Journal of Network and Computer Applications 180 (2021) 103021

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A scalable private Bitcoin payment channel network with privacy guarantees

Enes Erdin[a,*], Mumin Cebe[b], Kemal Akkaya[c], Eyuphan Bulut[d], Selcuk Uluagac[c]

[a] *Computer Science Department, University of Central Arkansas, Conway, AR, USA*
[b] *Computer Science Department, Marquette University, Milwaukee, WI, USA*
[c] *Electrical and Computer Engineering Department, Florida International University, Miami, FL, USA*
[d] *Computer Science Department, Virginia Commonwealth University, Richmond, VA, USA*

ARTICLE INFO

Keywords:
Blockchain
Bitcoin
Payment channel networks
Shortest path
Lightning network

ABSTRACT

While Bitcoin heavily dominates the cryptocurrency markets, its use in micropayments is still a challenge due to long transaction confirmation times and high fees. Recently, the concept of off-chain transactions is introduced that led to the idea of establishing a payment channel network called Lightning Network (LN), which utilizes multi-hop payments. Off-chain links provide the ability to make instant payments without a need to writing to Blockchain. However, LN's design still favors fees, and it is creating hub nodes or relays that defeat the purpose of Blockchain. In addition, it is still not reliable, as not all transactions are guaranteed to be delivered to their destinations. These issues hinder its wide adoption by retailers. To address this issue, in this paper, we argue that the retailers could create a private payment channel network among them to serve their business needs, just like the concept of private Blockchains. The goal is to build a pure peer-to-peer topology that will prevent the formation of hub nodes while also eliminating the need for any relays to increase the robustness of the payments. Assuming off-chain links as edges and retailers as nodes, we formulate the problem 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 for payments in a dynamic way by updating the edge weights when new paths need to be found. The order of transactions is randomized to provide fairness among the retailers. We further extend this approach to guarantee the privacy of payments by forcing all the payments to travel at least three hops. We utilized *k*-shortest path algorithm to choose from *k* options that will meet our criterion. The evaluations indicate that the proposed heuristic comes close to an optimal solution while providing scalability and guaranteeing user privacy.

1. Introduction

Some welcome Bitcoin as the next big innovation since the introduction of the Internet (Zebpay, 2017). Undoubtedly, 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 is now endorsed as an innovative solution in many areas such as healthcare, finance, government operations, logistics, etc. (Kuo, 2017; Hackius and Petersen, 2017; Cebe, 2018).

The idea in Bitcoin is to process batches of transactions and once they are validated by *miners*, they are stored in a chain of blocks maintained as a distributed ledger. Therefore, once a transaction is written in a block in the Blockchain after a consensus, it cannot be deleted or changed. This persistent, transparent and append-only structure of the Blockchain uncovers a strong platform where the shareholders can store or transfer ownership of their assets in a trustless way.

For sure, Bitcoin has unfolded many new opportunities. However, it has been widely criticized for its long transaction confirmation times and high fees charged for the transactions (Bloomberg, 2017; BitInfoCharts, 2017). The Bitcoin network, by design, tries to adjust the confirmation time of a block to 10 min. In general, a block is accepted to be valid after the confirmation of the 6th subsequent block, which yields the confirmation time of a transaction to be around 60 min. 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

* Corresponding author.
E-mail address: eerdin@uca.edu (E. Erdin).

https://doi.org/10.1016/j.jnca.2021.103021
Received 13 July 2020; Accepted 6 February 2021
Available online 18 February 2021

challenges make Bitcoin impractical for many day-to-day micropayment 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 try to alleviate the above problems of Bitcoin. To this end, as a solution, the concept of *off-chain* payment channels (Poon and Dryja, 2015) was introduced where transactions are done through *escrow*-like accounts. In this way, in the duration of an agreement, two parties can perform many instant payments in real-time without a need to always write them to the Blockchain. Thus, one can save the on-chain transaction fees that are conducted within the agreed term just because the off-chain mechanism requires typically two on-chain transactions; one for opening the escrow account and one for closing it.

Due to such advantages of off-chain payments, *payment channel networks* (PCNs) 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. A PCN is essentially a network topology that allows routing of payments from any source to any other destination.

Lightning Network (LN) is a PCN proposed in 2016 and deployed for Bitcoin in late 2017 which serves for, as of today, more than 10,000 nodes. The introduction of LN also introduced another level of privacy to the cryptocurrency users. In LN, when a channel is established between two parties for off-chain transactions, it has a certain *capacity* and can be either *private* or *public*. In the case of a private channel, the peers do not need to advertise their intent to the network. For a public channel, while it is known to everyone, the directional capacities (i.e., one-way transaction capacity to the other party) of the channel are still not disclosed to the network. The capacity information advertised by the peers is the total capacity of the peers who own the channel. In this way, the total assets of the users are kept private to a certain extent. Additionally, when there is a transaction following a multi-hop path, the intermediary nodes do not know the source and the destination nodes of the payment. They only know the next hop.

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 some of the nodes hold the most of the connections and capacity of the network. Consequently, this defeats the very idea of decentralization. A recent experiment where a practitioner was questioning the capacity of the channels in LN revealed interesting results (diar.co, 2018). 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 if success rates do not improve significantly. Second, allowing the relay nodes to become monopolies in forwarding poses vulnerabilities for denial of service (DoS) attacks (TrustNodes, 2018) and privacy analysis of customers' transactions assuming that some of these nodes are compromised to monitor transactions passing through them.

Hence, we advocate formation of a *private PCN* that will bring together retailers under a consortium rather than opening it to public as in the case of LN. 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 with additional privacy constraints for the participants. In this paper, we propose to build such a private PCN 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. Inspired by the multi-commodity flow problem (Haghani and Oh, 1996), the problem can be modeled as such where commodities will be our transactions. However, since the multi-commodity flow problem is NP-complete (Even et al., 1975), an optimization model will not scale.

We thus came up with a heuristic idea which will form a network topology by relying on the transaction intents between nodes using the shortest path algorithm. As nodes start to transfer money to each other, weights (or interchangeably referred to as costs) 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. There are three components in the weight of an edge, namely, link-establishment cost, transaction cost, and the new channel forcing cost. 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.

Finally, we propose to extend this approach for guaranteeing the privacy of the payments inspired by the approach in Tor where each message travels at least 3-hops. Similarly, we aim to achieve at least 3-hops for each payment path to satisfy privacy for the payments (Dingledine et al., 2004). To force this, we utilized k-shortest path algorithm for the paths that have path length less than 3 and conduct a re-routing.

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. We also achieved 3-hops payments with similar topology features with a slight increase in the computational time.

This paper is organized as follows: The 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 explains the extension of the proposed algorithm with guaranteed privacy. Section 6 presents the experimental setup and corresponding results. Paper is concluded in Section 7.

2. Related work

2.1. Payment channel 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 PCNs is a 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 (Poon and Dryja, 2015; Raiden, 2018). 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 (Thomas and Schwartz, 2015) and Atomic-CrossChain (Team).

2.2. Lightning Network

Among the current PCNs, LN is the most widely adopted solution since the introduction of the off-chain payment channel by the Bitcoin community (Bitcoin wiki). 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 (Roos, 2017; Malavolta and others, 2017; Prihodko, 2016; Miller et al., 2017), all of these solutions revolve around the existing LN structure and topology. In (Seres et al., 2019) 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 (Martinazzi, 2019) 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

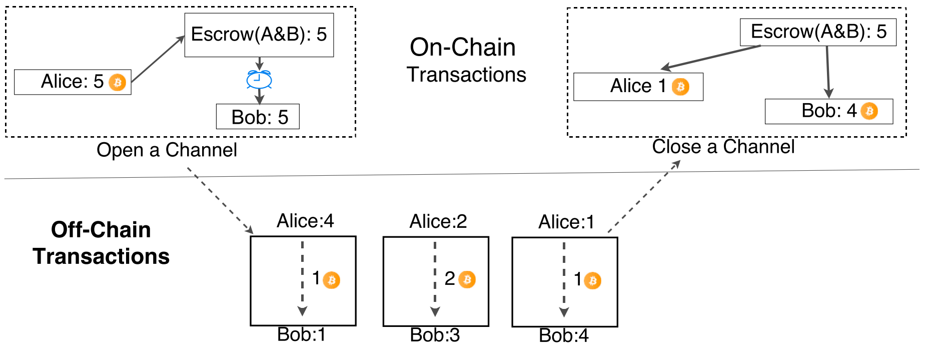

Fig. 1. Off-chain mechanism between two Blockchain nodes.

of LN is not strongly correlated with the development of the size of the network where capacity grows more slowly.

Works targeting the privacy of the LN is not common. Right now, due to being in the early development stage, LN enthusiasts have been discussing the privacy brought by LN on public forums or in communication channels. An attack to the undisclosed directional balance is introduced in (Herrera-Joancomarti et al., 2019). In this study, the authors send transactions with a never existing signature and observe the reaction of the nodes on the path. They increase the payment amount until they get an unsuccessful transfer information. Authors in (Rohrer et al., 2019) studies on three different snapshots of LN and calculates the robustness of the topology with respect to different known attacks and node failures in terms of privacy and transaction success rate.

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.

2.3. *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 (Even et al., 1975) 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 (Karakostas, 2008). 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 (Erdin et al., 2018). 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.

3. Background and motivation

3.1. *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 transaction is provided to the Blockchain for establishing a 2-of-2 multi-signature trustless escrow account, and future successive transactions take place using this shared account. The account activities are signed and tracked by the peers without being written to Bitcoin's public ledger. The amount put in the multi-signature account is decided individually by the participants 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 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

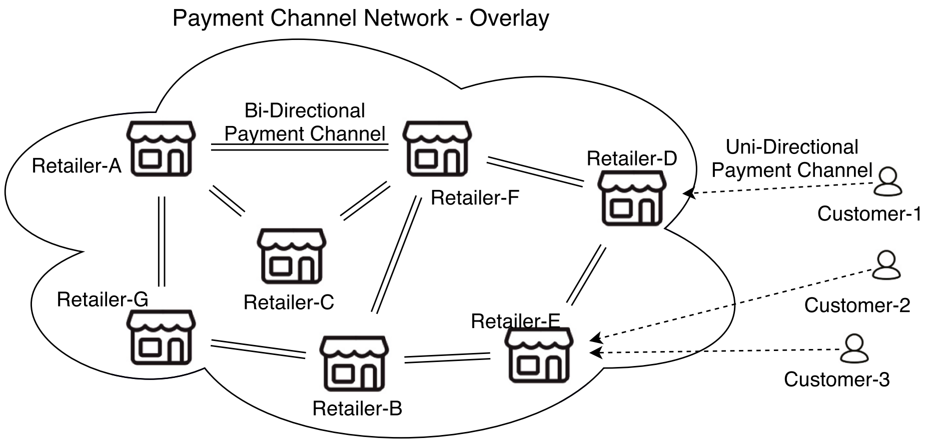

Fig. 2. An overview of the envisioned Payment Channel Network among retailers.

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)*" (Hash Time Locked Contracts). With every HTLC created by the peers, a peer gives ownership of some of her assets to the other if and only if she can fulfill the contract with a proof. The proof should be satisfied in a limited period of time. Keeping the state of the channel up-to-date is the duty of the channel owners.

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.

3.2. *Privacy in Lightning Network*

Bitcoin network is believed to be completely private by some of the Bitcoin enthusiasts. On the other hand, people with sceptical thoughts criticize Bitcoin to have no privacy. Although they put tangible ideas to prove their claims denying or accepting both are not suitable. For example, in Bitcoin Network tracing a series of payments might give clues about the assets of a user to some extent, but hiding those information is also possible when the pseudonymity idea is utilized properly.

By design, LN offers an additional level of anonymity for the user. In LN, the route for a payment is calculated by the sender by source-routing (Ethereum, 2019). The sender who has a topological view of the network, calculates the best route. The sender, then, encapsulates the route and sends the packet to the first node in the route. Thanks to encapsulation, all of the intermediary nodes only know the previous node from which the packet was sent and the next node to which the packet will be forwarded. This idea is very similar to the Tor network.

3.3. *Problem motivation and definition*

In this paper, we argue that current LN structure and features are not attractive for retailers to join it for their daily transactions. Specifically, we claim that retailers from certain business domains who would like to attract more business from cryptocurrency users could come together to form a **private** PCN that can be controlled and managed by them so that it can better satisfy customers' needs. In what follows, we explain the shortcomings of LN and justify the need for such a private PCN:

- **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 for a private PCN needs to guarantee network connectivity.
- **Network topology:** 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 experiencing 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, the attackers can easily analyze the payments passing through them which will expose the privacy of the customers using them as relays. The proposed topology for a private PCN needs to carry P2P features to prevent these issues.
- **Investment for each channel:** In LN, we mentioned that there is no guarantee for network connectivity. However, forming a connected network for our proposed PCN 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 in our proposed PCN may assume that it needs, say 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 already shown in (diar.co, 2018). For resolving this issue in the proposed PCN, either more investment should be planned in the channels in advance or there should be a reverse payment to balance the forward capacities. 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.

4. Proposed heuristic algorithm

In this section, we describe our proposed heuristic in more detail.

4.1. *Approach overview*

Our heuristic of PCN 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 first 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 are 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 remove any other channels.

In this heuristic, finding the path between a source and a destination retailer is crucial. When we look at 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

Table 1
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

options when finding a path.

We follow a similar rationale for our heuristic. Specifically, if there is a path from one retailer to another one, our heuristic uses that path by relying on a shortest path algorithm, namely Dijkstra's. If there is none, we open a new channel. Additionally, if total cost on the path will start to create inconveniences for intermediate retailers (i.e., adding a burden of forwarding), then we force our approach to open a new channel by adjusting the edge weights in the Dijkstra's shortest path algorithm. In a sense, we strive to find a sub-optimal approach for opening channels so that the participants of the network neither suffer from unfair load distribution nor pay excessive on-chain transaction fees. Next, we describe our heuristic details.

4.2. Finding paths

In order to find the best possible routes, Dijkstra's shortest path algorithm is used (Dijkstra, 1959). 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 a priori payment list. From the payment list, transactions are read one by one. At each reading, meaning iteration, a shortest (i.e., 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. The algorithm is shown in Algorithm 1 which utilizes the notation in Table 1.

Algorithm 1
Network Establishment

```
1: Input: $P$=Payment List, $H$=fully connected directed graph, $L_c$ = Link establishment
   cost, $W_i$ = New connection forcing cost
2: for every edge, $e$, in $H$ do
3:   $H_e.weight = W_i + L_c$
4:   $H_e.flow = 0$
5: end for//Initial assignments are done
6: for every payment in $P$ do//A payment is defined by a source, a destination and the
   transfer amount $T_a$
7:   $Path = ShortestPath(H, \text{from} = a, \text{to} = b)$
8:   for Each edge, $e$, in Path do
9:     $H_e.flow += T_a$
10:    $H_e.weight = W_i + H_e.flow$
11:  end for
12: end for
13: for All edges in $H$ do
14:   if $H_e.flow = 0$ then
15:     Remove edge from $H$
16:   end if
17: end for
18: 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. Specifically, in order to minimize the impact of dependence on the order, at each round, the order of the retailers is renewed with a new distribution. This approach is shown in Algorithm 2.

Algorithm 2
List Establishment.

```
1: Input: S=Set of Retailers
2: while All required payments are not fulfilled do
3:   TempS=S
4:   while TempS is not Empty do
5:     Pick 2 random retailers (a,b) from TempS
6:     if Transaction from $a$ to $b$ was not fulfilled then
7:       Add $a$ as source, $b$ as target in $P$
8:       Remove (a,b) from TempS
9:     end if
10:  end while
11: end while
12: Output: $P$=Payment List
```

According to this approach, first, two 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, 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 visit is complete, meaning the list of retailers is an empty set, the procedure is repeated. This new random list of payments is then fed to Algorithm 1.

4.3. Defining edge weights

As mentioned, after finding a path the edge weights need to be updated to inject the desired 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*. Below, we explain them next in more detail.

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 heuristic, all edges in the network have a non-negative L_c set to some value. Whenever an edge is used (i.e., 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 due to higher L_c. As some edges will have lower weight due to nullified L_c, they will be used heavily. 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

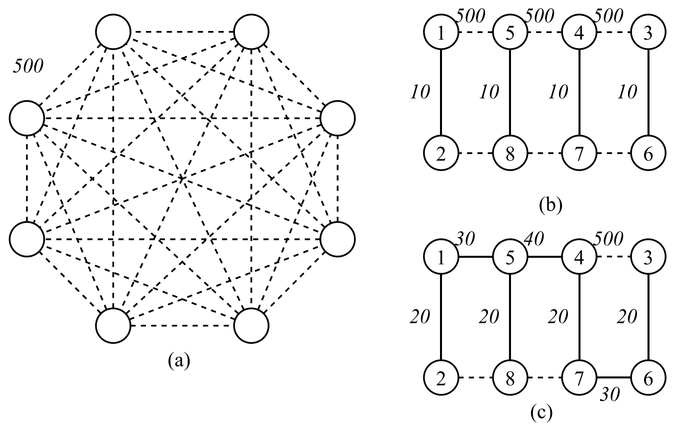

Fig. 3. Initially fully connected mesh network topology.

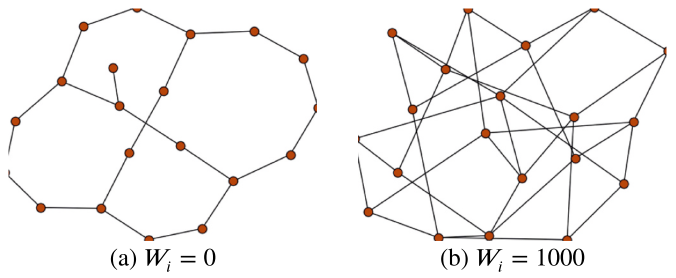

Fig. 4. Effect of W_i for a network of 20 nodes, $L_c = 3000$.

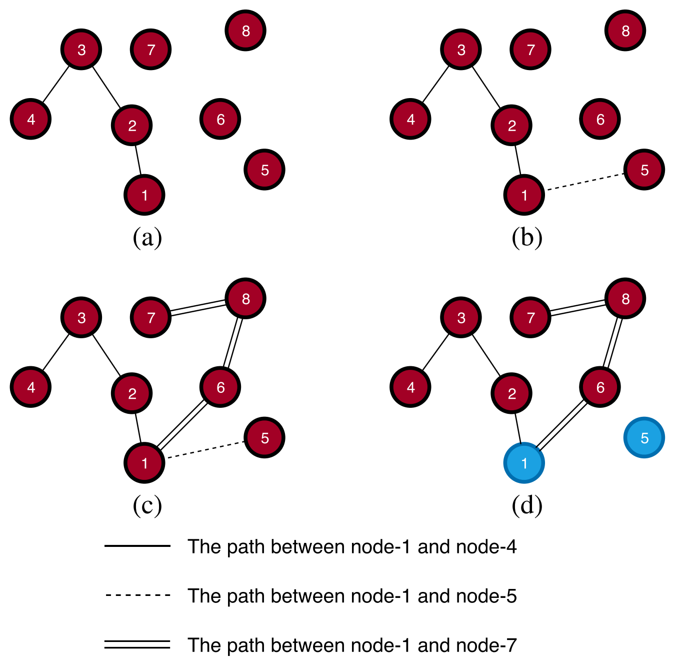
(a)
(b)
(c)
(d)
The path between node-1 and node-4
The path between node-1 and node-5
The path between node-1 and node-7

Fig. 5. Explanation of the rerouting mechanism.

edge, the amount transferred incurs a weight on that edge which is basically a *transaction 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}, is revised as follows to accommodate this transaction cost:

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

where U_{AB} is a binary variable and equals to 1 if there happens to be a flow on edge AB anytime during the procedure, and 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 investment 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 fully connected mesh 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 initiate 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, not all of the $L_c = 500$ wt 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. For a larger N, we will observe longer 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 longer paths will be hampered. Thus, the weight on edge (A, B) will be updated as follows:

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

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. 4(a). Introducing W_i and setting it to 1000 causes the topology to change to the one in Fig. 4 (b). We argue that Fig. 4(a) is not a desired topology because it is weak against node failures, and some nodes are highly centralized.

5. Extending the heuristic for privacy guarantees

The higher the number of hops a payment is traversing through, the better the privacy is. This is inspired by the idea of privacy in the Tor network where each message needs to traverse at least 3-hops (Dingledine et al., 2004). In LN the payments are transferred from a source to a destination within encapsulated messages. If a node in the center knows that it is the node in between the source and the destination it will be able to gather information about the users. This also comes with additional traceability of the payments because there is a possibility that the nodes can be traced back from the Bitcoin network. However, if at least 3-hops transaction mechanism is utilized or enforced by the users, a node on the route of the payment will not be able to get a clear view of the source and the destination about the payment (Dingledine et al., 2004).

While the Dijkstra's shortest path algorithm is highly efficient in finding the shortest path from a source to a target in weighted directed graphs, it cannot guarantee a minimum number of hops for a path. In order to increase the privacy of payments, we propose an extension to it by relying on re-routing. Specifically, the proposed extension method is comprised of two steps. In the first step, Dijkstra's algorithm is run with the default configuration as defined Algorithm 1, and all the shortest paths for payments are found. Note that, in this configuration, there is no dictated number of hops requirement. Among all of those paths, the ones with lower than R hops are saved in a temporary list, which will be used in the second step of the method.

In the second step, the shortest path algorithm is run again for the payments in the temporary list. However, in this case, instead of finding the shortest path, k-shortest paths algorithm is utilized. k-shortest path algorithm basically lists top k shortest paths from a source to destination which requires more computation. The computational complexity of the k-shortest path algorithm is shown to be $O(E + kNlogN)$ (Bouillet et al., 2007) where E represents the number of edges and N is the number of vertices in the graph, and k is the number of hops required. The goal here is to look for paths with at least R-hops, essentially forcing the paths that are in our list to perform a *re-routing*. Among all of the paths with R or more hops, the path with the minimum weight is picked as the solution. This process is hypothetically shown in Fig. 5. Based on this figure, during the Dijkstra's algorithm run, a 3-hop path between node-1 and node-4 is found as shown in Fig. 5(a). As the iterations carry on, a 1-hop algorithm is found between node-1 and node-5 as shown in Fig. 5(b) and in the next iteration another 3-hop path is found between node-1 and node-7 as illustrated in Fig. 5(c). Thus, when the first step is over, the re-routing with the k-shortest path algorithm starts for the 1-hop path found in Fig. 5(b). This 1-hop path should be forced for a re-route via more than R hops between node-1 and node-5 as shown in Fig. 5(d).

One might question why the first step of the proposed method is not directly utilizing k-shortest paths algorithm instead of following a two-step approach. This choice is due to the computational complexity that comes with k-shortest path algorithm with the initial topology. It takes a lot of time to find a path for the payments essentially turning the approach into a brute force search. The proposed privacy guaranteed method is shown in Algorithm 3.

Algorithm 3
Network Establishment for Privacy Guarantees.

```
1: Input: $P$=Payment List, $H$=fully connected directed graph, $L_c$ = Link establishment
   cost, $W_i$ = New connection forcing cost
2: for every edge, $e$, in $H$ do
3:   $H_e.weight = W_i + L_c$
4:   $H_e.flow = 0$
5:   $H_e.Tempflow = 0$
6: end for//Initial assignments are done
7: for every payment in $P$ do//A payment is defined by a source, a destination and the
   transfer amount $T_a$
8:   $Path = ShortestPath(H, from = a, to = b)$
9:   if length($Path$) < $R$ then
10:     Add payment to Temp list
11:     for Each edge, $e$, in $Path$ do
12:       $H_e.Tempflow += T_a$
13:       $H_e.weight = W_i + H_e.Tempflow + H_e.flow$
14:     end for
15:   else
16:     for Each edge, $e$, in $Path$ do
17:       $H_e.flow += T_a$
18:       $H_e.weight = W_i + H_e.flow + H_e.Tempflow$
19:     end for
20:   end if
21: end for
22://Remove effect of $H_e.Tempflow$ in $H$
23://Nullify Tempflow and make edge weights $L_c + W_i$ of edges with only positive Tempflow
24: for every payment in Temp do
25:   $AllPaths = AllSimplePaths(H, from = a, to = b)$
26:   $Path = x$ where $x \in AllPaths$ if length($x$) $\geq R$ and $weight_x$ is the minimum
27:   for Each edge, $e$, in $Path$ do
28:     $H_e.flow += T_a$
29:     $H_e.weight = W_i + H_e.flow$
30:   end for
31: end for
32: for All edges in $H$ do
```

(continued on next column)

Algorithm 3 (*continued*)

```
33:   if $H_e.flow = 0$ then
34:     Remove edge from $H$
35:   end if
36: end for
37: Output: $H$
```

6. Evaluation

In this section, we describe the experiment setup, performance metrics and discuss the evaluation results.

6.1. 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 that peers of a channel put the same amount in the channel they created. The proposed approach is implemented in Python and its performance is assessed extensively through various experiments. All the experiments are carried out on a computer with an Intel Xeon E5-2630 v4 @ 2.20 GHz CPU and 64 GB of RAM.

6.2. Metrics and benchmarks

The results of the experiments are assessed 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 follow 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:

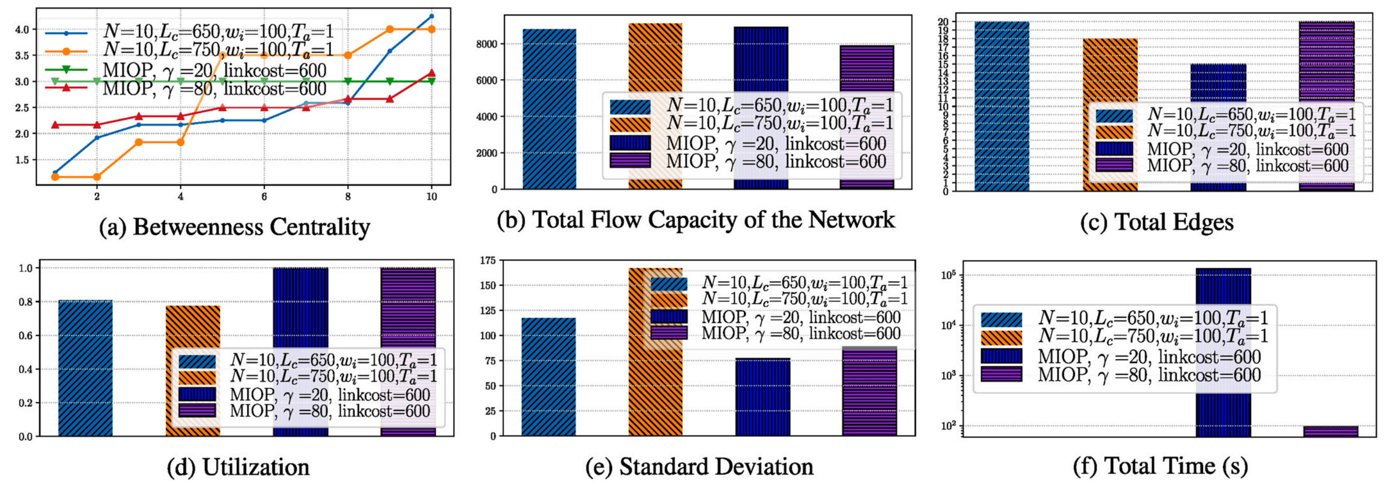

Fig. 6. Optimal vs. Heuristic Comparisons.

- **MIOP model**: The results of the heuristic are compared with an optimization model in (Erdin et al., 2018).
- **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 flows in the network.

6.3. *Experiment results and discussion*

6.3.1. *Comparison of heuristic with the MIOP model*

In this section, the results of the proposed heuristic approach are presented and compared with that of the MIOP model studied in (Erdin et al., 2018). The objective is to assess our approach's performance with respect to the ideal one. 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 useable. 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, the 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 betweenness centrality, standard deviation and number of edges. For the results of the heuristic approach, the same scenario is inherited. All the related results are shown in Fig. 6. In those figures, γ is a control parameter for the unfairness among node outbound flows, and *linkcost* is the link establishment cost, L_c in MIOP.

As can be seen from Fig. 6(b) and (c), our heuristic's performance 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. 6(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. 6(e). However, when W_i is 100 and L_c is 650 in the heuristic approach, standard deviation 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. 6(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. 6(f)). In summary, the proposed approach provides the same features as MIOP in a much faster/scalable manner but with some slight deviation from an ideal P2P topology.

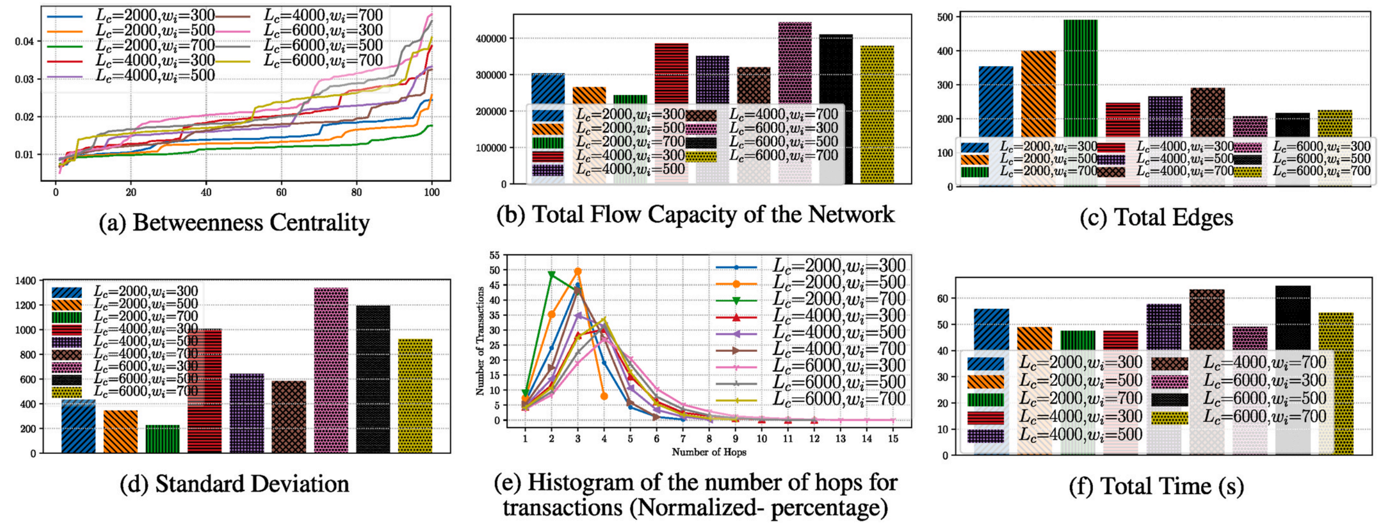

Fig. 7. Ideal parameter selection.

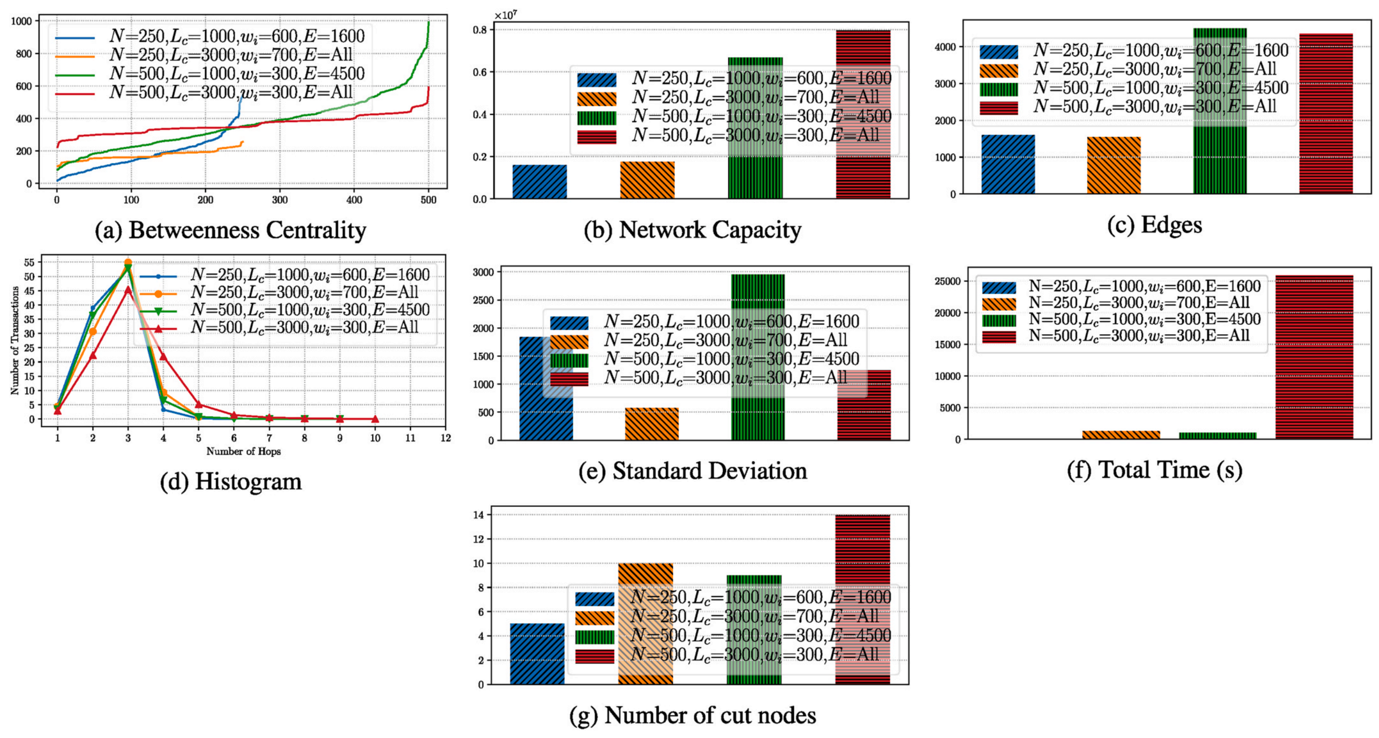

Fig. 8. Scalability test results.

6.3.2. *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. 7.

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 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. These parameters are used in the remaining experiments.

6.3.3. *Scalability of the heuristic*

In this experiment, we assessed the scalability features of the proposed heuristic. Specifically, the heuristic approach is run with different numbers of nodes, namely 250 and 500 nodes. However, as the number of nodes increases, the computation time required to finish the calculations increases 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 $O(N^2logN)$. In order to decrease the effect of the assumption of an initially fully connected mesh network, we also created networks with random initial connections as an alternative approach for comparison. To differentiate these two, in the figures, the initial number

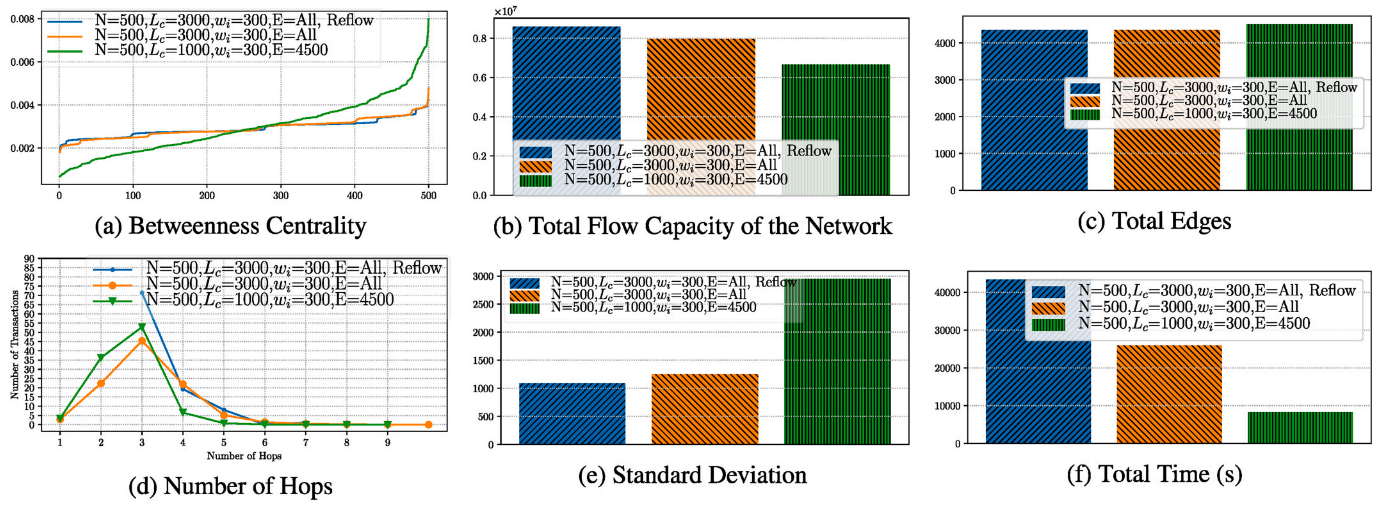

Fig. 9. Results for the Privacy aware method for 500 nodes.

of edges are depicted with the E parameter where $E = All$ indicates our approach with a fully connected network.

The results are shown in Fig. 8. Pre-pruned topologies (i.e., randomly connected) give an advantage in terms of total computation time, as expected, especially with 500 nodes. However, other results are generally slightly better for the initially fully connected mesh network setup. In particular, standard deviation of our approach with a fully connected topology is significantly reduced. 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 looked at the number of cut nodes. Fig. 8(g) 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 randomly connected network, the possibility of taking down the network is easier because attacking fewer nodes will be enough. This is not the case in our heuristic with the fully connected mesh network 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. This is one of the main strengths of our approach in terms of producing a good topology against DDoS attacks. We can conclude that up to a certain number fully connected topology might be a better choice. However, when we move beyond a certain number of nodes, for time savings, pre-pruned topologies may be preferred based on their cut node performance.

6.4. *Heuristic with privacy considerations*

In this subsection, we present results related to privacy extension of our approach presented in Section 5 which guarantees payment privacy with at least 3-hop payments. In these experiments, we compared two other baseline approaches with ours: The first baseline approach utilizes our Dijkstra heuristic without any privacy guarantees starting with a fully connected initial topology shown with $E = All$ in the figures. The second baseline is the same as the first except that it starts with a randomly connected initial topology with 4500 edges (shown as $E = 4500$). Our approach which guarantees privacy is shown as $E = All, Reflow$. We conducted the experiments with 500 nodes. The results are shown in Fig. 9 for the following metrics: betweenness centrality, total investment capacity, the total number of edges, the number of hops, the standard deviation and total processing time.

As seen in Fig. 9(a), with a randomly connected initial topology, the betweenness centralities of the nodes change drastically, which is expected. This is because with the random connection some of the nodes will be apparently dominant compared to others. However, betweenness centrality of the initially fully connected mesh network topologies gives a more flat measure. This behavior assures that the dominance of some of the nodes is decreased significantly. The initial fully connected mesh network topology and pseudo-random payment distribution is effective in getting that result. However, our approach with privacy guarantees is slightly better than the one without privacy for betweenness centrality. As shown in 9(c), while the number of edges did not change, the betweenness centrality measure is maintained. This is because some of the channels opened in the first step of the privacy approach are discarded in the second step which eventually helped in establishing a more balanced network.

The other results for the experiment with the privacy guarantee are promising too. Although the number of edges does not change, the main factor for the slight increase in the total capacity with respect to the approach where privacy is not guaranteed is the forcing of minimum 3 hops in the transfers which causes each node to invest more, meaning an additional investment for the others' payments too. Nonetheless, our privacy-guaranteed approach comes with the best standard deviation, even surpassing the non-privacy one as shown in Fig. 9(e). The one-time cost for these improvements comes with some time overhead as seen in Fig. 9(f). The total calculation time for the setup with the random connection network is the lowest because the complexity of the Dijkstra is directly related to the number of edges. For the privacy guaranteed approach, the time is the highest because all of the 1 and 2 hops payments were converted to at least 3 hops transfers by the k-shortest path algorithm which in turn brings an additional overhead to the total computation time.

7. Conclusion

Cryptocurrency based payment channel networks using the idea of off-chain payments has been emerging recently. This is not only because they reduce confirmation times but they also let users send micropayments in a very affordable way. Therefore, forming a reliable and scalable P2P payment network is an open question assuming a private consortium of retailers (nodes). In this study, based on some scenarios and assumptions, we developed a heuristic approach to form such a payment network topology using Bitcoin's off-chain concept and Dijkstra's shortest path routing and compared the results with the results of an optimal solution. We further extended this heuristic to guarantee privacy-aware routing in the network with at least 3 hops and compared its performance with the non-privacy case.

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. When privacy is to be guaranteed with at least 3 hops, the results show that the network topology becomes even better in terms of the considered topological metrics with some additional computation time overhead.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This study was partially funded by the National Science Foundation under award number NSF-CNS-1663051.

References

Bitcoin wiki. Bitcoin Contract. en.bitcoin.it/wiki/Contract.
BitInfoCharts, 2017. bitinfocharts.com/comparison/bitcoin-transactionfees.
Bloomberg, 2017. www.bloomberg.com/view/articles/2017-11-14/bitcoin-s-high-transaction-fees-show-its-limits.
Bouillet, E., Ellinas, G., Labourdette, J.F., Ramamurthy, R., 2007. Path Routing–Part 2: Heuristics.
Cebe, M., others, 2018. Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles. IEEE Commun. Mag. 56 (10), 50–57.
diar.co, 2018. Lightning Strikes, but Select Hubs Dominate Network Funds. https://diar.co/volume-2-issue-25.
Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271. https://doi.org/10.1007/BF01386390.
Dingledine, R., Mathewson, N., Syverson, P., 2004. Tor: the second-generation onion router. In: SSYM'04. USENIX Association. Berkeley, CA, USA: 21–21.
Erdin, E., Cebe, M., Akkaya, K., Solak, S., Bulut, E., Uluagac, S., 2018. Building a private Bitcoin-based payment network among electric vehicles and charging stations. IEEE International Conference on Blockchain.
Ethereum, 2019. BOLT 4: Onion Routing Protocol. https://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md.
Even, S., Itai, A., Shamir, A., 1975. On the complexity of time table and multi-commodity flow problems. IEEE 184–193.
Hackius, N., Petersen, M., 2017. Blockchain in logistics and supply chain: trick or treat? epubli 3–18.
Haghani, A., Oh, S.C., 1996. Formulation and solution of a multi-commodity, multi-modal network flow model for disaster relief operations. Transport. Res. Pol. Pract. 30 (3), 231–250.

Hash Time Locked Contracts. en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.
Herrera-Joancomarti, J., Navarro-Arribas, G., Pedrosa, A.R., Cristina, P.S., Garcia-Alfaro, J., 2019. On the Difficulty of Hiding the Balance of Lightning Network Channels. AsiaCCS.
Karakostas, G., 2008. Faster approximation schemes for fractional multicommodity flow problems. ACM Trans. Algorithm 4 (1), 13.
Kuo, T.T., others, 2017. Blockchain distributed ledger technologies for biomedical and health care applications. J. Am. Med. Inf. Assoc. 24 (6), 1211–1220.
Malavolta, G., others, 2017. Concurrency and privacy with payment-channel networks. In: ACM, pp. 455–471.
Martinazzi, S., 2019. The Evolution of Lightning Network's Topology during its First Year and the Influence over its Core Values arXiv preprint arXiv:1902.07307.
Miller, A., Bentov, I., Kumaresan, R., McCorry, P., 2017. Sprites: Payment Channels that Go Faster than Lightning. CoRR abs/1702.05812.
Poon, J., Dryja, T., 2015. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. Technical Report (draft).
Prihodko, P., others, 2016. Flare: an Approach to Routing in Lightning Network.
Raiden, 2018. raiden.network/.
Rohrer, E., Malliaris, J., Tschorsch, F., 2019. Discharged payment channels: quantifying the lightning network's resilience to topology-based attacks. In: 2019 IEEE European Symposium on Security and Privacy Workshops.
Roos, S., others, 2017. Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions arXiv preprint arXiv:1709.05748.
Seres, I.A., Gulyás, L., Nagy, D.A., Burcsi, P., 2019. Topological Analysis of Bitcoin's Lightning Network arXiv preprint arXiv:1901.04972.
Team LN. Atomic Cross-Chain Trading….
Thomas, S., Schwartz, E., 2015. A Protocol for Interledger Payments interledger.org/interledger.pdf.
TrustNodes, 2018. Lightning Network DDoS Sends 20% of Nodes Down trustnodes.com/2018/03/21/lightning-network-ddos-sends-20-nodes.
Zebpay, 2017. Bitcoin the Next Big Thing since the Internet Itself: Christine Lagarde. https://blog.zebpay.com/bitcoin-the-next-big-thing-since-the-internet-itself-christine-lagarde-4d4ba71d80dc.

Enes Erdin is an Assistant Professor in the Computer Science Department at University of Central Arkansas, Conway. He conducts research in the areas of hardware security, blockchain technology, and cyber-physical systems. Erdin received a Ph.D. in Electrical and Computer Engineering from Florida International University, Miami where he was a NSF CyberCorps fellow.

Mumin Cebe is an Assistant Professor in the Computer Science Department at Marquette University, Milwaukee. He conducts research in the areas of blockchain, wireless networking, and security/privacy that relates to the Internet of Things and cyber-physical systems, particularly in smart grids and vehicular networks. Cebe received a Ph.D. in Electrical and Computer Engineering from Florida International University, Miami.

Kemal Akkaya (A′08–M′08–SM′15) received the Ph.D. degree in computer science from the University of Maryland, Baltimore, MD, USA, in 2005. He joined, as an Assistant Professor, the Department of Computer Science, Southern Illinois University Carbondale (SIU), Carbondale, IL, USA, where he was an Associate Professor from 2011 to 2014. He was also a Visiting Professor with George Washington University, Washington, DC, USA, in 2013. He is currently a Professor with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA. His current research interests include security and privacy, energy aware routing, topology control, and quality of service issues in a variety of wireless networks. He was the recipient of the Top Cited Article Award from Elsevier in 2010. He is currently an Area Editor for the Elsevier Ad Hoc Network journal, and is on the Editorial Board of the IEEE Communication surveys and tutorials.

Eyuphan Bulut (M′08) received the Ph.D. degree in computer science from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2011. He was then a Senior Engineer with Mobile Internet Technology Group group, Cisco Systems, Richardson, TX, USA, for 4.5 years. He is currently an Assistant Professor with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA. His research interests include mobile and wireless computing, network security and privacy, mobile social networks, and crowd-sensing. He has been an Associate Editor for IEEE Access.

Selcuk Uluagac (suluagac@fiu.edu) is an associate professor in the Department of Electrical and Computer Engineering at Florida International University, Miami, where he leads the Cyber-Physical Systems Security Lab. His research focuses on security and privacy for the Internet of Things and cyberphysical systems, and he has many publications on the practical and applied aspects of these areas. Uluagac received a Ph.D. in electrical and computer engineering from the Georgia Institute of Technology, Atlanta.