2019 IEEE International Conference on Blockchain (Blockchain)

A Heuristic-based Private Bitcoin Payment Network Formation Using Off-Chain Links

Enes Erdin*, Mumin Cebe*, Kemal Akkaya*, Eyuphan Bulut† and Selcuk Uluagac*
*Florida International University, Miami, Florida
Email: {eerdi001, mcebe, kakkaya, suluagac}@fiu.edu
†Virginia Commonwealth University, Richmond, Virginia
Email: ebulut@vcu.edu

***Abstract*—While Bitcoin dominates the market for cryptocurrencies, its use in micropayments is still a challenge due to its long transaction validation times and high fees. Recently, the concept of off-chain payments is introduced that led to the idea of establishing a payment network called Lightning Network (LN). Off-chain links provide the ability to do transactions without writing to Blockchain. However, LN's design still favors fees and is creating hub nodes that defeat the purpose of Blockchain. In addition, it is still not reliable as not all the transactions are guaranteed to be transmitted to their destinations. If current retailers would like to use it, these problems might hinder its adoption. To address this issue, in this paper, we advocate creating a private payment network among a given set of retailers that will serve their business needs, just like the idea of private Blockchains. The goal is to build a pure peer-to-peer topology that will eliminate the need for relays and increase the robustness of payments. Using off-chain links as edges and retailers as nodes, the problem is formulated as a multi-flow commodity problem where transactions represent the commodities from various sources to destinations. As the multi-flow commodity problem is NP-Complete, we propose a heuristic approach that utilizes Dijkstra's shortest path algorithm in a dynamic way by updating the edge weights when new payment paths are to be found. The order of transactions is randomized to provide fairness among the retailers. The evaluations indicate that the proposed heuristic comes close to an optimal solution while providing scalability and user privacy.**

I. Introduction

Some of its enthusiasts accept Bitcoin as the next big innovation since the introduction of the Internet. Bitcoin has not only revolutionized the way payment systems can be designed in a purely distributed manner but it has also offered the novel Blockchain data structure that can be adapted in many other applications, from data storage to bookkeeping. The blockchain is now touted as an innovative solution in many areas such as healthcare, finance, government operations, logistics, etc. [1], [2], [3].

Without a shadow of a doubt, Bitcoin has opened many new opportunities. However, it has been long criticized for its slow transaction confirmation times and high fees charged [4], [5]. The Bitcoin network, by design, tries to adjust the confirmation time of a block to 10 minutes. In general, a block is assumed to be valid after the confirmation of the 6th subsequent block, which yields the confirmation time of a transaction to be around 60 minutes. Therefore, such long transaction confirmation times are not suitable for applications where timely payment evidence is critical. In addition, the transaction fees are not proportional to the amounts being transferred. These make Bitcoin impractical for many day-to-day micro-payment schemes such as buying a cup of coffee or paying for lunch.

Despite the mentioned impracticalities, Bitcoin is still the most widely used digital currency, and its market cap is above 50% among all digital currencies. So, it makes perfect sense to exploit this market cap and try to alleviate the above problems of Bitcoin. To this end, as a solution, the concept of *off-chain* payment channels [6] is introduced where transactions are done through escrow accounts. In this way, during a term of an agreement, two parties can perform many instant transactions in real-time without a need to write them back to the Blockchain. Thus, one can save the transaction fees that are conducted within the agreed term just because off-chain mechanism requires typically two transactions; one for opening the escrow account and one for closing.

Due to such advantages of off-chain, *payment networks* started to evolve by applying the off-chain concept widely such that a network of retailers and off-chain links can be created just like an Internet backbone to link every retailer and customer and allow multi-channel/multi-hop payments. Lightning Network (LN) is the payment network proposed in 2016 and deployed for Bitcoin in late 2017 which, as of today, serves for more than 4,000 nodes.

However, there are several issues with the current LN. First of all, instead of connecting retailers and customers directly, LN relies on *relay nodes* which act as bridges between retailers and customers. For the retailers this is a major shortcoming since this leads to a hub-and-spoke topology where there are some nodes which hold the most of the connections and capacity of the network. Consequently, this defeats the very idea of decentralization. An experiment where a practitioner was questioning the capacity of the channels in LN revealed interesting results [7]. During the time of that experiment, the average channel capacity was around $20 and the success rate for sending $5 and $0.43 was around 50% and 90% respectively. These numbers indicate that adoption of LN by current retailers will not be possible. Second, allowing the relay nodes to become monopolies in forwarding poses vulnerabilities for denial of service (DoS) attacks [8] and privacy analysis of customers' transactions.

Hence, we advocate the need for creating a *private payment network* that will bring together retailers under a consortium

978-1-7281-4693-5/19/$31.00 ©2019 IEEE
DOI 10.1109/Blockchain.2019.00046

to contribute to this network. This suggests that there will be a need for developing a highly decentralized topology which will be reliable and can support the needed amount of transactions.

In this paper, we propose to build such a payment network from scratch that will utilize off-chain payment channels with the objectives of distributing the forwarding loads evenly among all the nodes while minimizing the number of their off-chain channels to decrease the total fee cost of the network. By inspiring from the multi-commodity flow problem [9], we start with an optimization model that will optimally distribute the flow within an initial network topology. However, since the multi-commodity problem is NP-complete, the solution will not scale if all potential channel establishments are done on the initial network.

We thus come up with a heuristic idea which will form a network topology by relying on the transaction intents between nodes using a shortest path algorithm. As nodes start to transfer money to each other, weights on the edges will be updated so that the shortest path formations can be influenced in such a way that existing channels are favored to a certain extent. When all of the transactions are completed, we obtain a final topology by creating off-chain links on the used paths. We consider several criteria while initializing and changing the weights of the edges that will enable a highly decentralized topology. The evaluations using *Python* and *Gurobi solver* indicate that our proposed heuristics can provide comparable performance to that of the optimal solution while allowing scalability and fairness.

This paper is organized as follows: Next section summarizes the related work and in Section 3 we provide the background for the related concepts and the motivation for the problem. Section 4 explains the proposed algorithm and Section 5 presents the experimental setup and corresponding results. Paper is concluded in Section 6.

II. Related Work

A. Payment Networks

High transaction fees and long confirmation times are the major issues for the cryptocurrencies and there is a substantial interest in these issues from both the industry and academic community. Most of these efforts are concentrated around Bitcoin. Building Payment Channel Networks (PCN) is part of these efforts. PCNs can be classified into two categories. The first category relies on building a PCN for intra-blockchain operations. It allows transferring money between parties over already existing off-chain links without any confirmation delay but with some forwarding fees. LN and Raiden are examples that fall into this category [6], [10]. The second category of works relies on building inter-blockchain operations to allow transfers between different cryptocurrencies without expensive on-chain confirmation. Examples include Inter-Ledger [11] and Atomic-CrossChain [12].

Among the current PCNs, LN is the most widely adopted solution since the introduction of the off-chain payment channel by the Bitcoin community [13]. However, the LN framework is in its early phases and has many problems including reliability, scalability, privacy, and routing. While some of these problems such as privacy and efficient routing are being targeted by the Blockchain community [14], [15], [16], [17], these solutions all revolve around the existing LN structure and topology. In [18] the authors make a topological analysis of a snapshot of the LN taken in March 2018. They claim that LN is formed around a very small number of central nodes where periphery nodes are loosely connected to the center. The author of [19] statistically looks at the development of the LN in the course of 12 months since its establishment. With the findings, he suggests the capacity development of LN is not strongly correlated with the development of the size of the network where capacity grows more slowly. Our work in this paper has a different goal assuming that private PCNs can be created and offers efficient solutions from scratch to address the aforementioned issues.

B. Multi-commodity Flow Problem

The flow portion of our problem can be formulated similar to the multi-commodity flow problem which deals with the assignment of commodity flows from sources to destinations in a given network. However, multi-commodity flow problem has been shown to be NP-Complete [20] even if the number of commodities is two. When the problem becomes fractional and can be modeled with linear programming, it can be solved in polynomial time [21]. Nonetheless, in a multi-commodity flow problem, the flows are optimized on a given network topology. Our problem is different from the multi-commodity flow problem as we do not have the topology in hand and try to jointly optimize the topology and the total costs by respecting the flow constraints.

The same problem in the context of electric vehicle (EV) charging coordination has been studied and solved with an optimization model in [22]. However, as the number of charging stations, channels and EVs increase, the time for solving the problem increases dramatically. The solution in that work does not scale beyond 10 nodes. Our work in this paper aims to offer a scalable solution to the same problem through a heuristic approach.

III. Background and Motivation

A. Background on Off-chain Links

Off-chain transaction channels mechanism is used for saving transaction fees and time in the current Bitcoin system which constitutes the main motivation of this study. Specifically, an in-advance payment is provided to the Blockchain via establishing a 2-of-2 multi-signature escrow account, but future successive transactions are tracked and signed by the peers without being written to Bitcoin's public ledger. The amount put in the escrow account is decided individually by both parties and unless that amount is reached, the transactions can continue. In this scheme, the peers only pay fees for two on-chain transactions: one to open the channel and one to close it.

The example shown in Fig. 1 depicts this concept. Alice opens an off-chain channel with Bob. They both sign the

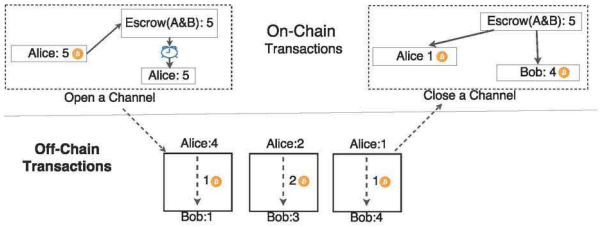


Fig. 1: Off-chain mechanism between two Blockchain nodes

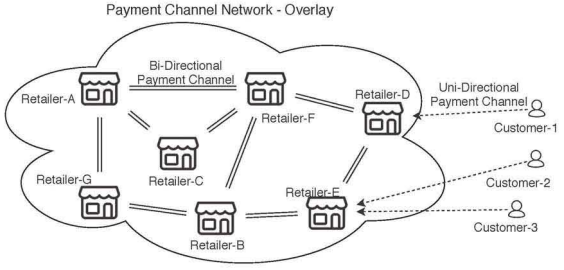


Fig. 2: An overview of the envisioned Payment Network among retailers.

new account separately. Alice then deposits 5 Bitcoins to the escrow account by performing an on-chain transaction which determines a directional channel capacity, from Alice to Bob, as 5 Bitcoins. From now on, Alice can make payments to Bob simply by giving the ownership of some of her Bitcoins to Bob until the capacity of the channel is reached. In the figure, we see only 3 transactions at different times: 1, 2, and 1 Bitcoins. Eventually, when the channel is closed, only the remaining Bitcoins and the total transferred Bitcoins are committed respectively to Alice and Bob and written to the public ledger. The payment channel provides guarantees to Alice and Bob to refund the balance in the escrow account at any time or at a mutually agreed channel expiration time. This guarantee is satisfied by a smart contract called "*Hash Time Locked Contracts (HTLC)*" [23].

LN exploits the off-chain concept to create multi-hop payment paths between participants. To enable this idea in practice, users are supposed to route their payments to any destination through a series of payment channels in a network of nodes. If such a channel/link series exist among the nodes, then a user can utilize one or more of these links (i.e., multi-hop links) to reach another node for making a payment. A sample payment network is shown in Fig. 2.

B. Problem Motivation and Definition

Even though the concept of LN is very attractive, its current structure requires the deployment of relay nodes between payers and payees. These relay nodes will eventually become major hubs in the network creating the risk of going through DDoS attacks to stop the payments in the network at any time. Another risk here is regarding customer privacy. If these big relay nodes are compromised, they can easily analyze the payments passing through them which will expose the privacy of the customers using them as relays.

In this paper, we argue that retailers who would like to attract more business from cryptocurrency users will be willing to come together to form a **private** PCN that can be controlled by them so that it can better satisfy customers' needs. Therefore, this new PCN should be separate from the existing LN and address its shortcomings as listed below:

- **Network connectivity:** In LN there is a basic assumption that a payment network can be formed by ad-hoc connections and without a specific topology plan. This ad-hoc assumption is not effective since there will be a certain probability of connectivity success which means that the final payment network may not be connected. The proposed topology needs to guarantee network connectivity.
- **Network scalability:** As more nodes join LN, the relay nodes become more congested and eventually the network topology is dominated by the relay nodes that may create an oligarchy. The proposed topology will create a P2P topology among the retailers and prevent any of the retailers to become a hub node.
- **Investment for each channel:** Forming a connected network will not be free. A valid channel means two mandatory on-chain transactions. Hence, the number of channels established by a node should be kept in an optimum level, namely, high enough to keep the transaction requests in the network to flow through but low enough to decrease the total on-chain transaction fees.
- **Partial usage of available payment capacity:** A node may assume that it needs 100 Bitcoins worth of total transaction volume for its own business. However, that capacity will be used by other nodes which use this node as a relay. Thus, at a given time, only a portion of the capacity will be available for the node itself to accept transactions from its own customers. This implies that one should invest much more than its anticipated transaction volume.
- **Diminishing channel capacity over time:** The capacity of channels in LN diminishes over time and thus some transactions which are set to use those channels may get stuck. Therefore, there may not be any payment guarantee as shown in [7]. For resolving this issue, either more investment should be planned in the channel in advance or there should be a reverse payment to balance the forward capacity. The proposed topology needs to guarantee that any payment will reach its destination at any time.

Based on these discussions, our problem can be formally defined as follows: *Let us assume N nodes (retailers). Let us also assume that a PCN among these retailers can be represented as a graph $G = (V, E)$, where V represents nodes (of N retailers) and E represents all payment channels among N retailers. Every edge between retailers has a capacity that denotes the amount of depositable Bitcoins. We assume that every vertex (retailer) $v \in V$ will make an initial total investment that represents the maximum Bitcoins that can be*

transmitted or forwarded over it. In other words, we are considering the maximum possible instantaneous payments that can be made from a retailer or forwarded by it. This can also be described as the maximum possible business capacity of a retailer within a certain time. Note that we assume that for each retailer there are $N-1$ registered customers making a unique transaction to another retailer. So, for example from $Node_1$, there exists $N-1$ transactions to other $N-1$ retailers.

Based on these inputs, how can we create a scalable virtual topology PCN among the retailers in such a way that 1) the total investment made by a retailer for creating channels with its neighbors will be minimized; 2) the topology will be close to an ideal P2P topology with no hub nodes but still satisfy all payment requests; and 3) the standard deviation of total investment costs among the retailers will be minimized to ensure fairness.

IV. Algorithm Description

A. Approach Overview

Our heuristic of payment network formation is based on the idea of in-advance planning of payments and flows. As every retailer has an idea of their business capacity and expectation, we use it to plan payment flows among the customers and retailers. We start distributing the flows in advance from various retailers to others in the best way we can (i.e., fair load and P2P distribution) assuming that there is already available channels among them at the beginning. We then look at the final used channels among retailers, set up the actual off-chain links and ignore any other channels.

In this heuristics, finding the path between a source and a destination retailer is crucial. When we observe today's LN, if there is a path between the payer and the payee, the payer can use that path if it is convenient to use meaning if there is enough capacity on the planned path. Otherwise, the other alternative is to establish a direct channel with the payee. However, in that case, there will be on-chain transaction fees for opening and closing channels. Therefore, one needs to weigh these two options when finding a path. We follow a similar rationale for our case. Specifically, if there is a path from one retailer to another one, use that path. If there is none, open a new channel. Additionally, if total cost on the path will start to create inconvenience for intermediate retailers (i.e., adding a burden of forwarding), then open a new channel. In a sense, we strive to find an approach for opening channels so that the participants of the network neither suffer from unfair load distribution nor pay excessive on-chain transaction fees.

B. Finding Paths

In order to find the best possible routes, Dijkstra's shortest path algorithm is used [24]. In Dijkstra's algorithm, the path with the lowest total weight is found between a source and a destination node. In our case, we have an apriori payment list. From the payment list, transactions are read one by one. At each reading, meaning iteration, a shortest (lowest weight) path from the source to the destination is found. After a path is found, the weights on the edges are updated according to the flow (i.e., payment amount) which will be detailed in the next subsection. In the light of the explanations done in Table I, the algorithm is shown in Algorithm 1.

TABLE I: Notations and their explanations

Symbol	Meaning
H	Directed Graph
H_e	Edge e in graph H
L_c	Link establishment cost
W_i	New connection forcing cost
γ	Parameter to control unfairness between the nodes
T_a	Transaction amount
T_{AB}	Transaction amount on edge (A, B)
$H_e.weight$	Weight of edge e
$H_e.flow$	Flow on edge e
U_{AB}	Binary var. represents existence of flow on edge (A, B)
E	Initial number of edges in the experiments

Algorithm 1 Network Establishment

Input: *P=Payment List*, *H=fully connected directed graph*, L_c=Link establishment cost, W_i=New connection forcing cost
for every edge, e, in H **do**
    $H_e.weight$=W_i+L_c
    $H_e.flow$=0
end for// *Initial assignments are done*
for every *payment* in P **do** // *A payment is defined by a source, a destination and the transfer amount* T_a
    $Path$=$ShortestPath$(H, from=a, to=b)
    for *Each edge*, e, in $Path$ **do**
        $H_e.flow$ +=T_a
        $H_e.weight$=W_i+$H_e.flow$
    end for
end for
for All edges in H **do**
    if $H_e.flow$=0 **then**
        Remove edge from H
    end if
end for
Output: H

Note that here the payments are picked in a round-robin fashion (i.e., finish a particular retailer's payments and move on to the next) which may greatly influence the resultant topology as we followed a certain order. This may create unfair load distributions and undesirable topologies.

In order to come up with a topology in which the loads are more evenly distributed, the randomly selected customers execute their transactions in a random round robin fashion. Namely, in order to minimize the impact of dependence on the order, at each round, the order of the retailers are renewed with a new distribution. This approach is shown in Algorithm 2. According to this approach, first, 2 nodes are selected randomly, one of which is the source, a, and the other is the destination, b. If there is an intended transaction from a to b, and if it is not fulfilled yet, a transaction from a to b with the transaction amount is added to a payment list, P. Afterward,

Algorithm 2 List Establishment

```
Input: S=Set of Retailers
while All required payments are not fulfilled do
    TempS=S
    while TempS is not Empty do
        Pick 2 random retailers (a,b) from TempS
        if Transaction from a to b was not fulfilled then
            Add a as source, b as target in P
            Remove (a,b) from TempS
        end if
    end while
end while
Output: P=Payment List
```

a, b pair is removed from the list of retailers. This removal is important because we want every node to be visited equally either as a source or as a destination. Whenever every retailer is visited equally, meaning list of retailers is an empty set, the procedure is repeated. This new random list of payments is then fed to Algorithm 1.

C. Defining Edge Weights

As mentioned, after finding a path the edge weights need to be updated to inject our influence to topology formation. To achieve this, we define a sophisticated **weight function**, on an edge, e.g. the weight between A and B, W_{AB}. Specifically, three components of the W_{AB} are defined: *link establishment cost*, *transaction cost*, and *new connection forcing cost*.

Link Establishment Cost ***(L_c):*** In LN, establishing a channel means doing at least two on-chain transactions on Bitcoin blockchain, which incur on-chain transaction fees. For a fully connected mesh network of N nodes, there will be $N \times (N-1)/2$ edges. With increasing N, the total fee paid by the network participants will be tremendously high. Instead of full connection in the network, a lower number of edges will be more acceptable as it lowers the total on-chain transaction fee. The edges should be reused cleverly to distribute the transactions among nodes in an acceptable way.

In order to encourage the reuse of the edges, a parameter called $LinkCost$, denoted as L_c is introduced. L_c mainly relates to the on-chain transaction fee. In the proposed heuristics, all edges in the network have a non-negative L_c set to some value. Whenever an edge is used, meaning there is a flow on the edge, the L_c on that edge is nullified (set to 0 to indicate that the channel is already open). So further transactions can use that edge on their paths. Nullifying the L_c encourages other transactions in such a way that, other transactions will prefer low-cost edges instead of opening a new one.

Transaction Cost: When an edge is used (or channel is established), L_c will be nullified, and thus all later transactions will tend to use that channel since other yet-never-used channels will have a higher weight coming from L_c. This high usage of the channel contradicts with the aim of establishing a flat network to execute all of the transactions. For that reason, whenever there is a transaction through an edge, the amount transferred incurs a weight on that edge which is basically a cost induced by channel usage. So, when there are edges with heavier loads, the transactions will start to look for new routes or open new channels. This helps to distribute the loads more evenly. Hence the weight, W_{AB}, can be revised as:

$$W_{AB} = L_c(1 - U_{AB}) + \sum T_{AB} \tag{1}$$

where U_{AB} is a binary variable and equates to 1 if there happens a flow on edge AB anytime during the procedure, 0 otherwise, and T_{AB} is the amount of all transactions (from all nodes in the network) passing on edge (A, B).

New Connection Forcing Cost: In some cases, when the links are established, during the algorithm run, the future transactions in the list tend to use those links which will increase the investments need to be made by intermediate nodes for maintaining these links. In such cases, we need an additional force to further increase these links' weights so that the Dijkstra's algorithm will not choose these links anymore.

As an example consider an initially all connected network topology shown in Fig. 3(a) where all of the edge weights are initialized accordingly, with $L_c = 500$. If we look at the established links after the first run of payments, we see that half of the nodes initiates transactions to the remaining half of the nodes randomly in one hop as shown in Fig. 3(b), and the effect of L_c is nullified and the weights are updated with the flows on the edges. Note that, for simplicity of the figure not all of the $L_c = 500$ weights are shown in the figures. In the second round of transactions, new randomly selected half of the nodes will initiate new transactions to other nodes. This will form new connections as shown in Fig. 3(c) but still the opened links will be in use. However, now unused edges in the topology will still have a higher weight of L_c (500 in this example), while other edges will have the weights only created by the transaction amounts. In the later rounds, no matter how random the nodes are picked, all of the transactions will follow the already established edges, since they will have lower weights due to their L_c being set to 0. Thus, the topology will continue to be as shown in Fig. 3(c), resulting in no significant change in the topology. For a larger N, we will observe long paths in the network and the topology will stay unchanged although more transactions are added. In order to prevent transactions traversing always through the same paths, we introduce a constant weight, W_i, for each edge to increase the total weight on the path and thus surpass the edges with L_c ((i.e., forcing new connections). In this way, the long paths will be hampered. Thus, the weight on edge (A, B) will be updated as follows:

$$W_{AB} = L_c * (1 - U_{AB}) + \sum T_{AB} + W_i \tag{2}$$

The effect of W_i can be better seen with an example shown in Fig. 4. In this sample experiment L_c is set to 3000, and the number of nodes, N, is 20. The payment lists are the same for both of the resulting topologies. Basically, without W_i, we will get a topology in Fig. 4a. Introducing W_i and setting it to 1000 causes the topology to change to the one in Fig. 4b. We

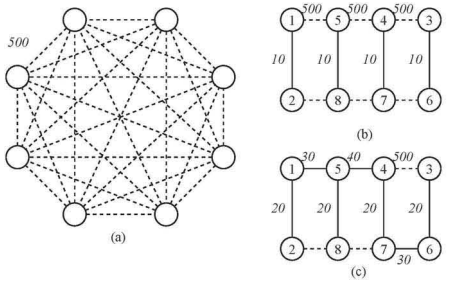
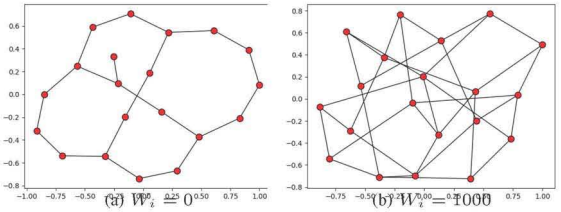


Fig. 3: Initially all connected network topology.

note that Fig. 4a is not a desired topology because it is prone to node based failures, and some nodes are highly centralized.



(a) $W_i = 0$ (b) $W_i = 1000$

Fig. 4: Effect of W_i for a network of 20 nodes, $L_c = 3000$

V. Evaluation

A. Experimental Setup and Implementation

N nodes (retailers) are assumed in the network. A single customer is assumed to be attached to a single node and it will create 10 unit worth transactions to every other node. So, the supply from a single customer to the network is $(N-1)\times 10$. Total money traversing in the network is $N\times(N-1)\times 10$. In LN channel formation, the peers can independently decide on the amount they want to put in the channel. However, for the completeness of the study, we assume thatpeers of a channel put the same amount in the channel. The proposed approach is implemented in Python and its performance is assessed in extensive experiments. All the experiments are carried out on a computer with an Intel Xeon E5-2630 v4 @ 2.20GHz CPU and 64 GB of RAM.

B. Metrics and Benchmarks

The results of the experiments are evaluated based on the following metrics:

- ***Betweenness Centrality of nodes:*** Betweenness centrality of a node in a network is a measurement showing how many times a node is visited while traveling between other nodes using the shortest path traversal. In a hub-and-spoke network model, hubs will have the highest betweenness score.
- ***Total Capacity of the Network:*** This metric shows the total amount of investment to be put by the vendors to the channels for the formation of the network.
- ***Number of Edges:*** This metric shows the number of edges established in the resultant topology.
- ***Standard deviation among the nodes:*** This metric shows the standard deviation among the outbound flows of the nodes. A high standard deviation hints that some of the nodes are used more like a relay compared to the other nodes. A zero standard deviation means all of the loads on the nodes are equal.
- ***Total Computation time:*** This metric is the measure to show how long it takes, in seconds, to finish all necessary computations for the final results.
- ***Utilization:*** This metric is the ratio of the total flow in the network to the total capacity of the network. It is calculated by dividing the sum of all transactions to sum of all established capacity in the network.
- ***Histogram of Number of Hops:*** This metric shows the histogram of the transactions in terms of the number of hops they make calculated in percentage.
- ***Cut Nodes:*** Cut nodes are the nodes whose removal entirely makes the network disconnected. The higher the better for a topology since this means more nodes need to be removed/failed to disconnect the network.

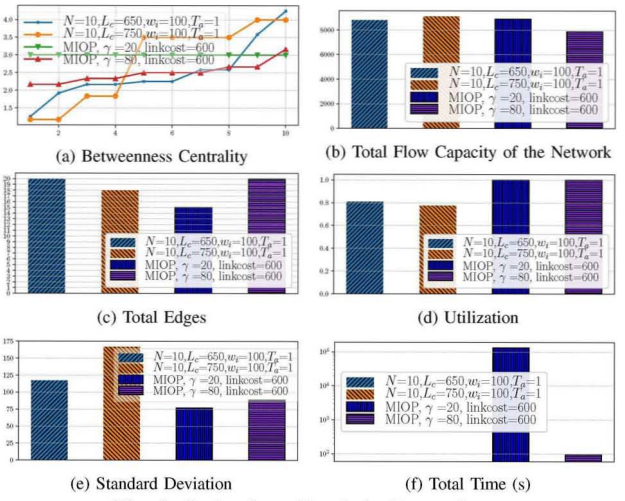
We compared our approach against certain benchmarks and methods as listed below:

- **MIOP model**: The results of the heuristic are compared with an optimization model in [22].
- **Random network topology**: The results of the heuristic are also compared with the results of a randomly connected network. The heuristic is run on the random network to get the results about the flow in the network.

C. Experiment Results and Discussion

1) Comparison of Heuristic with the MIOP model: In this section, the results of the heuristic approach are presented and compared with that of the MIOP model studied in [22]. The objective is to assess our approach's performance with respect to the ideal case. The optimization model was solved by Gurobi Solver. However, in the setup of this experiment, only 10 nodes are used since the MIOP model does not scale beyond 10 and thus in practice is not usable. Only for this experiment, different than the general scenario assumption, we assumed that these 10 nodes are serving to 80 customers which are distributed to these nodes randomly. Each customer sends money to 6 different nodes and each is of a value of 10 units. Hence total supply by the customers to the network is 4800 units. From the experiment results of the MIOP model, best ones are used in regards to having flat betweenness centrality, lower standard deviation and lower number of edges. For the results of the heuristic approach, the same scenario is inherited. All the related results are shown in Fig. 5. In those figures, γ is a control parameter for the unfairness among node outbound flows, and $linkcost$ is the link establishment cost in MIOP.

As can be seen from Fig. 5(b) and (c), our heuristic performs almost matches the performance of the MIOP solution in terms of total capacity and edges. It is also only 20% short of the utilization of MIOP (Fig. 5(d)). For the standard deviation metric, as MIOP has a significant control on unfairness, the standard deviation in MIOP solutions is lower than that of the heuristic approach as seen in Fig. 5(e). However, when W_i is 100 and L_c is 650 in the heuristic approach, standard deviation

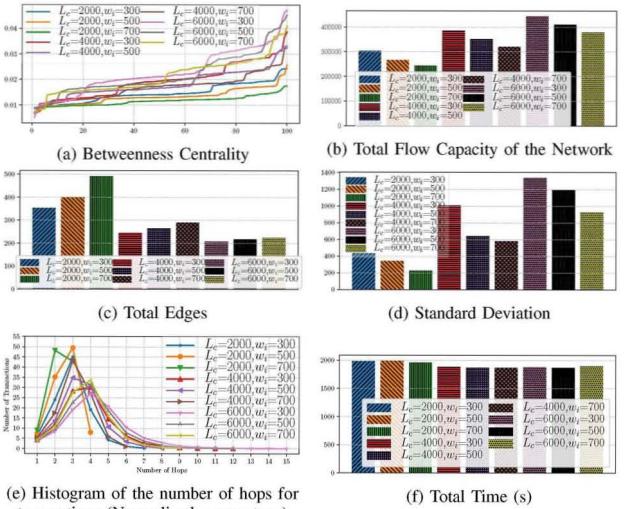


(a) Betweenness Centrality (b) Total Flow Capacity of the Network

(c) Total Edges (d) Utilization

(e) Standard Deviation (f) Total Time (s)

Fig. 5: Optimal vs. Heuristic Comparisons



(a) Betweenness Centrality (b) Total Flow Capacity of the Network

(c) Total Edges (d) Standard Deviation

(e) Histogram of the number of hops for transactions (Normalized- percentage) (f) Total Time (s)

Fig. 6: Ideal Parameter Selection

comes to a more comparable level, where the number of edges has a significant effect on this. This is because as the number of edges increases, the flows tend to be distributed more evenly since the flows can find shorter routes compared to a network with a lower number of edges. Finally, compared to MIOP solutions the betweenness centrality for our approach in Fig. 5(a) is slightly increasing but still maintains a topology close to P2P.

The obvious advantage of our approach is computational overhead. It reduces the computational time 100 to thousands folds (i.e., it scales much better) while still getting very close to the MIOP's overall performance (Fig. 5(f)). In summary, the proposed approach provides the same features as MIOP in a much faster/scalable manner but with some deviation from an ideal P2P topology.

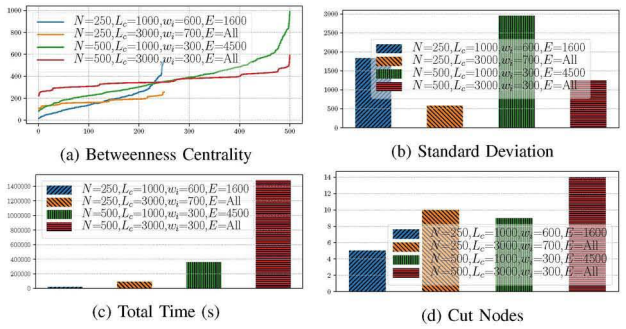
2) Ideal Parameter Selection for the Heuristic: Apparently, picking different parameters highly affects the resulting network topology for the heuristic approach. In this section, we conducted a series of experiments to determine the ideal parameters for our heuristic to run. The experiments are evaluated for different L_c and W_i cases and a fixed number of nodes, $N = 100$, which yields traversal of 99000 units of money in the network, with an exact amount of 990 units per node. The results are shown in Fig. 6.

Considering all of the different parameters visited in the course of this experiment, with the payment scenario assumption and under 100 nodes, we obtain a good topology when L_c is 4000 and W_i is 700. We call the topology good because, the standard deviation is around 600, with an average load per node is around 3000. The total number of edges in the network is close to 300 implying on average every node has 6 connections. Additionally, the maximum number of hops does not exceed 6 and resides around 3.

3) Scalability of the Heuristic: In this experiment, we assessed the scalability features of the proposed heuristic. Specifically, the heuristic approach is run with different number of nodes, namely 250 and 500 nodes and the behavior of the algorithm is observed. The betweenness centrality results is shown in Fig. 7(a). The flow capacity of the network is shown in Fig. 7(b). The number of edges in the network is shown in Fig. 7(c). The number of hops done for a transaction in percentage is shown in Fig. 7(d). The standard deviation between the capacity of the nodes is shown in Fig. 7(e). The time elapsed for the procedures to finish is shown in Fig. 7(f).

As the number of nodes increases, the computation time required to finish the calculations increase drastically due to the time complexity of Dijkstra algorithm which is, if implemented in simple form, $O(|E|log|N|)$, where E is the number of edges and N is the number of nodes. Since our heuristic starts with the assumption that all nodes are connected to each other, the number of edges becomes $E = N^2$. So the time complexity of the heuristic translates into $O(N^2 logN)$. In order to decrease the effect of the assumption of an initially-fully-connected network, we created networks with random initial connections. The initial number of edges are depicted with the E parameter in the figures ($E = All$ indicates our approach with a fully connected network). The heuristic is applied to the randomly connected networks for network flow calculation. Although it is very hard to get an exactly equal number of edges for both networks, we tried to keep them close and we believe the results are comparable. Note that, for $N = 500$ total amount circulating in the network is 2,495,000 units. Load per node is 4990 units. Pre-pruned edges give an advantage in terms of total computation time, as expected. However, other results are better for the initially-fully-connected network setup. Additionally, based on the results, we argue that making random connections may not degrade the total investment capacity in the network but comes with unfairness among the nodes as standard deviation among nodes changes too much.

As part of this experiment, we also observed the number of cut nodes. Figures 7(d) represents the results of the cut nodes for different parameters. As can be seen, when the network is randomly connected, we observe a lower number of cut nodes compared to our heuristic topology. That means, for a



(a) Betweenness Centrality (b) Standard Deviation

(c) Total Time (s) (d) Cut Nodes

Fig. 7: Scalability Test Results

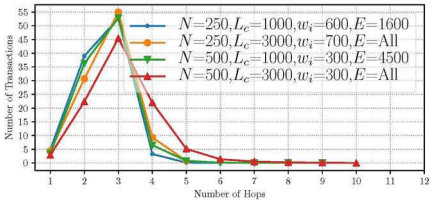


Fig. 8: Histogram from the scalability tests

randomly connected network, the possibility for taking down the network is easier because attacking fewer nodes will be enough. This is not the case in our heuristic as its betweenness centrality is more stable and thus more nodes need to be taken down in order to disconnect the network. As the network size doubles, this number also increases linearly indicating that our heuristic maintains a similar behavior as new nodes are added.

4) Privacy Experiments: In this experiment, we looked at the number of hops traveled for each transaction. The higher the number of hops, the better the privacy of the participants. Based on the results shown Fig. 8, we can observe that only around 5% of the transactions are carried out in a single hop. More than 50% of the transactions are realized in 3 or more hops which provides pretty good privacy. Note that having at least 3 hops in routing is a practice followed in Tor project [25]. Increasing the number of nodes or using random topologies do not have much impact on the results. In fact, with 250 nodes, the number of transactions having 3 hops is even slightly higher than that of 500 nodes. However, with increased node count, the percentage of transactions with 4 or more hops would increase as seen in Fig. 8.

VI. Conclusion

Cryptocurrency based payment networks using the idea of off-chain are recently emerging. This is not only because they reduce confirmation times but also let users send micro-payments in a very affordable way. Therefore, forming a reliable and scalable P2P payment network is an open question. In this study, based on some scenario and assumptions, we developed a heuristic approach to form such a payment network topology using Bitcoin's off-chain concept and compared the results with the results of an optimal solution. Compared to the optimal solution, the heuristic reduces the computational time significantly. Additionally, the fair distribution of the load among nodes, centrality measures and the total number of edges obtained in the networks are satisfying to ensure a truly P2P network topology features.

References

[1] T.-T. Kuo *et al.*, "Blockchain distributed ledger technologies for biomedical and health care applications," *Journal of the American Medical Informatics Association*, vol. 24, no. 6, pp. 1211–1220, 2017.

[2] N. Hackius and M. Petersen, "Blockchain in logistics and supply chain: trick or treat?" in *Proceedings of the Hamburg International Conference of Logistics (HICL)*. epubli, 2017, pp. 3–18.

[3] M. Cebe *et al.*, "Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles," *IEEE Communications Magazine*, October 2018.

[4] Bloomberg, www.bloomberg.com/view/articles/2017-11-14/bitcoin-s-high-transaction-fees-show-its-limits, 2017.

[5] BitInfoCharts, bitinfocharts.com/comparison/bitcoin-transactionfees, 2017.

[6] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," *Technical Report (draft)*, 2015.

[7] diar.co, "Lightning Strikes, But Select Hubs Dominate Network Funds," https://diar.co/volume-2-issue-25, June 2018.

[8] TrustNodes, "Lightning network ddos sends 20% of nodes down," *trustnodes.com/2018/03/21/lightning-network-ddos-sends-20-nodes*, 2018.

[9] A. Haghani and S.-C. Oh, "Formulation and solution of a multi-commodity, multi-modal network flow model for disaster relief operations," *Transportation Research Part A: Policy and Practice*, vol. 30, no. 3, pp. 231–250, 1996.

[10] Raiden, raiden.network/, 2018.

[11] S. Thomas and E. Schwartz, "A protocol for interledger payments," interledger. org/interledger. pdf, 2015.

[12] L. N. Team, "Atomic cross-chain trading." [Online]. Available: en.bitcoin.it/wiki/Atomic cross-chain trading

[13] Bitcoin wiki, "Bitcoin contract," en.bitcoin.it/wiki/Contract.

[14] S. Roos *et al.*, "Settling payments fast and private: Efficient decentralized routing for path-based transactions," *arXiv preprint arXiv:1709.05748*, 2017.

[15] G. Malavolta *et al.*, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 455–471.

[16] P. Prihodko *et al.*, "Flare: An approach to routing in lightning network," 2016.

[17] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, "Sprites: Payment channels that go faster than lightning," *CoRR abs/1702.05812*, 2017.

[18] I. A. Seres, L. Gulyás, D. A. Nagy, and P. Burcsi, "Topological analysis of bitcoin's lightning network," *arXiv preprint arXiv:1901.04972*, 2019.

[19] S. Martinazzi, "The evolution of lightning network's topology during its first year and the influence over its core values," *arXiv preprint arXiv:1902.07307*, 2019.

[20] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *16th Annual Symposium on Foundations of Computer Science*. IEEE, 1975, pp. 184–193.

[21] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," *ACM Transactions on Algorithms*, vol. 4, no. 1, p. 13, 2008.

[22] E. Erdin, M. Cebe, K. Akkaya, S. Solak, E. Bulut, and S. Uluagac, "Building a private bitcoin-based payment network among electric vehicles and charging stations," *IEEE International Conference on Blockchain*, 2018.

[23] "Hash Time Locked Contracts," en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.

[24] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[25] "The Tor Project," https://www.torproject.org/.