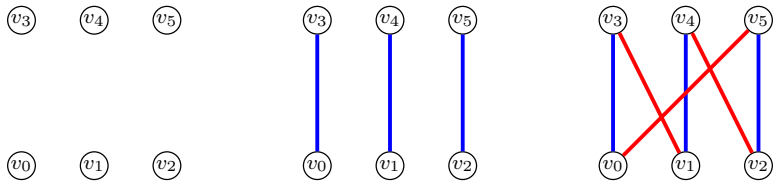
As we see, in case of initial topology Fig. 2(a) and demand matrix D, the best subsequent topology would be the topology described in Fig. 2(b).

4 Topologies for Off-chain Networks

Off-chain topologies and communication networks can both be restricted in their service due to their structure or potential link capacities. Inspired by that similarity, we overview common communication network topologies for the off-chain usecase. In particular, we refer to each topology to its number of edges and the diameter (maximal distance between a pair of nodes). Table 1 summarizes the qualities of the topologies above. In the following sections, we detail mapping algorithms for such topologies and examine their suitability through evaluating their performance

(i) The ***star*** topology - This is a simple basic network topology in which all the network nodes are individually connected to a central point of communication. When a node wants to communicate with another node, it passes on the data to the central hub and the central hub forwards the data to the destination. It has a diameter of 2 and $N - 1$ edges connect the N nodes. Previous work [3] [4] 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 topology also benefits from simple low routing complexity and can enjoy low routing fees.

(ii) The ***ring*** topology - In this topology, nodes create a circular data path. Each node is connected to its two adjacent nodes, like points on a circle. In this topology data travels from sender to recipient through intermediate nodes, clockwise



(a) Expander Nodes (b) First iteration (c) Second iteration

Fig. 3: Illustration of construction of a $2-$regular Random Bipartite Expander (Algorithm 1) with six nodes through $k = 2$ iterations.

or anti-clockwise until it reaches the recipient. For N nodes the diameter is $\frac{N}{2}$ and the number of payment channels in the topology equals the number of nodes N.

Algorithm 1: Random Bipartite Expander Construction Algorithm

Input:
N: number of nodes in $G = (V, E)$
k: node degree
Output:
$H_{N\times k}$: neighbors matrix of $G = (V, E)$
initialize $index[]$ with indexes from 1 to N
for $j \leftarrow 1$ ***to*** k **do**
  randomly permute the second half of $index[]$, making sure that no $index[i]$ holds one of its previous values
  for $i \leftarrow 1$ ***to*** $\frac{N}{2}$ **do**
    $H[i][j] = index[\frac{N}{2} + i]$
    $H[index[\frac{N}{2} + i]][j] = i$

(iii) ***Expanders*** are graphs conserving two main properties: sparseness and high connectivity. This is equivalent to having a relatively small overall number of edges that allow short distances between nodes. An undirected graph $G = (V, E)$ in which every relatively small subset S of vertices expands quickly, in the sense that it is connected to many vertices in the set $V \setminus S$ of complementary vertices, is a good expander.

In this paper, when constructing expanders we focus on $k-$regular graphs. In such graphs, we measure expansion parameter according to spectral property, which is an algebraic definition based on the eigenvalues of the adjacency matrix $A(G)$ that defines the graphs, where $A_{i,j} = A_{j,i}$ is the number of edges between vertices i and j with values in the range $[0, k]$. The spectral gap is the difference between the absolute values of the two largest eigenvalues of matrix $A(G)$, and in $k-$regular graph case, it equals $k - |\lambda_2|$, when λ_2 is the second largest eigenvalue. For a better expander, we need to maximize the spectral gap thus minimize $|\lambda_2|$. In particular, a special family of $k-$regular graphs known as Ramanujan [5] are the best spectral expanders with the largest spectral gap. Intuitively, Ramanujan graph a is $k-$regular graph where all eigenvalues of its adjacency matrix, except for $\pm k$, satisfy $|\lambda_i| \leq 2 \cdot \sqrt{k-1}$.

We choose to work with bipartite $k-$regular random graphs. For larger graphs with a higher number of nodes, the bipartite $k-$regular random graphs generate good expander that is almost Ramanujan [6]. Algorithm 1 describes the construction of the bipartite random expanders we used. The matrix $H_{N\times k}$ lists for each user i its k neighbors that are randomly permuted k times. Fig. 3 illustrates a construction of $2-$regular random bipartite graph following the algorithm.

(iv) ***HyperX*** topology (described by Ahn et al., 2009 [7]) organizes nodes in a lattice of L dimensions. The HyperX topology is common in the design of datacenter communication networks [8]. The number of nodes in a dimension can vary and is denoted by $S_i \geq 2$ for dimension $i \in [1, L]$ such that without loss of generality $S_1 \leq S_2 \leq \ldots \leq S_L$. A node is represented by an integer coordinate vector $X = (x_1, \ldots x_L)$ such that $x_i \in [0, S_i - 1]$ and the total number of nodes is $N = \prod_{i=1}^{L} S_i$.

Connectivity: Each node is connected to all other nodes by which it differs by (exactly) a single value of the coordinate vector. Links are all bidirectional. In particular, a node is connected to $S_i - 1$ other nodes from which it differs in the value of dimension i. We denote by d the total number of nodes each node is connected to, satisfying $d = \sum_{i=1}^{L}(S_i - 1)$. The total number of edges in the topology is $\frac{1}{2} \cdot N \cdot d = \frac{1}{2} \cdot \prod_{i=1}^{L} S_i \cdot \sum_{i=1}^{L}(S_i - 1)$. The diameter of the topology is the number of dimensions L. In a regular HyperX topology, the number of nodes in a dimension is fixed $S_1 = \ldots = S_L = S$ such that $N = S^L$. Here the diameter satisfies $L = log_S(N)$.

Fig. 4 illustrates a HyperX topology of $L = 3$ dimensions with $S_1 = 2$, $S_2 = S_3 = 3$ and $N = 2 \cdot 3 \cdot 3 = 18$ nodes. Links along three dimensions are shown in dashed red, dotted black and solid blue. Each node is connected to $d = \sum_{i=1}^{L}(S_i - 1) = (2-1)+(3-1)+(3-1) = 5$ other nodes. The number of edges is $\frac{1}{2} \cdot \prod_{i=1}^{L} S_i \cdot \sum_{i=1}^{L}(S_i - 1) = \frac{1}{2} \cdot (2 \cdot 3 \cdot 3) \cdot (1+2+2) = \frac{1}{2} \cdot 18 \cdot 5 = 45$.

5 Static Mapping and Dynamic Topology Solutions Design

5.1 Overview of the Algorithms

Problem 1 looks for improving off-chain performance by enhancing the placement of the users in a given off-chain topology. We achieve that by mapping pairs of users with a high interaction to close nodes in the topology graph. Our Algorithm's general structure consists of the input *(i)* Topology *(ii)* Demand matrix / Pending payment graph. Its output includes *(i)* User mapping, *(ii)* Routing scheme. In such a scheme we adopt a constant cost function that refers to the distance between nodes, reflecting the behavior of the evaluation metric. We design various mapping algorithms for the topologies of a star, ring, and expander. The output of our mapping algorithm is an off-chain network where each user is mapped to a single node in the given topology, as well as a routing scheme that specifies how to route payments in the off-chain network.

In Problem 2 one has to dynamically construct an off-chain network topology aiming to minimize the total spent fees. We attain that by adding channels to the network topology that minimizes the fees spent while serving payments off-chain. The general structure of our Algorithm is as follows: Input of *(i)* Topology *(ii)* Demand matrix / Pending payment graph. Output of *(i)* Updated topology, *(ii)* Routing scheme. We consider constant cost function that reflects the evaluation metric, computing fees for serving demand payments.

For both Problem 1 and 2 we assume a given network topology $G(V, E)$, and a demand matrix $D_{N\times N}$, when $\forall i, j \in [1 : N] \rightarrow D_{i,j}$ describes the number of payments from i to j, in Problem 1, and the amount of tokens in payments from i to j, in Problem 2. To simplify the problems we assume shortest path routing, for future work we suggest studying mapping and topology problems with more complicated routing schemes.

At the first iteration of off-chain construction, all the users have the information about the payments pool from the data available on-chain, therefore, construction the off-chain can be completed in a decentralized manner, where each user runs the related algorithm locally with the same payments pool as other users. To enable updating the off-chain network topology dynamically in a decentralized manner, we let each user collect information of all pending payments (rather than only those the user is involved in). This allows computing the algorithm similarly by all users so that each user is aware of the same required update of the topology.

5.2 Traffic Aware User Mapping Algorithm

For each of the three topologies, we develop a concurrent mapping algorithm that improves average path length.

For **star** we map the user of the highest number of payments to the center of the star, see Algorithm 2. We explain that such a mapping is indeed optimal among all mappings for the topology, namely, it minimizes the value of the cost function.

Theorem 1 *The mapping found by Algorithm 2 finds an* optimal *mapping over all mappings for the star topology.*

Proof. If $center$ is the index of the user mapped to the topology center, the value of the evaluation metric function for star topology is:

$$\Psi(D, \psi) = 2 \cdot S_D - \Big(\sum_{i\in[1:N]} D_{center,i} + \sum_{i\in[1:N]} D_{i,center} \Big).$$

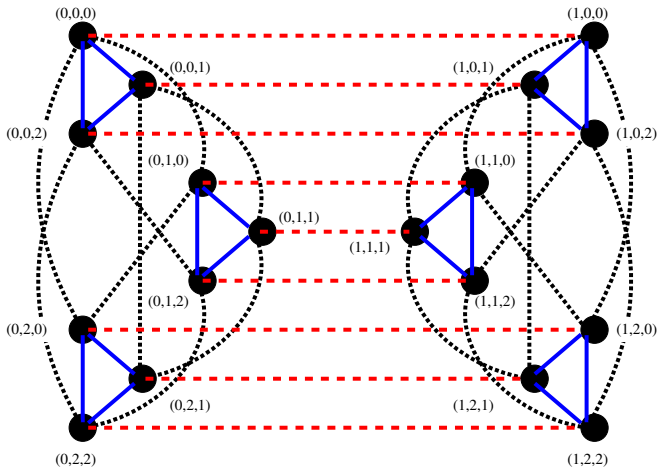


Fig. 4: HyperX topology illustration of $L = 3$ dimensions with $S_1 = 2, S_2 = S_3 = 3$. There are $N = \prod_{i=1}^{L} S_i = 2 \cdot 3 \cdot 3 = 18$ nodes, each represented by three coordinates. Links along the different dimensions are shown in different colors and line styles. Each nodes is connected to $d = \sum_{i=1}^{L}(S_i - 1) = (2 - 1) + (3 - 1) + (3 - 1) = 5$ other nodes.

Algorithm 2: Star Mapping Algorithm

Input:
Star Topology Graph $G = (V, E)$ with $|V| = N$
Demand Matrix $D_{N \times N}$
Output:
User→Node Mapping function ϕ
$degree[i] \leftarrow 0$
for $j \leftarrow 1$ ***to*** N **do**
  $degree[j] \leftarrow \sum_{i \in [1:j]} (D_{i,j} + D_{j,i})$
$maxD \leftarrow$ User of highest degree in $degree[]$
$j \leftarrow 1$
for $i \leftarrow 1$ ***to*** N **do**
  if $i == maxD$ **then**
    $\phi(maxD) = v_0$ (central node)
  else
    $\phi(i) = v_j$
    $j \leftarrow j + 1$

Algorithm 3: Ring Mapping Algorithm

Input:
Ring Topology Graph $G = (V, E)$ with $|V| = N$
Demand Matrix $D_{N \times N}$
Output:
User→Node Mapping Function ϕ
$M \leftarrow \emptyset$ (mapped users), $Q \leftarrow \emptyset$ (mapped nodes)
$pairDegree[(i, j)] \leftarrow 0$
foreach $j \in [1 : N]$ *and* $i \in [1 : j]$ **do**
  $pairDegree[(i, j)] \leftarrow (D_{i,j} + D_{j,i})$
$pairs[] \leftarrow$ sorted $pairDegree[]$ in decreasing order
foreach *pair* $u_1, u_2 \in pairs[]$ **do**
  if $u_1, u_2 \notin M$ **then**
    $nodeID \leftarrow Random()$
    while $v_{nodeID} \in Q \parallel v_{nodeID+1} \in Q$ **do**
      $nodeID \leftarrow Random()$
    $\phi(u_1) \leftarrow v_{nodeID}$
    $\phi(u_2) \leftarrow v_{nodeID+1}$
  else if $u_1 \in M$ *and* $u_2 \notin M$ **then**
    $nodeID \leftarrow$ closest ID to $\phi(u_1)$ & $v_{ID} \notin Q$
    $\phi(u_2) \leftarrow v_{nodeID}$
  else if $u_1 \notin M$ *and* $u_2 \in M$ **then**
    $nodeID \leftarrow$ closest ID to $\phi(u_2)$ & $v_{ID} \notin Q$
    $\phi(u_1) \leftarrow v_{nodeID}$
  $M = M \cup \{u_1\}$ and/or $M = M \cup \{u_2\}$
  $Q = Q \cup \{v_{nodeID}\}$ and/or $Q = Q \cup \{v_{nodeID+1}\}$

While S_D is fixed for all mappings, by locating as the center of the topology the user with maximum payments, we maximize $(\sum_{i \in [1:N]} D_{center,i} + \sum_{i \in [1:N]} D_{i,center})$ and minimize $\Psi(D, \psi)$.

For **ring** we make sure to map pairs with the highest common number of payments to direct neighbors when possible. Otherwise, when one of the users is already mapped, we try to find the closest node to it and map the other user, as described in Algorithm 3.

For **expander** we examined multiple $k-$regular random expander where each user has k neighbors, here again, we make sure to map pairs of users of high interaction to neighbor nodes in the topology graph, and in case it is not possible we randomly map them to other, being aware of the fact that in good expanders nodes exist within a short distance in high probability. First we sort in, decreasing order, all pairs of users according to their communication degree ($degree[(i, j)] = D_{i,j} + D_{j,i}$). Then we go over the sorted pairs, in case both users are still not mapped yet, we try to find them a pair of neighbor nodes in the topology graph and map them to these nodes, if we are not able to find such nodes, we map them to random nodes and continue to the next pair. otherwise, when one of the users is already mapped we try to find a neighbor node of this user to map the other user, when no such no exists we map the other user to a random node and continue to the next pair, see Algorithm 4.

The mapping algorithm for **HyperX** can be found in Algorithm 5. We assign users to nodes in the topology iteratively. We go over nodes in the topology according to the order of the dimensions. For each node we select the user that benefits the most from locating it in the node according to the distribution of its payments demand to each of previously assigned users. While the diameter of the topology is L, the distance to previously assigned nodes is in the range $1, \ldots, L$. Intuitively, we select to assign to the current node the user that sends a large portion of its traffic to users located close to the considered node of the topology.

Algorithm 4: Expander Mapping Algorithm

Input:
Expander Topology Graph $G = (V, E)$ with $|V| = N$
Demand Matrix $D_{N \times N}$
Output:
User→Node Mapping function ϕ
$M \leftarrow \emptyset$ (mapped users), $Q \leftarrow \emptyset$ (mapped nodes)
$pairDegree[(i,j)] \leftarrow 0$
foreach $j \in [1:N]$ *and* $i \in [1:j]$ **do**
  $pairDegree[(i,j)] \leftarrow (D_{i,j} + D_{j,i})$
$pairs[] \leftarrow$ sorted $pairDegree[]$ in decreasing order
foreach *pair* $u_1, u_2 \in pairs[]$ **do**
  if $u_1, u_2 \notin M$ **then**
    Find neighbors $v_1, v_2 \in V$ and $v_1, v_2 \notin Q$
    if $!v_1$ *and* $!v_2$ **then**
      $v_1, v_2 = Random() \notin Q$
    $\phi(u_1) \leftarrow v_1$
    $\phi(u_2) \leftarrow v_2$
  else if $u_1 \in M$ *and* $u_2 \notin M$ **then**
    $v_2 =$ Find neighbor of $\phi(u_1) \notin Q$
    if $!v_2$ **then**
      $v_2 = Random() \notin Q$
    $\phi(u_2) \leftarrow v_2$
  else if $u_1 \notin M$ *and* $u_2 \in M$ **then**
    $v_1 =$ Find neighbor of $\phi(u_2) \notin Q$
    if $!v_1$ **then**
      $v_1 = Random() \notin Q$
    $\phi(u_1) \leftarrow v_1$
  $Q = Q \cup \{v_1\}$ and/or $Q = Q \cup \{v_2\}$
  $M = M \cup \{u_1\}$ and/or $M = M \cup \{u_2\}$

Algorithm 5: HyperX Mapping Algorithm

Input:
HyperX Topology Graph $G = (V, E)$ with dimensions $S_1, \ldots, S_L$ and $N = \prod_{i=1}^{L} S_i$ nodes
Demand Matrix $D_{N \times N}$
Output:
User→Node Mapping Function ϕ
$M \leftarrow \emptyset$ (mapped users), $Q \leftarrow \emptyset$ (mapped nodes)
$pairDegree[(i,j)] \leftarrow 0$
foreach $j \in [1:N]$ *and* $i \in [1:j]$ **do**
  $pairDegree[(i,j)] \leftarrow (D_{i,j} + D_{j,i})$
while $Q \neq V$ **do**
  set $v \in (V \setminus Q)$ *as the next node in the topology*
  $A = \emptyset$
  foreach *user* $u_1 \notin M$ **do**
    $w_{u_1} = \sum_{u_2 \in M} pairDegree[u_1, u_2]$
    if $w_{u_1} > 0$ **then**
      $\psi_{u_1} = \sum_{u_2 \in M} pairDegree[u_1, u_2] \cdot \psi(v, \phi(u_2))$
      $A = A \cup \{(u_1, \frac{\psi_{u1}}{w_{u1}})\}$
  if $A \neq \emptyset$ **then**
    set u *as the user with minimum weighted cost* $\frac{\psi_u}{w_u}$ *in* A
  else
    select u *randomly from* $U \setminus M$
  $\phi(u) \leftarrow v$
  $Q = Q \cup \{v\}$
  $M = M \cup \{u\}$

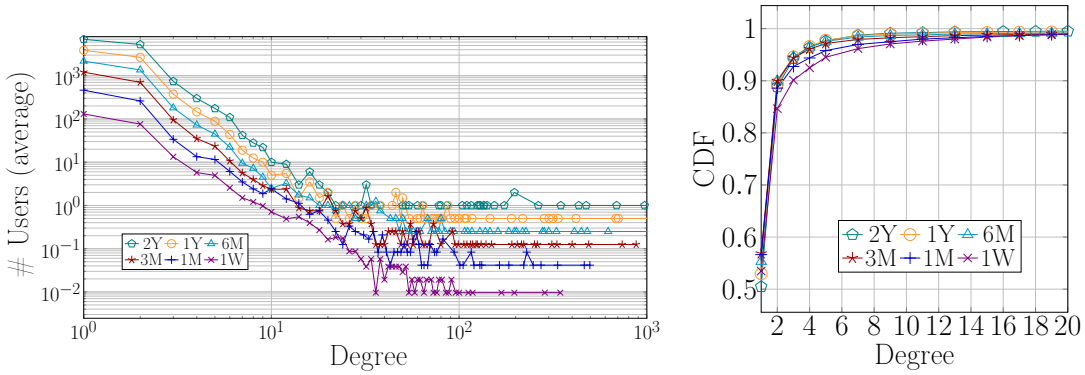
5.3 Demand Aware Dynamic Topology Algorithm

Another aspect for improving the off-chain performance is by constructing a topology based on the demand of the users and the tokens flowing in the network. This is what we study in Problem 2. Our approach for dynamically constructing the off-chain networks is as follows. For a given payment (or multiple of them) to serve, we compare the cost of serving it in the current network with the cost of establishing a new payment channel. A new channel is added when its addition is expected to reduce the total cost.

In Algorithm 6 we assume a constant fee of F for opening a new payment channel and a fee of f for routing a single token through a channel in a multi-hop path. For each payment between two users we compare the cost of using an existing path to route it to the cost of opening a new channel and routing through it, we assume users do not pay fees when routing

Algorithm 6: General-Topology Dynamic off-chain Algorithm

Input:
Topology Graph $G_{in} = (V, E_{in})$: $|V| = N$
Demand Matrix $D_{N\times N}$
Cost Function $\psi(v_1, v_2)$
Output:
Topology Graph $G_{out} = (V, E_{out})$: $|V| = N$
Data: defined channel opening fee F
$pairTokens[(i,j)] \leftarrow 0$
foreach $j \in [1:N]$ *and* $i \in [1:j]$ **do**
  $pairTokens[(i,j)] \leftarrow (D_{i,j} + D_{j,i})$
$pairs[] \leftarrow$ sorted $pairTokens[]$ in ascending order
$E_{out} \leftarrow E_{in}$
foreach *pair* $v_i, v_j \in pairs[]$ **do**
  $P =$ Find Path between v_i, v_j
  if $P == \emptyset$ **then**
    $E_{out} \leftarrow E_{out} \cup e_{\{v_i,v_j\}}$
  if $\psi(v_i, v_j) \cdot D_{i,j} + \psi(v_j, v_i) \cdot D_{j,i} \geq F$ **then**
    $E_{out} \leftarrow E_{out} \cup e_{\{v_i,v_j\}}$
$G_{out} = (V, E_{out})$



(a) Raiden Users' Degree (b) User Degree CDF

Fig. 5: Raiden Data for various epoch lengths

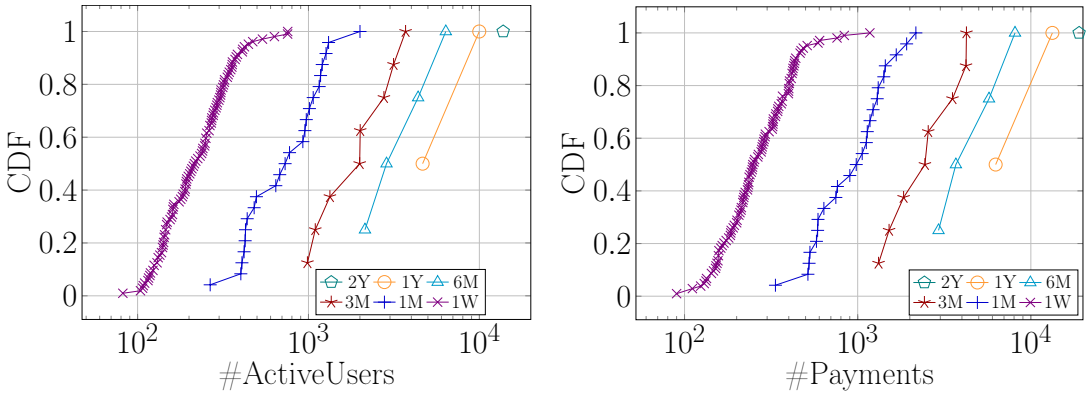
through direct channels. In this algorithm $\psi(v_1, v_2)$ is updated after adding each channel. We prioritize smaller payments when adding new payment channels, by [12] most payments are small and many small payments reoccur.

6 Raiden Data Analysis

In this section, we study Raiden Network [10], the off-chain network of Ethereum [2]. Since the performance and cost of an off-chain network are affected by its number of users and payment channels we would like to examine their typical values in time epochs of various lengths. We collected Raiden data of two years between May 1, 2018, and April 30, 2020. The data included 13782 different users that performed 74804 payments involving 19575 various pairs of users.

We analyzed the Raiden data by categorizing it into disjoint equal-length epochs of 1-week, 1-month, 3-months, 6-months, 1-year, and the whole period of 2-years. For each epoch, we define an undirected graph $G_R = (V_R, E_R)$ based on the observed payments. V_R are the active users, participating in at least one payment during the epoch. An edge $e_{\{u_1,u_2\}}$ for $u_1, u_2 \in V_R$, belongs to E_R only if u_1, u_2 have a joint payment. Fig. 5(a) describes the number of neighbors (degree) each user has in G_R, when user u_1 is a neighbor of user u_2 only if $e_{\{u_1,u_2\}} \in E_R$. The user degree follows a power-low distribution, like in Ethereum [31] and Lightning [32], meaning the probability for a user degree k is $P(k) = k^{-\alpha}$ for $\alpha > 0$. The majority of the users transfer tokens only to very few users, and a minority of users interact with many others, similar to Ethereum results [33]. From studying the CDF of the users' degree Fig. 5(b) we learn that most users have a low degree, with 94.5-95.7% of them with at most five neighbors for 1-week and 1-month.

Fig. 6(a) and 6(b) show the CDF of the number of active users and the number of payments, respectively, in each epoch. For instance, the median numbers of active users for 1-week and 1-month are roughly 217 and 752, involving only a small portion of the total number of 13782 distinct users. We noticed a reduction (not shown in the graph) in the number of active users within each epoch in later parts of the two-year data. Fig. 7 describes the number of connected component of G_R in a single epoch, when looking only at the active users and their payments at that epoch, we do not see a reduction in the number of connected components for longer periods, and this is because many new users join the network constantly. After two years the 13782 users are structured in 873 connected components (2-year epoch).



(a) Raiden Number of active users (b) Raiden Number of payment

Fig. 6: Raiden Number of payments and users for various epoch lengths

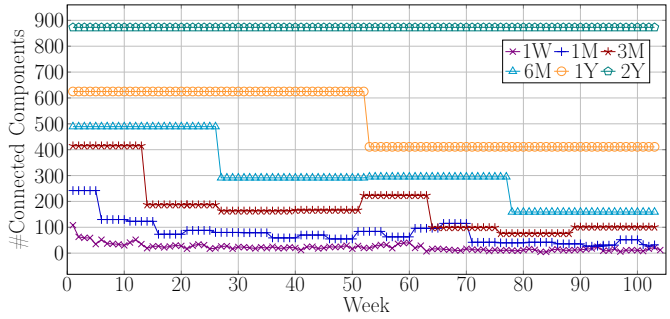


Fig. 7: Raiden connected components number calculated at each epoch, for all epoch lengths

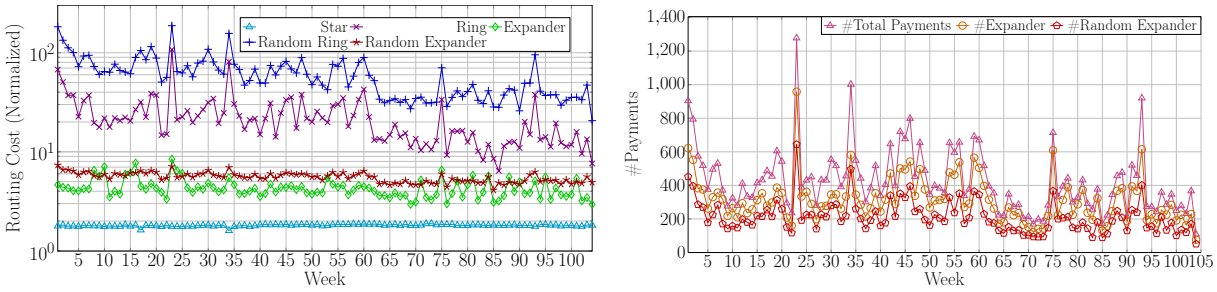
The three figures above Fig. 6(a), 6(b) and 7 help us calculate, at epoch t of some length, the minimal number of off-chain channels needed to serve off-chain all the epoch's payments. If U active users are organized in C connected components (as implied by a set of payments), then at least $U - C$ channels are needed to successfully route all payments. For example, following the above Raiden values, for serving all the 19575 different payments in the two years, at least $13782-873 = 12909$ off-chain channels are needed.

7 Experimental Evaluation of the Algorithms

For the evaluation of the algorithms from Section 5, we use the Raiden data of two years described in Section 6. The algorithms receive as part of the input the demand matrix D. We refer to two-time epochs, the *learning period* and the *evaluation period*. The input demand matrix is computed based on the payments in the learning period. Payments from the evaluation period are used to evaluate the performance of the solutions. A payment request can be served off-chain only if its both users are mapped to nodes in a topology $G = (V, E)$ and a path connecting them exists in E. First, we examine the case where learning and evaluation periods refer to the same epoch. In this scenario, all the involved users in the payments to be routed are mapped to nodes, and accordingly, in connected topologies such as star or ring, we can route all payments. However, for an expander topology, this does not necessarily hold since it might include more than a single connected component. To be able to serve payments of some epoch without a delay, we also study the practical scenario where the learning period precedes the evaluation period without an overlap.

We start with solutions to Problem 1 with *star*, *ring* and *expander*. Fig. 8(a) and 9(a) present the performance of serving off-chain payments during epochs of single week in different topologies. The star, ring and expander (with degree $k = 2$) are built based on the algorithms presented above. For a fair comparison, the number of channels in the topologies is almost similar with N channels in the ring and expander and $N - 1$ for the star. Besides, as a baseline, we evaluate random ring and random expander topologies for which the mapping of users to nodes is done randomly (regardless of the demand matrix).

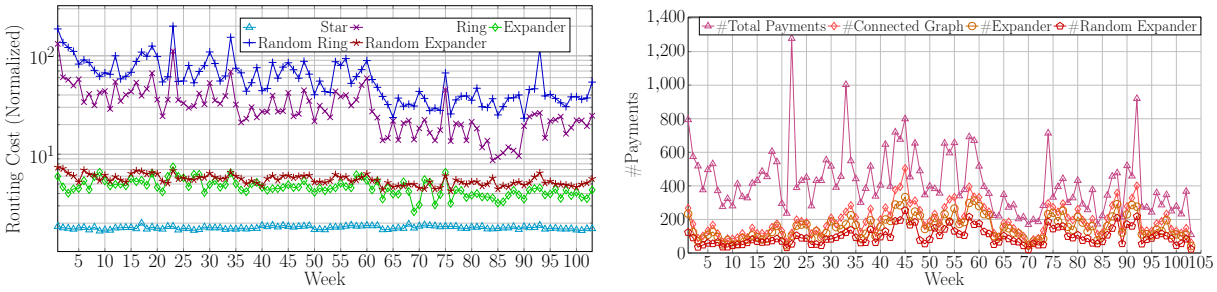
In Fig. 8 the performance is computed for the case that the learning and evaluation collide where Fig. 8(a) refers to routing cost and 8(b) to success ratio. By Fig. 8(a) the average cost per payment for ring is 22.66, for expander 4.34 and for star only 1.8.Random mappings of users achieve poor performance with a cost increase of 184.1% and 31.9% to the results of Algorithms 3 and 4 for ring and expander, respectively. Fig. 8(b) shows the percentage of payments successfully served by expander topologies. For expander, the percent of successfully routed payments ranges between 57.4% and 89.6% (mean of 74.5%), and for random expander between 35% and 60.4% (mean 49.6%).



(a) Off-chain Routing cost per successfully served payments

(b) The number of payments successfully routed off-chain

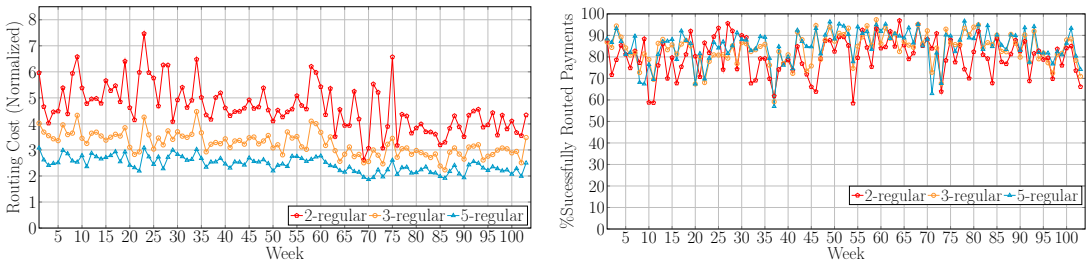
Fig. 8: Raiden payments routed on Star, Ring and 2-regular Expander (with N channels in Ring and Expander, $N-1$ channels in Star). Learning and evaluation periods collide, one week epoch.



(a) Off-chain Routing cost per successfully served payments

(b) The number of payments successfully routed off-chain

Fig. 9: Raiden payments routed on Star, Ring and 2-regular Expander (with N channels in Ring and Expander, $N-1$ channels in Star). The learning period precedes evaluation period in one week.



(a) Routing Cost of Bipartite $\{2,3,5\}$-Regular Random Expanders.

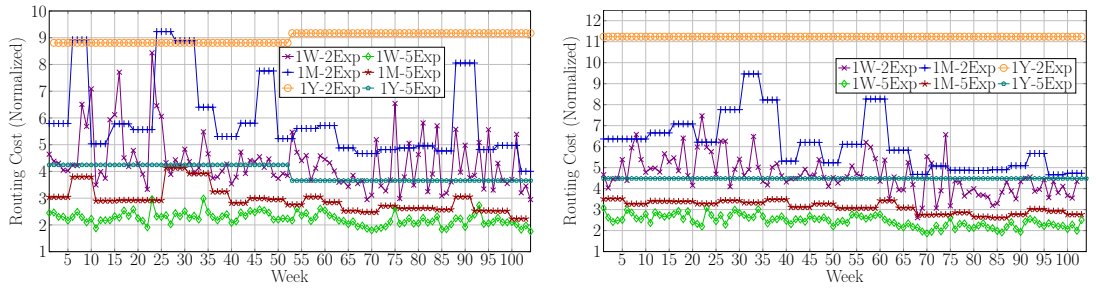
(b) Successfully Routed payments percent of the mapped users' payments, through Bipartite $\{2,3,5\}$-Regular Random Expanders

Fig. 10: Bipartite $\{2,3,5\}$-Regular Random Expanders comparison, in epochs of one week, learning period preceding evaluation period

Fig. 9(a) examines the case for which the learning period precedes the evaluation period. Here a user that first appears in the evaluation period is not mapped to the topology, eliminating the ability to serve its payments. This is in addition to potential failures by mappings users to nodes in different connected components. As expected we observe an increase in the routing cost and a degradation in the percentage of served payments in Fig. 9(a) vs. Fig. 8(a). The performance degradation is explained due to differences in the payments demands in the two periods of learning and evaluation. For example, some pairs of users showed high interaction in the evaluation period without an early indication for that in the learning period. In this scenario, the average cost per payment for ring is 31.54, for expander 4.64 and star 1.8. Interestingly, an increase in the number of requested payments was observed in week 21. This implies an increase in the routing cost for both scenarios since it is challenging to optimize the mapping for a larger number of payments in the demand. Note that the performance of the star topology was almost not affected by the change due to its low diameter.

Fig. 9(b) shows the total number of payments in each epoch and the number of successfully served payments off-chain. Connected topologies (star, ring) show better percents ($9.3\% - 82.2\%$, mean 47.3%) than expander topology ($8\% - 72.2\%$, mean 38%), since in a connected topology the only reason for unsuccessful serving is unmapped users. In late epochs the number of successfully routed payments is closer to the total number of payments which indicates that the active users are more stable among epochs.

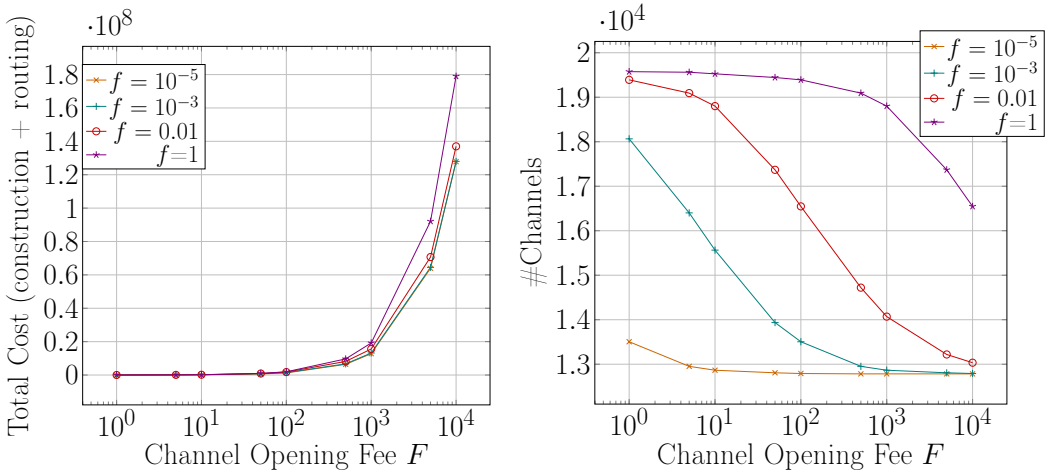
The last comparisons assumed for the expander topology a degree of $k = 2$ to ensure almost identical numbers of edges in all topologies. To analyze the impact of the expander degree in the topology graph on off-chain performance, we present Fig. 10 comparing the performance of $\{2,3,5\}$-regular random expanders in epochs of one week. Higher degree ensures



(a) $\{2,5\}$-regular Expanders, assuming colliding learning and evaluation periods

(b) $\{2,5\}$-regular Expanders, assuming learning period precedes evaluation period

Fig. 11: Routing Cost of Raiden payments on $\{2,5\}$-regular expander topology in various period lengths



(a) Cost of constructing $G = (V, E)$ and routing $D_{N\times N}$

(b) Number of opened channels

Fig. 12: Dynamically constructing $G = (V, E)$ for various fees for opening a channel (F) and routing a single token (f)

better graph connectivity, therefore the bipartite 5-regular random expander enjoys the best performance. First, Fig. 10(a) examines the routing costs with average values of 4.64, 3.23 and 2.45 for $k = 2, 3$ and 5, demonstrating a reduction of 30.38% for $k = 3$ and an even larger reduction of 47.19% for $k = 5$ vs. the cost for $k = 2$. In Fig. 10(b) we show the percentage of the successfully routed payments out of the mapped users' payments, taking into account only payments without new users. We see that mostly the percentage on 5-regular expander is higher than the others. The average is 80.12%, 83.9% and 85.17%, for 2-regular, 3-regular and 5-regular expander, respectively.

Fig. 11 illustrates comparing of normalized routing cost of $\{2,5\}$-regular random expanders when looking at learning and evaluation periods of different length: 1-week, 1-month and 1-year that impacts the frequency new mappings are computed using Algorithm 4. For both expanders, we see that for shorter epochs the routing cost is mostly lower, especially for one week, in both case, when the learning period collides with the evaluation period Fig. 11(a) and when the learning period precedes the evaluation period Fig. 11(b). For example, for an epoch of 1-week the average cost is 4.64 and 2.45 per payment for 2-regular and 5-regular expanders, respectively (in Fig. 11(b)). For 1-month the costs increase to 6.07 and 3.09 for these two degrees of the expander. A higher degree ensures better graph connectivity, therefore the bipartite 5-regular random expander enjoys the best performance in all periods, as illustrated on Fig. 11. For example, in Fig. 11(b) for a period of 1-week, routing costs average values of 4.64 and 2.45 for $k = 2$ and 5, demonstrating a reduction of 47.19% for $k = 5$ vs. the cost for $k = 2$.

Last, we evaluate Algorithm 6 for Problem 2. We compute the demand matrix $D_{N\times N}$ based on all the two-years data, in this case, total payments generate demand matrix with $C_D = 19575$ (as the number of distinct pairs, smaller than the actual number of payments). Here we assign a range of values for F (opening a new channel fee) and f (routing a single token fee), and plot the total cost spent for serving 19575 different payments (in Fig. 12(a)) and the total number of channels in $G = (V, E)$ (in Fig. 12(b)). Since the ratio between the unique pairs $C_D = 19575$ and the number of users 13782 is small (roughly 1.42), we can see from Fig. 12(a) that the cost of constructing an off-chain network and routing its payments depends on the fee of opening a channel, not on the routing fee. On the other hand from Fig. 12(b) we observe that the number of channels in the off-chain network is highly influenced by the routing fee f. In particular, for relatively low routing fees concerning the channel opening fee, the constructed topology includes fewer channels.

8 Conclusions

In the paper, we studied the impact of topologies on the performance of off-chain blockchain networks and positioned that their construction should be demand-aware. We presented two natural optimization problems of finding the mapping of users to an existing topology and the construction of topology based on the demand for payments to be served. Our findings based on analyzing real off-chain Raiden data emphasize the importance of connectivity in the topologies and the ability to use characteristics of the payments for the topology design to achieve good performance.

References

1. Satoshi Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Bitcoin white paper, 2008
2. Gavin Wood, Ethereum: A secure decentralised generalised transaction ledger, https://gavwood.com/paper.pdf, Ethereum project yellow paper, 2014
3. Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry and Arthur Gervais, SoK: Layer-Two Blockchain Protocols, Financial Cryptography and Data Security (FC), 2020
4. Georgia Avarikioti, Gerrit Janssen, Yuyi Wang and Roger Wattenhofer, Payment network design with fees, Springer Int. Workshop on Data Privacy Management, Cryptocurrencies and Blockchain Technology, 2018
5. Alexander Lubotzky, Ralph Phillips and Peter Sarnak, Ramanujan graphs, Combinatorica, 1988,
6. Salil P. Vadhan, Pseudorandomness, Foundations and Trends in Theoretical Computer Science, 1–336, 2012
7. Jung Ho Ahn, Nathan L. Binkert, Al Davis, Moray McLaren and Robert S. Schreiber, HyperX: Topology, routing, and packaging of efficient large-scale networks, ACM/IEEE Conference on High Performance Computing Networking, Storage and Analysis (SC), 2009
8. Jens Domke, Satoshi Matsuoka, Ivan R. Ivanov, Yuki Tsushima, Tomoya Yuki, Akihiro Nomura, Shin'ichi Miura, Nie McDonald, Dennis L. Floyd and Nicolas Dubé, HyperX topology: First at-scale implementation and comparison to the fat-tree, ACM/IEEE Conference on High Performance Computing Networking, Storage and Analysis (SC), 2019
9. Joseph Poon and Thaddeus Dryja, The Bitcoin Lightning network: Scalable off-chain instant payments, 2016
10. Raiden network, http://raiden.network/, 2017
11. Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia C. Fanti and Pramod Viswanath, Routing Cryptocurrency with the Spider Network, ACM HotNets, 2018
12. Peng Wang, Hong Xu, Xin Jin and Tao Wang, Flash: Efficient dynamic routing for offchain networks, ACM CoNEXT, 2019
13. Utz Nisslmueller, Klaus-Tycho Foerster, Stefan Schmid and Christian Decker, Toward Active and Passive Confidentiality Attacks On Cryptocurrency Off-Chain Networks, arXiv preprint 2003.00003, 2020
14. Rami Khalil, A Zamyatin, G Felley, P Moreno-Sanchez and A Gervais, Commit-Chains: Secure, Scalable Off-Chain Payments, Cryptology ePrint Archive, Report 2018/642, 2018
15. Lightning RFC: Lightning Network Specifications, https://github.com/lightningnetwork/lightning-rfc, 2019
16. Joshua Lind, Ittay Eyal, Florian Kelbert, Oded Naor, Peter Pietzuch and Emin Gün Sirer, Teechain: Scalable blockchain payments using trusted execution environments, arXiv preprint 1707.05454, 2017
17. Stefan Dziembowski, Lisa Eckey, Sebastian Faust and Daniel Malinowski, Perun: Virtual Payment Hubs over Cryptocurrencies, IEEE Symposium on Security and Privacy (SP), 2019
18. Pavel Prihodko, Slava Zhigulin, Mykola Sahno, Aleksei Ostrovskiy and Olaoluwa Osuntokun, Flare: An approach to routing in Lightning network, White Paper, 2016
19. Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg, Settling payments fast and private: Efficient decentralized routing for path-based transactions, arXiv preprint 1709.05748, 2017
20. Stefanie Roos, Martin Beck, and Thorsten Strufe, VOUTE-virtual overlays using tree embeddings, arXiv preprint 1601.06119, 2016
21. Subhra Mazumdar, Sushmita Ruj, Ram Govind Singh and Arindam Pal, HushRelay: A Privacy-Preserving, Efficient, and Scalable Routing Algorithm for Off-Chain Payments, arXiv preprint 2002.05071, 2020
22. Stefan Dziembowski, Grzegorz Fabianski, Sebastian Faust and Siavash Riahi, Lower Bounds for Off-Chain Protocols: Exploring the Limits of Plasma., IACR Cryptol. ePrint Arch., 2020
23. Joseph Poon and Vitalik Buterin, Plasma: Scalable autonomous smart contracts, White paper, 2017
24. Saar Tochner, Aviv Zohar and Stefan Schmid, Route Hijacking and DoS in Off-Chain Networks, ACM Conference on Advances in Financial Technologies (AFT), 2020
25. Saar Tochner and Stefan Schmid, On Search Friction of Route Discovery in Offchain Networks, IEEE International Conference on Blockchain (Blockchain), 2020
26. Somanath Tripathy and Susil Kumar Mohanty, MAPPCN: Multi-hop Anonymous and Privacy-Preserving Payment Channel Network, International Conference on Financial Cryptography and Data Security, 2020,
27. Zhiping Cai, Fang Liu, Nong Xiao, Qiang Liu and Zhiying Wang, Virtual network embedding for evolving networks, IEEE Global Telecommunications Conference (GLOBECOM), 2010
28. Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo and Jie Wang, , Virtual network embedding through topology-aware node ranking, ACM SIGCOMM Computer Communication Review, 2011
29. Chen Avin and Kaushik Mondal and Stefan Schmid, Demand-aware network designs of bounded degree, Distributed Computing, Springer, 2019
30. Chen Avin, Alexandr Hercules, Andreas Loukas, and Stefan Schmid, rDAN: Toward robust demand-aware network designs, Information Processing Letters, Elsevier, 2018
31. Shahar Somin, Goren Gordon and Yaniv Altshuler, Social signals in the Ethereum trading network, arXiv preprint 1805.12097, 2018
32. Elias Rohrer, Julian Malliaris and Florian Tschorsch, Discharged Payment Channels: Quantifying the Lightning Network's Resilience to Topology-Based Attacks, IEEE European Symposium on Security and Privacy (SP) Workshops, 2019,
33. Ting Chen and Yuxiao Zhu and Zihao Li and Jiachi Chen and Xiaoqi Li and Xiapu Luo and Xiaodong Lin and Xiaosong Zhang, Understanding Ethereum via Graph Analysis, IEEE INFOCOM, 2018
34. Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura Netto, Peter Hahn and Tania Maia Querido, A survey for the quadratic assignment problem, Eur. J. Oper. Res., 2007
35. Sartaj Sahni and Teofilo F. Gonzalez, P-Complete Approximation Problems, J. ACM, 1976