Computer Networks 172 (2020) 107098

Contents lists available at ScienceDirect

Computer Networks

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

A Bitcoin payment network with reduced transaction fees and confirmation times

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

[a] Florida International University, Miami, FL 33172, United States
[b] University of Massachusets Amherst, Amherst, MA 01003 United States
[c] Virginia Commonwealth University, Richmond, VA 23284 United States

ARTICLE INFO

Article history:
Received 10 July 2019
Revised 10 December 2019
Accepted 2 January 2020
Available online 9 January 2020

Keywords:
Bitcoin
Lightning network
Integer programming
Network formation

ABSTRACT

The high transaction fees and confirmation times made Bitcoin unfeasible for many applications when the payments are in small amounts and require instant approval. As a result, many other cryptocurrencies were introduced for addressing these issues, but the Bitcoin network is still the most widely used payment system. Without doubt, to benefit from its network of users, there is a need for novel solutions that can successfully address the problems about high transaction fees and transaction verification times. Recently, payment network ideas have been introduced including the Lightning Network (LN) which exploits off-chain bidirectional payment channels between parties. As off-chain links can be configured to perform aggregated transactions at certain intervals without writing to blockchain, this would not only reduce the transaction fees but also decrease the verification times significantly. Nevertheless, LN still deploys relay nodes which charge fees and its growth leads certain nodes to become monopolies which defeat the very purpose of decentralization. Despite the thought that LN will provide a "scale-free network mechanism" along with high decentralization, how to form such a network among multiple parties is never studied before. Therefore, in this paper, by exploiting the LN, we offer to form a purely decentralized payment network which will allow Bitcoin to handle a lot of transactions with enough capacity. The idea is to form a network of retailers (i.e., nodes) which do business with Bitcoin by connecting them through off-chain links (i.e., payment channels) based on the business needs. The problem is initially modeled as a network optimization but since scalability would be a concern the next steps follow a heuristic approach where certain links are pruned to force evenly distributed payment flows while minimizing the total investments made to create initial off-chain links. The evaluations demonstrate the extent of scalability of the network along with the trade-offs between the distribution of flows and the initial flow capacities of the nodes.

1. Introduction

Since its introduction, 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 for data storage and bookkeeping. Thanks to such data structure, Blockchain is now touted as an innovative solution that can be used in many areas such as healthcare, finance, government operations, logistics, etc. [1–3].

While Bitcoin has opened new opportunities such as pure peer-to-peer (P2P) money transaction and transparency against censorship, it has been long criticized for its slow transaction confirmation times and high transaction fees [4,5]. The transactions in Bitcoin are written within a block and these blocks are typically verified by nodes that are referred to as miners. In Bitcoin, a block creation time is, by design, around 10 min, and the general heuristic for accepting a transaction to be valid is when the transaction is 6 blocks-old which yields 60 min on average. However, as the miners are inclined to give priority to the transactions which offer higher fees, a typical transaction will take longer to be approved during congested times. In that case, either the sender should agree to pay a high fee or s/he should wait until the transaction request is accepted by a miner. During once of such an extreme case, in 2017 Bitcoin boom, the payees had to pay more than

* Corresponding author.
E-mail address: eerdi001@fiu.edu (E. Erdin).

https://doi.org/10.1016/j.comnet.2020.107098
1389-1286/

$20 for transaction fees or they had to wait 1188 min on average [6]. Therefore, such transaction confirmation times are not suitable for applications where timely payment evidence is critical. In addition, the transaction fees are not proportional to the transaction amounts and this makes Bitcoin ineligible for many daily micropayments such as buying coffee or paying for car charging service via smart apps, etc. The transaction fee in Bitcoin is directly related to the hard-coded block size limit, and it will not change in the foreseeable future.

As Bitcoin is still the most widely used digital currency, and its market cap is above 50% among all digital currencies, it makes perfect sense to exploit this market cap and try to alleviate the above problems of Bitcoin. To this end, recently the concept of Lightning Network (LN) is introduced [7]. The idea in this concept relies on payment channels that enable *off-chain* transactions through multi-signature (escrow) accounts. In this way, for an agreed term two parties can perform regular transactions in real-time without a need to write it back to the Blockchain. The total net transactions can be computed at the end of the agreed term and the final net transaction can be committed in the Blockchain at that time (i.e., *on-chain*). In this way, one can avoid most of the transaction fees that are conducted within the agreed term because off-chain payment channel requires typically two on-chain transactions; one for opening the channel and one for closing. Also, as transactions are not always written to the Blockchain, no party will experience slow verification times. Current LN applies 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. This is essentially a payment network which currently serves more than 7000 nodes.

However, instead of connecting retailers and customers directly, LN relies on *relay nodes* which act as bridges between parties. Eventually, these relay nodes become powerful hubs which forward most of the payments and hence may charge high forwarding fees. Obviously, this is against the very idea of reducing the transaction fees for creating such a payment network. Furthermore, allowing the relay nodes to become monopolies in forwarding poses vulnerabilities for denial of service (DoS) attacks and privacy analysis of customers' transactions. For instance, there was a recent DDoS attack on LN which took 20% of the nodes down and gave hard times to the transactions [8]. In addition to these issues with the structure, there are also issues with routing of payments. Any payment made in the current LN has a chance of less than 1% to make it to its destination if the transaction amount is greater than $200 [9]. This would obviously be a big problem for a retailer that needs to collect payments without having any issues on time.

Hence, there is a need for a highly decentralized topology not only for attack/privacy protection but also minimization/elimination of possible unfair transaction costs while also guaranteeing routing of the payments. To this end, we advocate creating a *private payment channel network* that will bring together retailers under a *consortium* to contribute to this payment network as opposed to relying on the unreliable public LN. In this way, micropayments will be available for customers along with certain guarantees to retailers in terms of reliability, privacy, and monopolies.

In this paper, we propose to build such a private payment channel network topology from scratch using the off-chain concept of Bitcoin. Our objective is to distribute 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 formation. By inspiring from the multi-commodity flow problem [10] in real-life, we start with an optimization model that will optimally distribute the flow within an initial network topology. However, since the multi-commodity flow problem is NP-complete [11], the solution will not scale if all potential channel establishments are done on the initial network.

We thus start pruning this network's links so that we can pick the right network topology by monitoring certain metrics instead of following a brute force approach. In addition, we also minimize the number of channels to reduce network formation costs. However, pruning needs to be done carefully as it may favor some of the channels or paths that will create an unfair distribution (i.e., create hub nodes and put a burden on these hubs). Therefore, we consider several criteria in pruning and continue such a process until we achieve a certain standard deviation among the capacities of the channels. We also consider the trade-offs between the capacity and standard deviation.

Our contributions in this paper are as follows:

- We developed a linear programming model to fairly distribute the money flow in the payment network among the nodes.
- We developed a mixed-integer programming model which decides to establish channels between nodes and hence a P2P topology at the end.
- We further developed a heuristic approach to overcome the performance bottleneck in the mixed-integer programming model.

The evaluations using *Python* and *Gurobi solver* indicated that our proposed heuristics can provide comparable performance to that of the optimal solution while allowing scalability. In addition, the heuristic creates purely distributed topologies that are resilient towards DDoS attacks and potential privacy leakages.

This paper is organized as follows: Next section summarizes the related work and in Section 3, we give the Background and Preliminaries for understanding the related concepts. Section 4 explains the proposed optimization approach and Section 5 presents a heuristic for improving the running time of the optimization approach. Section 6 discusses the experiment results. Paper is concluded in Section 7.

2. Related work

2.1. Payment channel networks

There are several recent efforts in both industry and academic community to address high transaction fees and slow confirmation times and scalability issues of cryptocurrencies especially Bitcoin. One of these efforts is building a Payment Channel Network (PCN) as done in this study. Off-chain payment channel method was initially introduced by the Bitcoin community [12], and LN is the first widely deployed implementation utilizing off-chain payment channel concept to establish a PCN. The community is actively enhancing the idea, and they are introducing new concepts in order to increase the efficiency of the network like channel factories, watchtowers, macaroons, etc. [13].

PCNs can be classified into two categories. The first category relies on building a PCN for intra-blockchain operations. LN and Raiden are examples that fall into this category [7,14]. LN allows transferring Bitcoin between parties over already existing off-chain links without any confirmation delay but with some forwarding fees. The similar idea is followed by Raiden to build a PCN for Ethereum [14]. 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 [15] and Atomic-CrossChain [16].

Our work in this paper is within the first category but it is a special one where the members are not public as in the case of LN or Raiden. We advocate the need for a private PCN which will serve the need of the members of a business consortium based on the application domain.

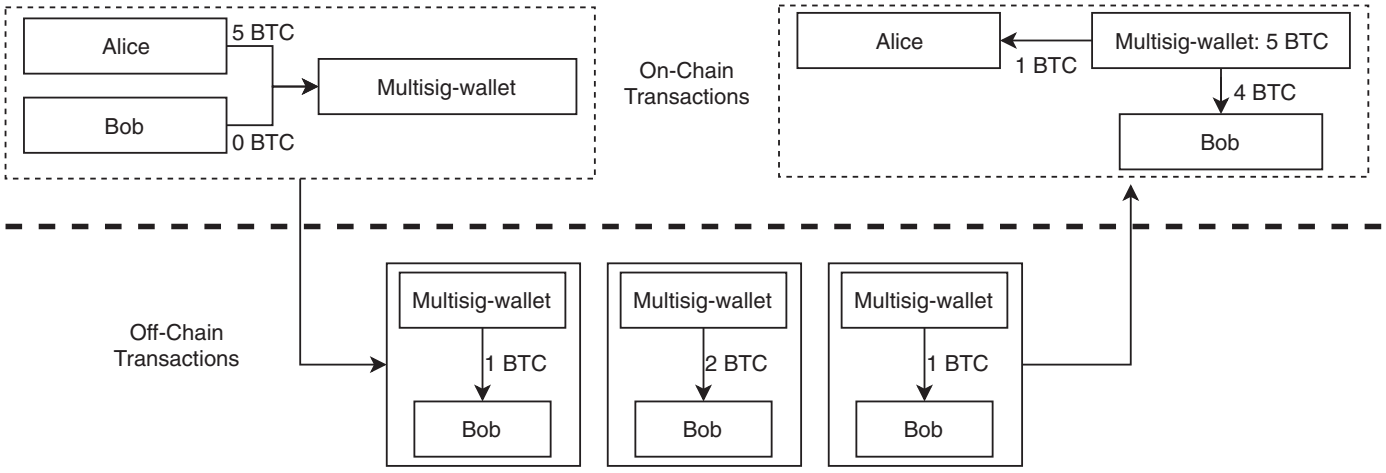


Fig. 1. Off-chain mechanism between two Blockchain nodes.

2.2. Lightning network

Lightning network is a (layer-2) PCN using Bitcoin Network as a foundation. The previous section included privacy related studies on LN. In addition to that, studies carried on other properties of the LN are relatively new. This is due to the fact that codebase for LN is still being developed and transition from an idea into practice has not finished. Research on topological observations on LN is not seen frequently due to the fast-changing behavior of the network, but efforts are being observed. For instance, in [17] the authors make a topological analysis of a snapshot of the LN taken in March 2018. They provide a brief mathematical investigation on the robustness of the LN by looking at its reaction against random failures and targeted attacks. 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 [18] statistically looks at the development of the LN in the course of 12 months since its establishment. With an interesting finding, 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.

We would like to note that our work in this paper does not offer any changes/improvements to LN. It utilizes LN's features to build a new and separate payment network similar to LN but have specific desirable characteristics that will serve its members needs. For instance, our proposed payment network will enable connectivity among any customer and retailer where retailers share the channel creation costs fairly and thus eventually forwarding fees may be opted out by the retailers (except the opening and closing fees for off-chain channels). We also provide guarantees for a connected network be created among every retailer which enables access to customers as long as they have a channel with one of these retailers.

The flow portion of our problem is very similar to multi-commodity flow problem which deals with the assignment of commodity flows from sources to destinations in a given network where the problem has applications in various domains including transportation, logistics, and telecommunications. The problem has been shown to be NP-Complete [11] even if the number of commodities is two. When the flows in the solution to the problem become fractional, a linear programming model can be designed which can be solved in polynomial time [19]. However, our problem is different as we not only optimize the flows in the network but also decide the existence of channels between peers.

3. Background and preliminaries

3.1. Background on off-chain transaction channels

The main motivation of this work comes from the concept of *off-chain transaction channels* mechanism [20,21] that is used for saving transaction fees in the current Bitcoin system as shown in Fig. 1. 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 kept mutual without being written to Bitcoin's public ledger. The amount put in the escrow account is contributed by both parties and unless that amount is reached, the transactions can continue. In this way, the participants typically pay fees for two on-chain transactions: one to open the channel and one to close it.

The example shown in Fig. 1 illustrates this concept in more detail. Specifically, Alice opens an off-chain channel by instantiating an escrow account with Bob, and they both sign this new account separately. Alice then deposits 5 Bitcoins to the escrow account by performing an on-chain transaction. This determines the channel capacity from Alice to Bob as 5 Bitcoins. Now, Alice can make many payments to Bob by transferring Bitcoins from the escrow account to Bob until the capacity of the channel is exhausted. In the figure, we see 3 transactions at different times: 1, 2, and 1 Bitcoins. Eventually, when the channel is closed, only the remaining Bitcoins (e.g., 1 Bitcoin) and the total transferred Bitcoins (4 Bitcoins) are committed respectively to Alice and Bob and written to the public ledger. There is no way to accomplish a transaction volume more than 5 between Alice and Bob without depositing more Bitcoins to the escrow account. The payment channel provides guarantees to the peers 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 special smart contract called "*Hashed Timelock Contracts (HTLC)*" [22].

PCNs such as LN exploit the off-chain mechanism to create multi-hop payment paths between participants. These paths are used for accomplishing the Bitcoin transfer between arbitrary parties. To enable this idea in practice, users are supposed to route their payments to any destination through a series of payment channels. 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. For instance, in Fig. 2, Alice can make a transfer to Bob via Carol using the off-chain links which are already established. Assuming already

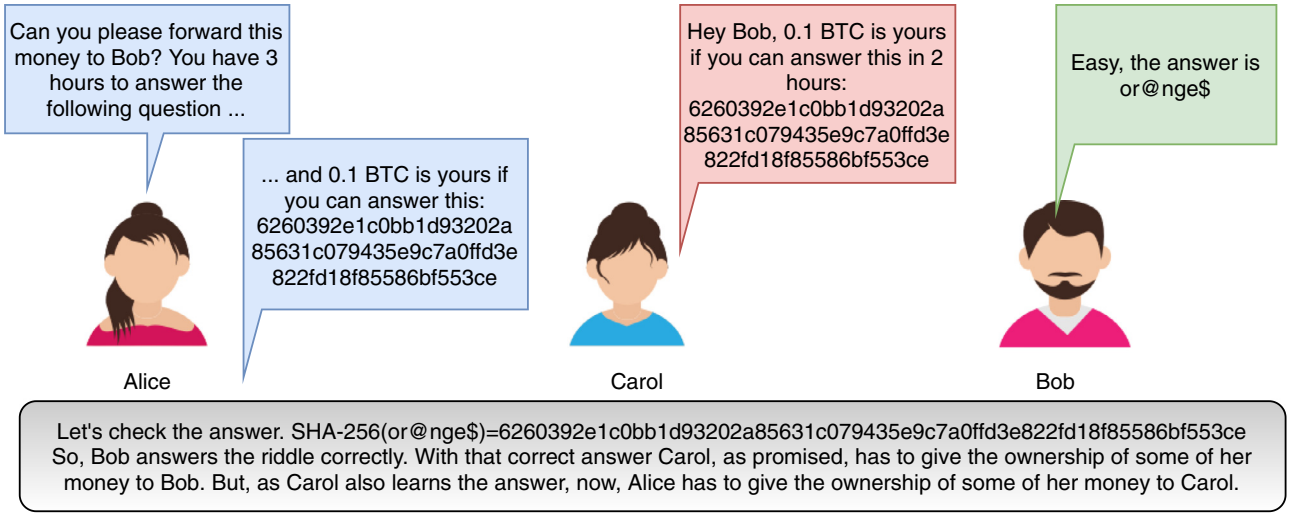


Fig. 2. Sample payment network using off-chain links.

established and funded channels, Alice-to-Carol and Carol-to-Bob, the payment between Alice and Bob is accomplished in the following way: Bob sends a hash of a secret (i.e., a disposable key) to Alice. First, Alice signs a commitment transaction destined to Bob and forwards it to Carol. Here HTLC comes to the scene. When Alice is forwarding the money to Carol, she locks the payment with the secret she received from Bob. Hence, unless Carol has the key in hand, she will not be able to unlock the payment. In the second hop, Carol commits a new transaction locked by the same secret and forwards it to Bob. To receive the payment, Bob has to reveal the secret which he generated at the very first stage of the payment. When Bob reveals the secret (a.k.a. pre-image), he unlocks the contract and gets his payment from Carol. As Bob revealed the secret, now, Carol knows it and by revealing the secret to Alice, she unlocks her share and gets her money from Alice. After every transaction, the peers update the state of the channel with new secrets in a trustless manner.

The senders also attach a deadline to the Hash-Lock that is agreed while establishing the channel which is the "Time Lock" part of the HTLC. With the Time-Lock feature, the sender can enforce the receiver to reveal the secret within a certain amount of time. Otherwise, the transaction fails and the sender can claim his/her money back. The process is depicted in Fig. 2. The state of the channel between the peers is kept current by mutual signatures. The final claim of the remaining funds are done on the on-chain Bitcoin network. In HTLC mechanism, whenever a peer claims funds with fabricated channel usage or claims funds with a more previous channel state there will be enough time for the other peer to open a dispute on the blockchain. If the complainant brings evidences forward to prove her case all of the channel capacity will be funded to the complainant. This mechanism is offered to prevent double spending or forging of the channels as a disincentive. The discovery of intermediate nodes between Alice and Bob is accomplished via running *node and channel discovery mechanism* within LN. This basically allows nodes to obtain a local view of the network's topology so that Alice can discover a route to Bob.

3.2. Suitability of LN for real-time payments

As LN is geared for micropayments, the urgency of payments could sometime be important to be used in real life applications such as buying a cup of coffee. Therefore, before moving to the problem definition we instantiated real LN nodes on the test network to measure the round trip time (RTT) for a payment. We measured the RTT in seconds for various number of hops and showed the results in Table 1. 2-hop payments are done to *htlc.me*, 3-hop payments are done to *starblocks.acinq.co*. Entrance node is connected to the LN via Tor network so that it introduces additional delays to the communication. The length of an LN payment packet is 1366 Bytes and the permissible number of hops is 20 [23]. The connection between the nodes can be of any type, i.e., wired, wireless, optical or combination of any of those. The RTT measurements suggest that even though the number of hops grows, the delay only increases slightly in seconds and thus LN is a viable network architecture proposal for instant cryptocurrency payments. We would like to also note that a longer RTT does not constitute to double spending problem as it does in the Bitcoin network. This is because the transactions are not immediately written to the blockchain but the state of the channel is always tracked by the peers, whenever one peer tries to forge the channel the other can open a dispute on Bitcoin network and get all of the money in the channel with a valid proof which is a disincentive against forging.

Table 1
RTT measurements for various number of hops.

Number of Hops	Min (s)	Avg (s)	Max (s)
2	2.3	2.7	4.3
3	2.3	2.9	5.4

3.3. Privacy in Lightning Network

LN, as in Bitcoin network, operates with the pseudonym addresses. Although the addresses are pseudonyms, there are still privacy concerns and solutions are being introduced. First of all, when the nodes join LN, with a public advertisement, they advertise some properties of the channels they establish. The IP address is one of those properties. The nodes need to know the IP addresses of other nodes in order to route the payments. The LN community offers using Tor Network in order to further anonymize the identities. The second one is the channel capacity. For a successful routing, the sender needs to know the capacities of the channels that the payment will be routed through so that the transaction can be successful. However, in LN in order to keep the directional capacities of the nodes secret, the channel capacity of a channel is expressed as sum of all fundings provided by the peers.

Third is the possible privacy leakage in the transactions, namely, analyzing the source node, destination node, and the amount being transferred. In LN, source-routing is utilized which makes a payer node to decide on the route of transaction. Source-routing idea comes with additional extensions where onion based routing is utilized in addition to a mix-net approach inspired from `Sphinx` protocol. With these properties, in a multi-hop payment, an intermediary node only knows the predecessor and the successor nodes.

3.4. Problem motivation and definition

Even though the concept of LN is very attractive to introduce Bitcoin in micropayment-market, its current structure requires the deployment of relay nodes which will act as bridges between customers and retailers. This brings two major issues: First, these relay nodes charge forwarding fees and thus the goal of minimizing transaction fees through the payment network will be violated. Second, eventually, these relay nodes will become major hubs in the network creating the risk of having DDoS attacks against these nodes to stop the payments in the network at any time. The third risk here is regarding customer privacy. If these 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.

These risks may deter any business to adapt the current LN for its payments. In this paper, we argue that a planned approach for creating a network topology from scratch is needed for forming a **private payment network** because of the following observations/issues on the current LN topology: suggests that this simple payment scheme, shown in Fig. 2, would eventually lead to an entire payment network when customers and retailers start establishing payment channels randomly as time passes. This would allow all the payments to be made within LN. The main argument of this paper is that this assumption would not be realistic for a significant network size without a properly planned approach due to the following challenges:

- **Network connectivity:** LN had a basic assumption that a payment network can be formed by mostly random connections without a topology plan. This assumption is not valid since there will be a certain probability of connectivity success which means that the final payment network may not be connected or some of the nodes can be loosely connected to the network. As a result, payment options for a customer will be limited. To solve this issue the developers of `lnd` (Lightning Network Daemon) introduced the optional '`Autopilot`' feature to help a node to initialize connections to other nodes in the network. `Autopilot` uses constrained Barabasi-Albert (BA) method which is being used to model connections in social networks [24]. However, we believe this approach, which relies on "network influencers" (i.e., hubs).
- **Investment need for each channel:** Forming a connected network will not be free. Each channel establishment results in two mandatory on-chain transactions. Hence, the number of channels established by a node should be kept optimum, namely, high enough to keep the needed transactions flowing but low enough to decrease the on-chain 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, most of the times 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 planned transaction volume when a consortium of business come together to form a private payment network.
- **Diminishing channel capacity over time:** An important and unique property of LN is that the capacity of channels diminishes over time. At the initial creation of a channel, it will be set to have a maximum forwarding capacity. When nodes use this channel for payment forwarding, its capacity degrades each time. For resolving this issue, either more investment should be done for the channel or there should be a reverse payment forwarding to balance the capacity. This feature needs to be incorporated into the design of a new topology.

We propose to utilize the idea of creating payment channels among the retailers of a consortium assuming that every node in this network will create in-advance payment channels (i.e., links with certain transaction capacities) with some other nodes as shown in Fig. 3. The network of the participants is an overlay network on top of the Bitcoin network where rather than the physical proximity of the nodes, the existence of channel establishment agreements between the nodes is of importance. However, due to the pointed challenges, forming such a network requires careful design to both establish fairness between cooperating nodes to share the associated costs and achieve asserted scalability, micropayment, and instant confirmation features.

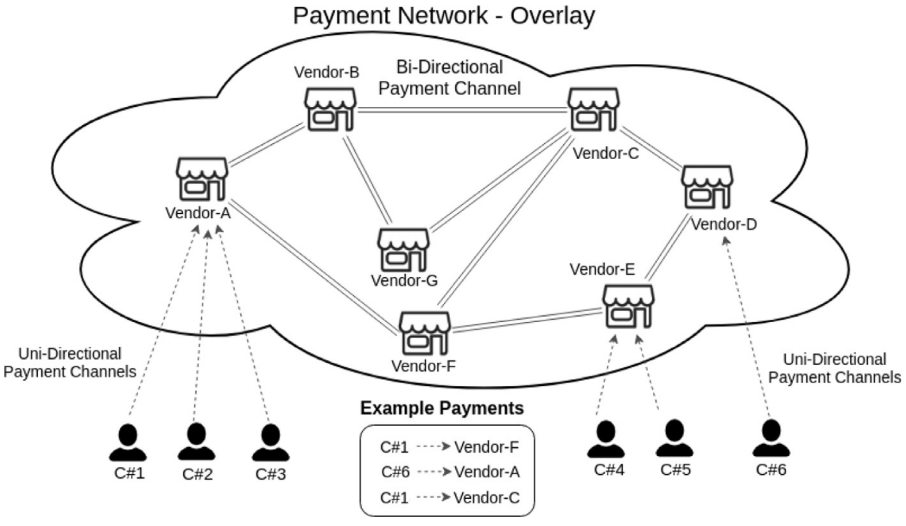


Fig. 3. An overview of the envisioned Payment Network among retailers.

Based on these discussions, our problem can be formally defined as follows: "*Let us assume M retailers and N customers. Let us also assume that a payment channel network (PCN) can be represented as a graph* $G = (V, E)$, *where V represents all Bitcoin accounts (of M retailers) and E represents all payment channels among M retailers. Every edge between retailers has a capacity that denotes the amount of depositable Bitcoins (i.e., the actual investment put in the escrow account by a retailer). 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. Based on these inputs, how can we create a virtual topology PCN among the retailers in such a way that 1) the average transaction fee for a customer will be minimized; 2) the total investment made by a retailer for creating channels with its neighbors will be minimized; 3) the standard deviation of total investment costs among the retailers will be minimized.*"

4. Flow and network optimization model

4.1. Overview

The proposed PCN in this paper will form a virtual topology on top of the on-chain operations, thus providing a valuable infrastructure which is able to guarantee P2P payment service without requiring any on-chain transactions. While designing the network, the payment channels between retailers should have enough capacity for both routing the payments of others and for receiving the payments intended for themselves. Moreover, the channel capacity between the retailers should be created in such a way that retailers with similar intent and opportunity (i.e., in terms of business capacity) should contribute to the network in a similar way.

Our approach considers all these issues by proposing a mixed integer programming model inspired by multi-commodity network flow problems [11]. The model minimizes the PCN design cost according to different cost-sharing scenarios. These costs are the network flow cost, link establishment cost, and unfairness in channel capacity distribution among the retailers.

4.2. Formulation of the network optimization model

Let us assume that there are N users in the network with I defining the set of the customers where $I = \{Cu_1, Cu_2, \ldots, Cu_N\}$ and $N = |I|$. Similarly, let us assume that there are M retailers (denoted as "RT") in the payment system and P is the set of the retailers and $P = \{RT_1, RT_2, \ldots, RT_M\}$ and $M = |P|$. From now on, the term "node" and "retailer" will be used interchangeably, and definition of each term is given in Table 2. Each customer will be registered to a retailer. The payment of a customer will be initiated from the registered retailer to the payee retailer. Since there will be a money traversal the problem can be formulated with graph representation where vertices are the retailers and edges are the links between retailers. Let the graph $G(V, E)$ be defined such that $V = I \cup P$ and $E = \{[(i, j) : i \in I, j \in P] \cup [(j, j') : j, j' \in P]\}$. Each customer, Cu_i, is assumed to be making payments to a set of retailers, J_i, during the payment period. This time period can be of any length. Hence, $|J_i|$ is the expected total number of payments that will be made by Cu_i. Let a_{ij} be the expected payment amount by $Cu_i \in I$ to the retailer $j \in J_i$ during the planning period.

Let decision variable $u_{jj'}$ denote the capacity of the payment channel (i.e., Bitcoins put on the escrow account) on edge (j, j') to be established between $j, j' \in P$, where each unit of capacity incurs a variable cost of $c^v_{jj'}$ (i.e., the opportunity cost for keeping Bitcoins in reserve since this money will stay there untouched and thus there will be no way to use it in other profitable investments). In our model, to represent a real-time scenario, we assume that setting up a payment channel will carry a fixed "channel establishment fee (i.e., on-chain transaction fee)". So, a fixed cost of $c^f_{jj'}$ is assumed if a payment channel is established between $j, j' \in P$.

Suppose that the optimization objective involves the minimization of a function of the total cost of establishing payment channels across the entire network ensuring that all payments by customers are processed following some path on the network where the termination node is the recipient of a given payment.

To allow for a multi-commodity flow type integer programming formulation, which is known to be NP-Complete [25], we further define the following decision variables. We let x^k_{ij} define the payment flow on arc (i, j) for $i \in I, j \in P$ intended for retailer $k \in J_i$. Moreover, $y^{ik}_{jj'}$ refers to the flow on arc (j, j') for $j, j' \in P$ which has originated from customer $i \in I$ with destination $k \in J_i$. In order to indicate the channel 'opening' decision between any two retailers, we define the binary variable $z_{jj'}$ such that $z_{jj'} = 1$, if there is positive flow on arc (j, j') for $j, j' \in P$, and $z_{jj'} = 0$, otherwise.

We then can formulate the optimization problem through Eqs. (1)–(10) as follows:

$$z_{jj'} = \begin{cases} 1, & u_{jj'} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\min \sum_{j \in P} \sum_{j' \in P} c^v_{jj'} u_{jj'} + c^f_{jj'} z_{jj'} \tag{2}$$

$$\text{s.t.} \sum_{j \in P} x^k_{ij} = a_{ik} \qquad \forall i \in I, k \in J_i \tag{3}$$

and Eq. (1) can be modeled as:

$$u_{jj'} \leq \epsilon z_{jj'} \tag{4}$$

$$u_{jj'} \geq \delta z_{jj'} \tag{5}$$

Table 2
Notations and their explanations.

Symbol	Meaning
$u_{jj'}$	Channel capacity from node j to j'
$z_{jj'}$	Binary decision variable, equals 1 if $u_{jj'} > 0$, equals 0 else
$c^v_{jj'}$	Cost multiplier for capacity from node j to j'
$c^f_{jj'}$	Cost multiplier for channel establishment from node j to j'
γ	Parameter to control unfairness between the nodes
x^k_{ij}	Flow originated from customer i, flowing from j destined to k
$y^{ik}_{jj'}$	Flow originated from customer i destined to node k, flowing from j to j'
a_{ik}	Total amount of intended payment originating from customer i destined to node k
$totFlows_j$	Sum of all flows originating from node j
Γ	Total cost of unfairness among nodes
$IUBpN$	Investment upper bound per node

where ϵ is the upper bound for the decision variable $u_{jj'}$ and δ is a positive small number close to the lower bound (e.g. 0.001). That constraints will force the solver to find a suitable z either 0 or 1 as z is defined to be a binary number which takes the possible solutions for z from linear range to integer range. Eq. (2) is the objective function and Eq. (3) states that for Cu_i sum of all money supplied to the network through any node for RT_k equals to a_{ik}. It basically means that a customer can register to more than one retailer.

We need to have the node transaction conservation equations as:

$$\sum_{i\in I}\sum_{k\in J_i}\sum_{j'\in P} y_{j'j}^{ik} + \sum_{i\in I}\sum_{k\in J_i} x_{ij}^{k} - \sum_{i\in I}\sum_{k\in J_i}\sum_{j'\in P} y_{jj'}^{ik} = d_j \qquad \forall j \quad \in P \tag{6}$$

where d_j defines the total '*demand*' of retailer j as $d_j = \sum_{i\in I} a_{ij}$ for all $j \in P$. Eq. (4) states that for a node j, i.e. RT_j, sum of all of the flows coming to RT_j minus sum of all of the flows leaving RT_j must be equal to the demand of RT_j.

In addition, the capacity of the links should be large enough to accommodate the flows on the arcs and the fixed cost structure should be defined. As the established channel can be bidirectional, we further assume that "channel capacity" from one node to the other is symmetric. Note that the amount that will be put into the escrow accounts can be decided by the peers at the time of agreement:

$$\sum_{i\in I}\sum_{k\in J_i} y_{jj'}^{ik} \le u_{jj'} \qquad \forall j, j' \in P \tag{7}$$

$$\sum_{i\in I}\sum_{k\in J_i} y_{jj'}^{ik} \le C' z_{jj'} \qquad \forall j, j' \in P \tag{8}$$

$$u(j, j') = u(j', j) \qquad \forall j, j' \in P$$
$$x, y, u \in R^{+}, z \in \{0, 1\} \tag{9}$$

where C' is some upper bound for the capacity on a given channel. This C' can also be chosen different for each possible arc. For this study C' is set to a constant high value for every edge.

Finally, in order to assign a customer to a single predefined retailer, we include the following equation:

$$x_{ij}^{k} = a_{ik} \qquad \forall i \in I, k \in J_i \text{ and } j = RT_{ij}^{k} \tag{10}$$

For that equation to hold RT_{ij}^{k} is a retailer chosen randomly for delivering a payment from customer i to retailer k starting at retailer j.

4.3. Fair distribution of network design costs among nodes

A member retailer of the PCN should open payment channels with other peers more than its own demand so that it can relay the payments for others. However, these payment channels have an associated cost due to keeping Bitcoin in escrow and related transaction fees. Therefore, the members of PCN strive to keep their investment costs at a minimum while participating in the formation of a PCN.

Thus, in the optimization, a new cost, namely, *unfairness cost* represented by Γ, is introduced in order to provide a mechanism for sharing network formation costs fairly. For that purpose, capacity difference between nodes with maximum and minimum outbound flows are multiplied by the parameter γ as follows:

$$totFlows_j = \sum_{i\in I}\sum_{k\in J_i}\sum_{j'\in P} y_{jj'}^{ik} \qquad \forall j \in P \tag{11}$$

$$maxFlow \ge totFlows_j, \qquad \forall j \in P \tag{12}$$

$$minFlow \le totFlows_j, \qquad \forall j \in P \tag{13}$$

$$\Gamma = \gamma\,(maxFlow - minFlow) \tag{14}$$

Note that *maxFlow*, *minFlow*, and $totFlows_j$ are non-negative decision variables. Given the minimization objective, the optimization should ensure that the *maxFlow* is set equal to the highest of $totFlows_j$ values, and *minFlow* is set equal to the lowest of $totFlows_j$ values.

The final objective function becomes

$$\min\Big(\sum_{j\in P}\sum_{j'\in P} c_{jj'}^{v} u_{jj'} + c_{jj'}^{f} z_{jj'} + \gamma\,(maxFlow - minFlow)\Big) \tag{15}$$

Throughout the rest of the paper, the model described in Sections 4.2 and 4.3, combined, will be referred to as mixed-integer optimization (MIOP) model.

5. Pruning-based heuristic

As the presented MIOP model falls under the category of NP-hard problems in general, the computational complexity would be high and thus the model will not scale. Therefore, in this section, we present a heuristic approach to improve the scalability of the MIOP model.

5.1. Simplifying the MIOP model

In our heuristic approach, MIOP model is modified in such a way that the solver will not decide on the links to be established and the unfairness among nodes. Specifically, $z_{jj'}$ and Γ variables are removed from the model. Hence, the model turns into linear programming (LP) one. Throughout the rest of the paper, this modified model will be called the *LP model*. LP model basically only optimizes the flows in the network.

Furthermore, in our heuristic, the standard deviation among flows (i.e., unfairness) is controlled by assigning an investment upper bound per node (IUBpN) for all the flows passing through a channel as shown in Eq. (17). This approach not only removes the computational overhead in the model but also decreases the possible solution domain of the model and intrinsically helps in shortening the computational time.

$$totFlows_j = \sum_{j'\in P} u_{jj'} \qquad \forall j \in P \tag{16}$$

$$totFlows_j \le IUBpN \qquad \forall j \in P \tag{17}$$

5.2. Algorithm for building the topology

In order to keep the problem in the LP domain (with a polynomial time algorithm [26]), we build the network manually and let the LP model calculate the flows in the network in polynomial time. Note that for network flow calculations there are many other alternative algorithms like Ford-Fulkerson [27] or Dinic's algorithms [28]. However, in these greedy algorithms, the result at each step varies based on the nodes selected initially. In other words, the algorithm becomes order oriented and thus finding the best order becomes a challenge. To prevent this challenge, we opted to stick with the LP model to get rid of the order dependency.

Once the flows are determined by the LP, we start pruning certain edges of the initial network topology that carry the created flows in an iterative fashion by applying the following procedure:

The algorithm, pseudo-code of which is shown in Algorithm 1, starts with a fully connected network topology. Customer registration information, payment plan, and edge connections are fed to the solver. The solver calculates the optimal transaction flow by considering the availability of the connections between nodes and upper bounds for the decision variables. When the flows are obtained, the edge capacities and edge counts of the nodes are temporarily saved. In order to start pruning, we create a table where

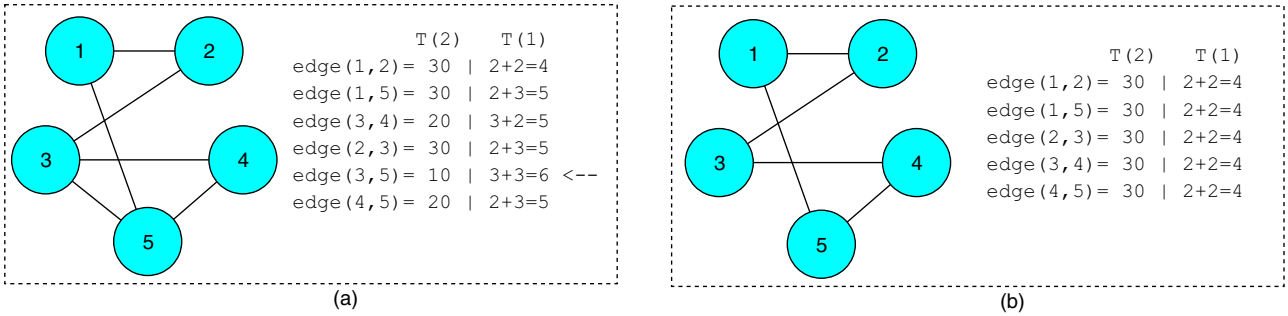


Fig. 4. A scene from the flow of the pruning heuristic. a) Initial table entries before pruning. b) Revised table entries and edges after pruning.

Algorithm 1 Network Establishment.

1: Input: *G(V,E)*=Fully connected undirected graph, *V* representing set of nodes, *E* representing set of edges, $E_{jj'}$ representing edge between j and j'
2: Input: Anticipated payment plan
3: Using LP create flow model ***m*** using ***Edges*** and payment assumptions
4: Create a 2-dimensional Table: $T[][]$
5: Solve ***m***
6: **while m** is feasible **do**
7: **for** $E_{jj'}$ in E **do**
8: $T[E_{jj'}][1]$ = total number of edges j and j' have
9: $T[E_{jj'}][2]$ = capacity of $E_{jj'}$
10: **end for**
11: Sort T in descending order w.r.t. $T[*][2]$
12: Sort T in ascending order w.r.t. $T[*][1]$
13: Remove the last edge in T
14: Update ***m***
15: Solve ***m***
16: Reset T
17: **end while**
18: Output: G

every edge in the final flow diagram is tracked. Let us assume we track an arbitrary edge (j, j'). The first column of the table would represent the total number of connections that nodes j and j' has. The second column of the table represents the established capacity on edge (j, j'), i.e., $u_{jj'}$. This is shown in an example in Fig. 4. Fig. 4a is the resulting representation of a 5-node network after 5th iteration. $T[1]$ is the sum of connections of the peers of the corresponding edges whereas $T[2]$ represents capacities of the edges. The edge which has the highest in $T[1]$ and lowest in $T[2]$ is pruned. For instance, the 5th row in Fig. 4 shows that edge(3,5) has an edge capacity of 10, node 3 has 3 connections and node 5 has 3 connections. Hence, peers of edge(3,5) have 6 connections in total. Thus, the edge(3,5) will be pruned. If there was another edge with the same total edge count, our algorithm would pick the edge with the minimum capacity. The idea behind aiming the edge with the lower capacity is avoiding unnecessary on-chain transactions for a channel that will not be used frequently as establishing a channel between two nodes will incur on-chain transaction fees. It is not desirable to have an edge for a small capacity, hence choosing the edge with the minimum capacity is important. And similarly, the nodes with more edges would have the potential to become hubs and thus create unfairness. Therefore, some of their edges are also pruned. Additionally, observing the total number of connections of the nodes ensures that we do not end with a disconnected network. Specifically, if a node has lower number of connections, it is highly probable that its edges will not be pruned. Fig. 4b shows the resulting representation after the 6th iteration.

This procedure is an iterative and greedy process. When an iteration ends, the network graph is updated with the pruned edge and procedure restarts, until topology becomes infeasible for the solver. We note that there are two reasons for a network to become infeasible, one reason is that the parameters are selected in such a way that it is impossible for the solver to come up with a flow solution. Another reason is that the network becomes disconnected after pruning so that flows cannot be forwarded to the intended destination.

5.3. Incorporation of privacy in the heuristic

In order to increase the privacy on the flow of transactions in the network, the best practice is forcing the transactions to make multiple hops. For doing so, we can introduce binary decision variables which will be set to one if there is a positive $y^{ik}_{jj'}$ flow on an edge. So, putting a constraint on these binary decision variables, such that sum of them will be greater than some number will successfully force the flow to make hops. However, introducing binary decision variables will push the flow optimization model from linear polynomial time domain to NP-complete domain as explained previously.

To avoid this situation, we opted for another approach for satisfying privacy by introducing Eq. (18) as a constraint in the flow optimization model:

$$\sum_{j\in P}\sum_{j'\in P} y^{ik}_{jj'} \geq Y^{ik} \qquad \forall i, k \in J_i \tag{18}$$

where Y^{ik} is a lower bound for satisfying privacy. Eq. (18) states that, for a particular source and destination, the sum of all of the flows carried out in the network particularly for these peers should be larger than a lower bound. For example, if Y^{ik} is set to be 3 times of a_{ik} the flow optimization solver will need to find additional routes in order to satisfy this constraint.

6. Experimental evaluation

This section presents the experimental details in terms of setup, assumptions, metrics, benchmarks and performance analysis.

6.1. Experiment setup and assumptions

The LP and MIOP optimizations were solved using Gurobi solver version 7.2 with Intel Xeon E5-2630 v4 @ 2.20 GHz CPU and 64GB of RAM PC running Ubuntu Linux Kernel version 4.15.0.

We make the following assumptions: for the optimization implementation, the available number of nodes may vary and it is explained in every figure and is denoted as N. For every node, there

is one customer registered to it. Single customer per node is chosen in order to decrease the number of equations in flow optimization. Every customer sends transactions to every other node. This assumption is made in order to not end with a disconnected graph after pruning operations. Each transaction from each customer is 10 units. γ and $c^f_{jj'}(linkcost)$ parameters should be thought as adjustment parameters, they don't directly imply monetary values. $c^v_{jj'}$ in Eq. 3 is assumed to be 1.

6.2. Performance metrics and benchmarks

The evaluation of the performance of the heuristic approach is done according to the metrics below:

- *Total Investment Cost for the Network*: This metric is the amount of total investment put by the nodes (in escrow accounts) to create a network composed of nodes establishing off-chain payment channels.
- *Total Number of Edges in the Network*: This metric indicates the total number of channels (i.e., edges or links) created in the network.
- *Standard Deviation among the Nodes' Actual Investment Costs*: This metric is to assess the fairness among the nodes in the resultant network. It measures the standard deviation among the investment costs for each node in the network.
- *Computational Time*: This metric measures the computational time elapsed for finding out the resultant payment network.
- *Betweenness Centrality of the network*: Betweenness centrality is one of the measures that show how the nodes are dominant in a network. It simply defines how many times in total a node is visited while traversing from one node to other with the shortest path. This metric gives hints about a node in a network if it becomes a hub.
- *Percentage of Cut Nodes in the network*: Cut nodes set is the set of nodes that break the connectivity of the network when they are removed.

Using those metrics, we compare the performance of these approaches:

- **MIOP approach**: This solution is based on the MIOP formulation discussed previously. The initial topology network assumed to be a fully connected one.
- **Heuristic approach**: This approach involves the pruning of the edges. It is an iterative approach, and the flows are calculated by the LP model, and edges are pruned until an infeasible setup is reached. Note that for our heuristic there are two versions: one without privacy guarantees (shown as "Our Heuristic" in the figures) and one with the *privacy guarantees* (shown as "Privacy" in the figures). For privacy guarantee, we enforced all transactions make at least 3-hops. For this heuristic, we considered various initial topologies as follows:
  - *Fully connected Topology*: The heuristic starts with a fully connected topology where each node has a link to every other node in the network.
  - *Peer-to-Peer (P2P) Torus Topology*: The heuristic starts with a purely P2P topology that creates an equal number of edges among the nodes by following the idea of a torus. Torus topology is one of the most popular topologies in parallel computing [29] so we wanted to adapt it in our study. Fig. 5 shows an example 16 node torus topology.
  - *Barabasi-Albert (BA) Model Topology*: Currently, a restricted version of the Barabasi-Albert model is utilized in LN under `Autopilot` tool. In our tests, the same heuristic is applied to a network with an initial topology created by the Barabasi-Albert model.

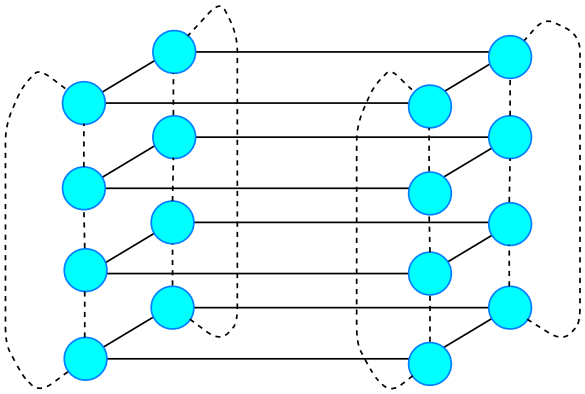


Fig. 5. 16 node torus topology.

  - *P2P Hypercube Topology*: Hypercube topology is a multidimensional network which has 2 nodes in each dimension. In every 2 dimension squares are generated. Due to this nature number of nodes in a hypercube is 2^m, m being an integer.
  - *P2P Star Topology*: In star topology all of the nodes are connected to a central node. This type of topology is very efficient in low latency multi-hop communication.
- **Random topology**: In this approach, we pick a random topology which has a number of edges comparable to the results of the heuristic that starts with the fully connected network. Only the flows are calculated on this random topology without applying pruning.

6.3. Experiment results

6.3.1. Comparison of heuristic with MIOP model solution - Experiment 1

In this experiment series, we compared the performance of our heuristic starting with an initial fully connected network with the performance of the MIOP model solution [30].

We utilized the same payment schedule used for the MIOP model solution and for the heuristic experiments. We created 10 retailers (nodes), a fully-connected network. We varied the *IUBpN* for our heuristic experiments and varied γ and $c^f_{jj'}$ (shown as *linkcost* in the figures) for the MIOP experiments. Each retailer has 48 registered customers, so in total there are 480 customers. For the payment scenario, we assume that each customer initiates a single transaction to a retailer other than its registerer retailer which is selected randomly with uniform distribution. Each transaction has a monetary value of 10 units. Note that, in this setup, every node -via customers- supplies an equal amount of money to the network but demands of the nodes are not guaranteed to be equal. Hence, total supply to the network by all of the customers is of 4800 units. MIOP finds the optimum network topology based on this scenario and it is being controlled by the γ and *linkcost* parameters.

Our heuristic algorithm stops execution when the LP model becomes infeasible after iterative edge pruning. The LP model can become infeasible mainly for 2 reasons. The first reason for infeasibility is that the connectivity in the network can be broken because of excessive pruning. The second reason for the infeasibility is that the IUBpN can be tight that LP can not distribute the flow with the obtained connectivity of the network, hence a solution does not exist in that case.

The results are given in Fig. 6a–d for total number of edges, standard deviation among the nodes' actual investments, total actual investment of the nodes in the network and total computational time to run the experiments respectively. In the

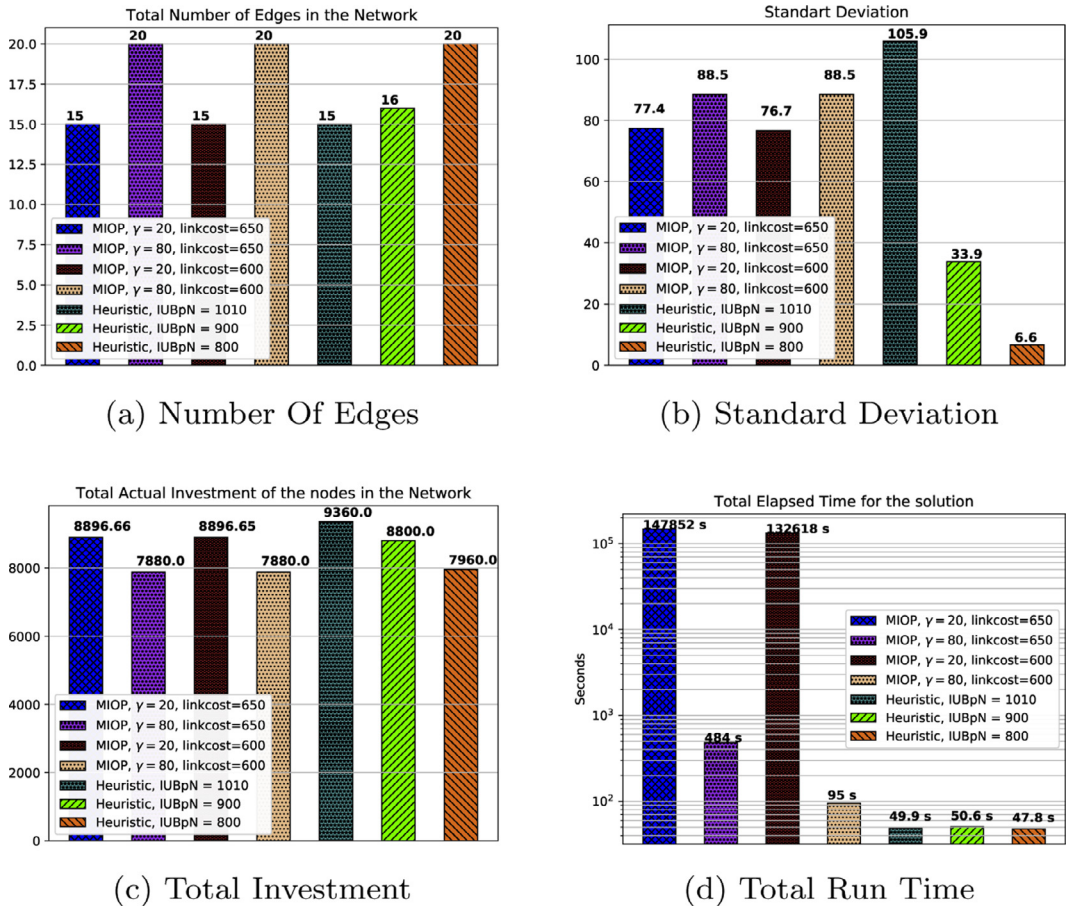


Fig. 6. Heuristic approach compared to MIOP solution.

experiments, we tried to adjust IUBpN in a way that the results would be comparable in terms of number of edges.

It is seen that the MIOP model gives only slightly better results in terms of total investment. However, this comes with a drawback of huge computational time requirement. Because of the previously mentioned complexity of the MIOP model, the required running time increases significantly.

There are several other observations: First of all, the heuristic produces less number of edges compared to the MIOP solution when IUBpN increases. Because with a lower IUBpN the heuristic terminates earlier, so that, it leaves a network with a higher number of connections. Hence, the heuristic can be adjusted for savings from channel creations.

Standard deviation results are interesting since we observe the high impact of increasing the IUBpN. However, when the right IUBpN is selected, such as 800 in our case, the flows become much fairly distributed among the nodes and surpasses the MIOP (optimal) solution. This behavior makes sense because after IUBpN is made tighter, the node capacities are kept the highest possible to fulfill the constraint, yielding a close outbound flow among nodes. Similarly, when we look at the total actual investment cost in the network, we see that the heuristic almost matches that of the MIOP model. For instance, compared to the $\gamma = 20$ case, there is only about 3% cost increase and yet the number of edges is equal to the total number of edges for the same case which is matching the heuristic. This means the heuristic can get very comparable results in terms of standard deviation and total actual investment costs while achieving scalability. Also, the IUBpN parameter works efficiently to have control over the standard deviation among the nodes.

6.3.2. Comparison of full connected topology with others: Torus topology, Barabasi-Albert Model topology, and Random topology - Experiment 2

In these experiments, our goal is to show the scalability properties of our heuristic by trying it on a P2P Torus topology, on a topology created by Barabasi-Albert (BA) method and also compare the results with a randomly connected network. To this end, we created topologies of sizes 25, 30, and 35 nodes. The topology created by BA model has 4 edges per node on average. We then compared the performance of our heuristic starting with a fully connected network to that of a Torus network, and to that of a BA model network. We also created a random network topology with the same number of nodes and same "final" number of edges, so there is no heuristic applied to random topology, only the flows are calculated with the same parameters.

Fig. 7a shows the resulting total number of edges in the networks. Fig. 7b shows the total computational time for the experiments. Fig. 7c shows the total actual investment of the nodes. Fig. 7d shows the results of the standard deviations among the investment costs of the nodes. Legend for the figures is tabulated in Table 3.

Our first observation after starting the experiments was that Torus and Barabasi-Albert model topologies were becoming infeasible in much earlier iterations when the same IUBpN is applied compared to the fully connected network. For instance, heuristic for a fully connected network could be applied up to an acceptable number of iterations when $IUBpN = 800$ but the others could not be solved, namely, with those parameters and flow requirements, the networks seem infeasible to the solver. Keeping that in mind, we decided to go with an increased IUBpN value for all

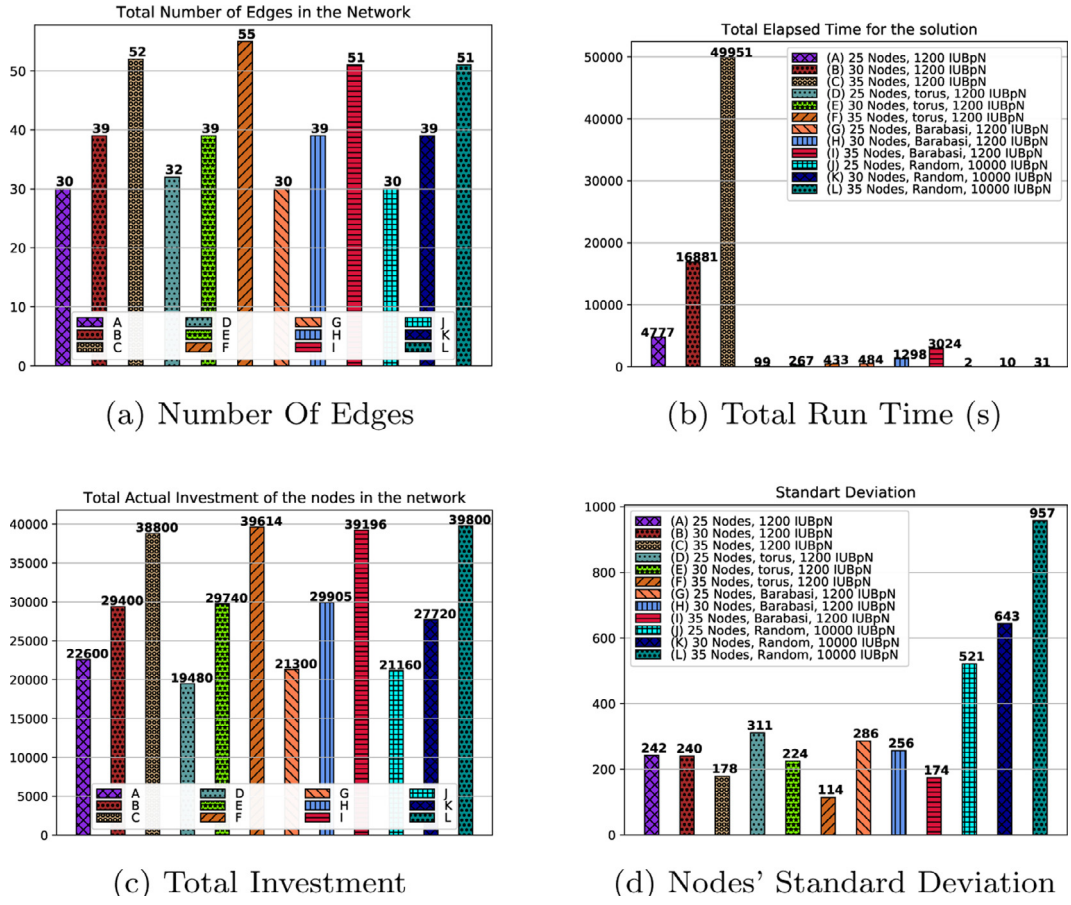


Fig. 7. Comparison of results related to heuristic for Full Connected network, Barabasi-Albert model, Torus Topologies vs. Random Topologies.

Table 3
Legend for Fig. 7.

Abbreviation	Legend
A	25 Nodes, 1200 IUBpN, full connected
B	30 Nodes, 1200 IUBpN, full connected
C	35 Nodes, 1200 IUBpN, full connected
D	25 Nodes, 1200 IUBpN, torus
E	30 Nodes, 1200 IUBpN, torus
F	35 Nodes, 1200 IUBpN, torus
G	25 Nodes, 1200 IUBpN, BA Model
H	30 Nodes, 1200 IUBpN, BA Model
I	35 Nodes, 1200 IUBpN, BA Model
J	25 Nodes, 10,000 IUBpN, Random
K	30 Nodes, 10,000 IUBpN, Random
L	35 Nodes, 10,000 IUBpN, Random

cases which is set to 1200 in the experiments. The random connected graph was the worst among all in feasibility, so, for getting a result we had to set IUBpN 10,000 for the random topology.

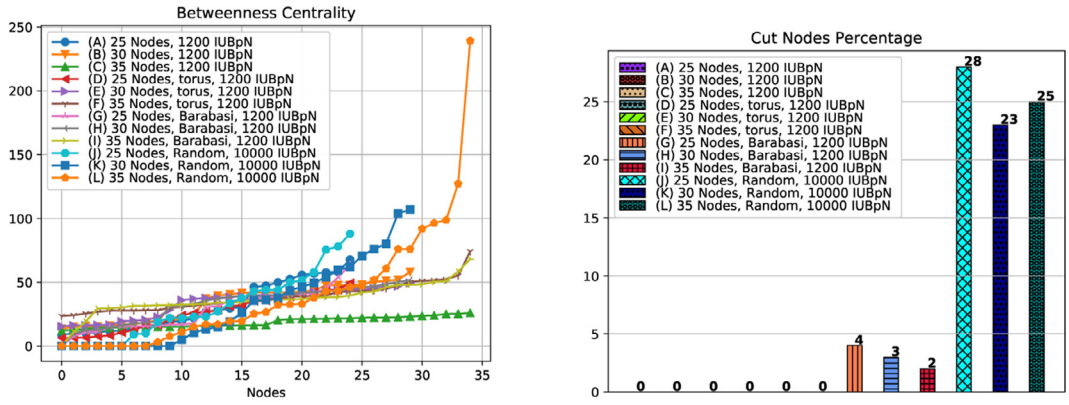
Since the heuristic steps take place in an iterative fashion, Torus and the BA model topologies have the advantage of starting from a pre-pruned state with much less number of edges. This property is well seen in total time comparisons (Fig. 7b). Additionally, with lower number of edges, the LP model deals with less number of equations which makes the equations solvable in much less time. For the random topology, as the network has less number of edges and no further pruning is done, the solver only calculates flows which is much faster compared to others. These results suggest that Torus and BA can scale much better.

An interesting observation from the results is that both BA generated topologies and Torus topologies can perform as good as a fully connected network for the total number of edges and total actual investment. For instance, for 35 nodes case, it is interesting to observe that although Torus has more edges than the fully connected topology, the total capacity of Torus network exceeds that of the fully connected one (Fig. 7c). In general, a network with a higher number of edges should have lower total capacity. This is because, if the number of edges decreases, the transactions tend to follow paths with more number of hops. High number of hops incurs additional investment requirement for other nodes, in turn, increasing the total network capacity. Because of the strict connection in Torus, the flexibility of finding shorter paths decreases resulting in higher investments.

The standard deviations of nodes' actual investments in Torus and BA are comparable to a fully connected network as seen in Fig. 7d. For the random network, however, the standard deviations are high since it has a very high IUBpN parameter. This hints that some of the nodes have higher loads, resulting in topologies more similar to a hub-and-spoke. The betweenness centralities of the networks shown in Fig. 8a support this finding as random topologies have the highest scores.

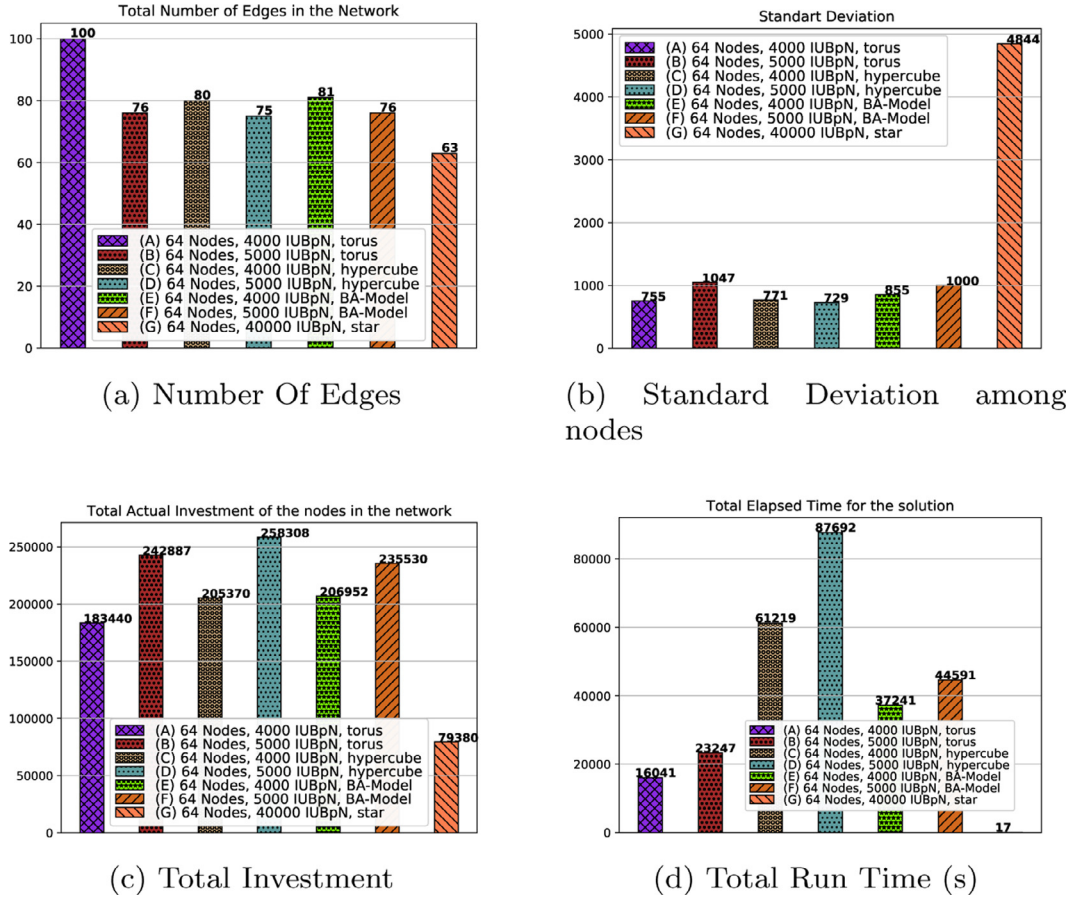
On top of statistical and computational calculations made on edges and nodes, visiting the topological properties of the networks gives us a better perspective on advantages and disadvantages of resulting topologies. Fig. 8a shows the betweenness centrality of the networks and Fig. 8b shows the percentage ratio of cut nodes (articulation points) to all of the nodes in the network.

With total capacity, total number of edges and total investment costs of the topologies being close to each other, heuristic applied to the initially fully connected network gives the best results in terms of betweenness. Percentage of cut nodes shown in Fig. 8b shows that randomly connected network is very weak against



(a) Betweenness Centrality (b) Percentage of cut nodes

Fig. 8. Betweenness centrality and Cut Node Results of full connected, Torus, BA model and random topologies after applying heuristic.



(a) Number Of Edges (b) Standard Deviation among nodes

(c) Total Investment (d) Total Run Time (s)

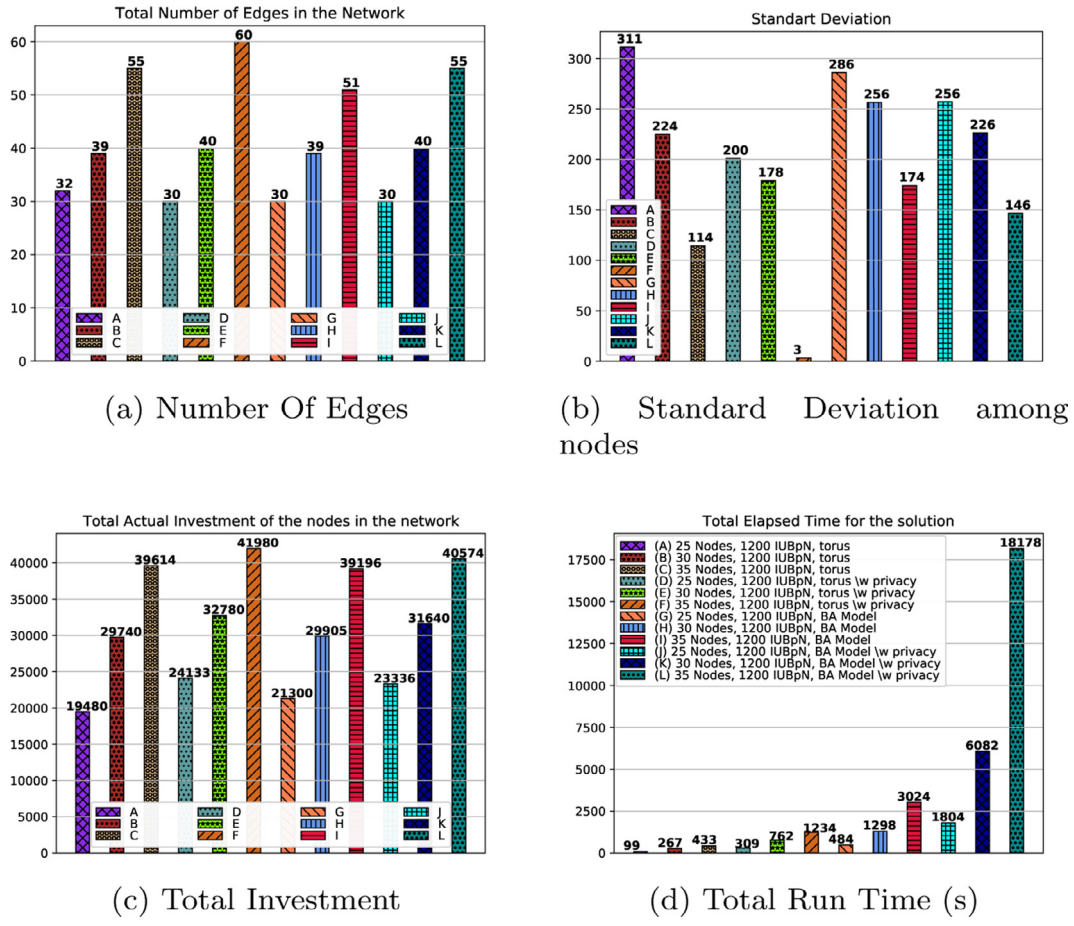
Fig. 9. Comparison of Torus, BA model, Hypercube and Star topologies.

attacks as there are many nodes to attack which will highly impact the operational success. BA model has similar vulnerabilities but significantly lower than that of the randomly connected network. Fully connected and Torus networks seem to be more robust.

Consequently, we can conclude that given the much more efficient computational time for Torus, it can be a more viable option in real-life as long as the right IUBpN is selected. Additionally, if BA model is created with a higher average number of edges, the results may come closer to the fully connected network results. Because there will be more room to fine tune the final topology of the network, but, with an increase in computation time.

6.3.3. Comparison of others: Torus, Hypercube, BA, and Star topologies - Experiment 3

In this experiment, we investigate how our approaches perform compared to other related topologies. We conducted experiments to with an initially torus connected network, hypercube connected network, BA model connected network, and star connected network topologies. As hypercube being combinations of squares in



(a) Number Of Edges

(b) Standard Deviation among nodes

(c) Total Investment

(d) Total Run Time (s)

Fig. 10. Heuristic applied to topologies of BA and torus with and without privacy.

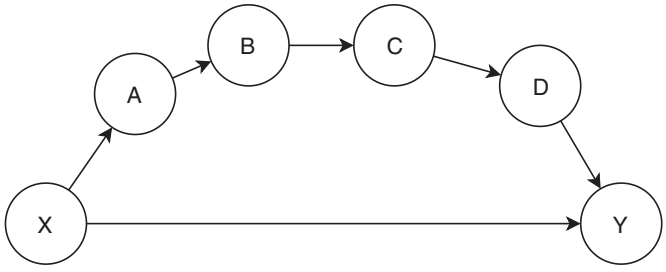


Fig. 11. Privacy-preserving flow from X to Y.

n-dimensional space, for a hypercube topology, number of nodes in the network is 2^n where n is an integer. Hence for this experiment the number of nodes in the networks is selected to be 64. Fig. 9a–d show total number of edges, standard deviation, total investment cost, and total time spent in the final topologies.

In star topology the central node has to forward all of the payment requests to corresponding nodes so *IUBpN* for central node is extremely high, 40,000. Otherwise, with a lower *IUBpN* the solver can not optimize. Torus, hypercube and BA-model topologies give comparable results in terms of total capacity and standard deviation for the same *IUBpN*s. Hypercube topology starts with a higher number of nodes compared to Torus. That causes more edges to be pruned and a higher total time for the method to finish for the same *IUBpN*. Additionally, initial higher number of edges means more equations and that makes the solver to take longer times to calculate the flows. For the torus' 4000 *IUBpN* case it can be said that because torus has a larger diameter, (8) than the hypercube (6), the average number of hops is higher. Thus, for a small *IUBpN* in torus, the solver will stop earlier because average load on the nodes increases. That explains why the solver stopped when the number of edges was 100. Although hypercube topology gives comparable results to the torus topology, for the remaining experiment we will continue with the torus network since it is much more efficient in terms of total computation time and scales better.

6.3.4. Privacy-Guaranteed Heuristic experiments

In these experiments, we looked at the impact of guaranteeing privacy by revising our heuristic and compared it with the case when privacy is not necessarily guaranteed (i.e., normal heuristic). The initial topologies for the experiments are chosen as Torus and BA model generated topologies as they provide a close performance to a fully connected network. For the privacy case, we set the constraint (i.e. Y^{ik}) in Eq. (18) as 30 units. Since a transaction flow is worth 10 units, we would like to observe 3 hops for a successful transaction in an ideal case. For this particular experiment, the number of nodes in the networks is set to be 25, 30, and 35. The IUBpN in the experiments is fixed and is 1200. As applied before, stopping criteria for the algorithm is reaching to an unfeasible flow distribution solution.

Fig. 10a–d show the resulting final number of edges, the standard deviation among the final actual investment of the nodes, final total actual investment in the network and the total computational time for the experiments, respectively where the legend for the figures are is in Table 4.

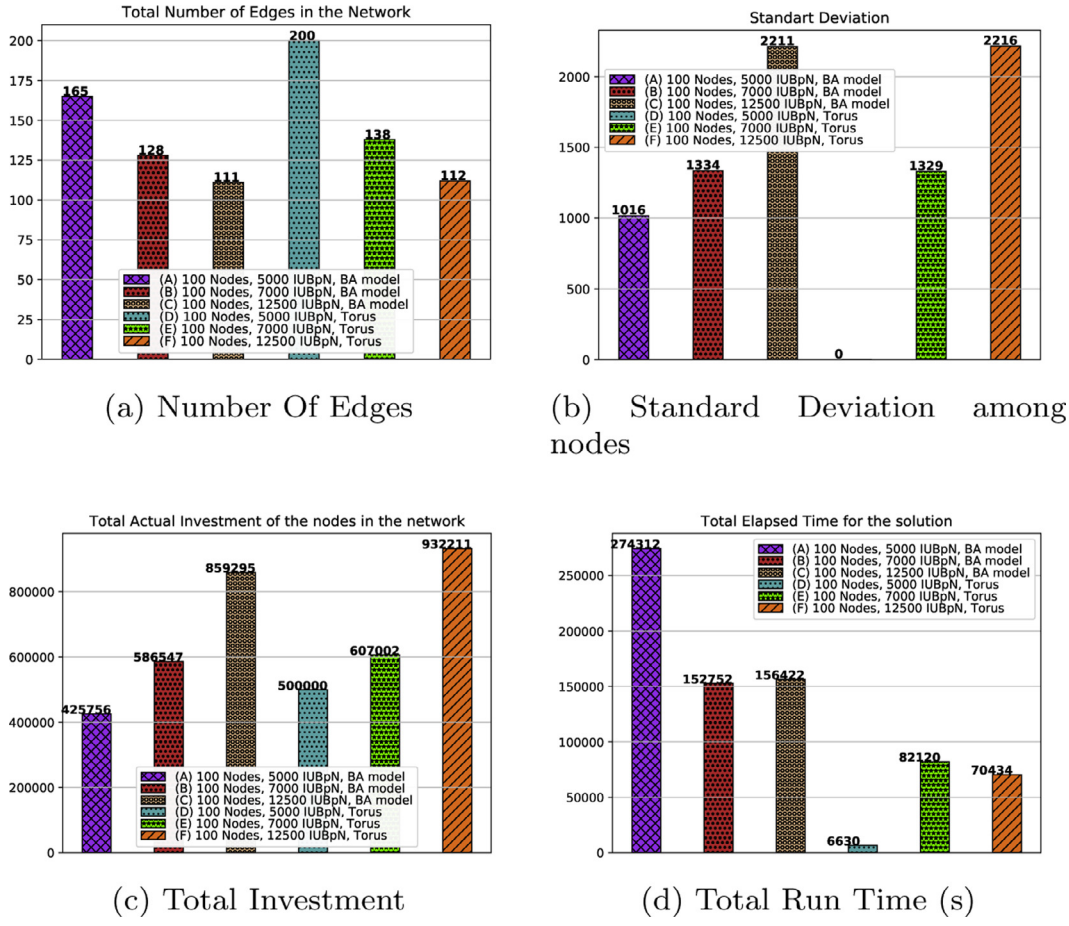


Fig. 12. Further scalability tests for Torus and BA model generated topologies.

Table 4
Legend for Fig. 10.

Abbreviation	Legend
A	25 Nodes, 1200 IUBpN, torus
B	30 Nodes, 1200 IUBpN, torus
C	35 Nodes, 1200 IUBpN, torus
D	25 Nodes, 1200 IUBpN, torus, with privacy
E	30 Nodes, 1200 IUBpN, torus, with privacy
F	35 Nodes, 1200 IUBpN, torus, with privacy
G	25 Nodes, 1200 IUBpN, BA Model
H	30 Nodes, 1200 IUBpN, BA Model
I	35 Nodes, 1200 IUBpN, BA Model
J	25 Nodes, 1200 IUBpN, BA Model, with privacy
K	30 Nodes, 1200 IUBpN, BA Model, with privacy
L	35 Nodes, 1200 IUBpN, BA Model, with privacy

One of the observations in these experiments was that when the stress on the solver is increased by increasing the limit in Eq. (18), the solver starts to create bogus payments. For example, in a case, we observed that the receiving node was making transactions to other nodes just to fulfill the requirements of the constraint. We solved that problem by modifying the model such that a node will not accept a transaction it makes. We either observe multi-hop flows or flows similar to the ones shown in Fig. 11. This forwarding scheme is acceptable as an attacker with malicious intent will not be able to learn full details about transactions.

For this experimental setup, as expected, in order to satisfy the privacy constraint the solver consumes more time, 3 to 4 folds, compared to the normal heuristic for both topologies. Another observation is that for Torus topology, the solver converges to a solution faster compared to the BA model topology, thanks to its more ordinate structure. Likewise, the tidier arrangement in Torus topology results in a relatively higher number of edges. Since the edges are connected in a strict array type, the possible solutions are consumed rapidly. That's why the standard deviation tends to decrease with given IUBpN. However, BA model topologies tend to have some degree of liberty in connections.

We can see that the total number of edges and total investment are only slightly higher in the privacy case while standard deviation among the investment of the nodes is smaller. This can be attributed to the fact that IUBpN upper limit in the experiment still gives enough room for successful distribution of the transactions. Additionally, in the case without privacy, some transactions are already satisfying the privacy requirement. So we do not observe a drastic increase in total capacity. However, as the node flows are closer to the IUBpN limit, we observe a lower standard deviation indicating a fair distribution of the flows which is good.

6.3.5. Scalability experiments

In order to further evaluate the scalability property of our heuristic, we made several experiments with networks of 100 nodes. As a benchmark, we compared the Torus and BA model networks with different IUBpN values. Fig. 12a–d depicts the total number of edges, standard deviation, total investment cost and total time in the final topologies. As discussed in the previous sections, the more ordinate connection arrangement in the Torus topology results in a higher number of edges and higher investment cost but a better control on the standard deviation among node capacities. However, in BA model topology there is more room to find shorter paths yielding in lower number of hops and

consequently lower investment cost throughout the network although the control on the standard deviation is harder. For the very same reason, Torus topologies terminate sooner than the BA model based topologies. Consequently, the computational time required in the Torus topology experiments is significantly lower than that of the BA model topologies. Considering the final number of edges, final total capacity, and comparable standard deviation control BA model gave better solutions compared to Torus. However, when computation time is important Torus topology has an undeniable advantage.

7. Conclusion

In this paper, we designed a private payment network from scratch for Bitcoin by exploiting LN technology for a marketplace where retailers and customers are available. In developing this payment network, we achieved several objectives: First, we eliminated the high transaction fees and confirmation times by using the off-chain concept of LN. Second, we ensure forming a connected payment network which is capable of transferring any payments between customers and retailers while establishing fairness between cooperating retailers to share the associated costs. Finally, we both reduced the success of DDoS attacks on the network and the possibility of privacy leaks by creating a pure P2P topology. The development included an optimization model for flow maximization while performing pruning for the edges in order to reduce the number of channels to be opened.

The evaluation with Python and Gurobi indicated that the performance of the heuristic approach is very close to the MIOP solution while allowing a certain scalability. The results also suggested that the topologies generated by Barabasi–Albert came as a favorable initial topology that can scale much faster and still provides comparable results as long as the upper bound for the channel capacities are picked rightly.

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.

CRediT authorship contribution statement

Enes Erdin: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Mumin Cebe:** Conceptualization, Writing - original draft, Writing - review & editing, Visualization. **Kemal Akkaya:** Writing - original draft, Writing - review & editing, Supervision. **Senay Solak:** Conceptualization, Writing - original draft, Writing - review & editing. **Eyuphan Bulut:** Writing - review & editing. **Selcuk Uluagac:** Writing - review & editing.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Award Number 1663051. The views expressed are those of the authors only.

References

[1] T.-T. Kuo, et al., Blockchain distributed ledger technologies for biomedical and health care applications, J. Am. Med. Inform. Assoc. 24 (6) (2017) 1211–1220.
[2] N. Hackius, 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 Commun. Mag. (2018).
[4] Bloomberg, 2017, (http://www.bloomberg.com/view/articles/2017-11-14/bitcoin-s-high-transaction-fees-show-its-limits).
[5] BitInfoCharts, 2017, (http://bitinfocharts.com/comparison/bitcoin-transactionfees.html).
[6] R. Browne, Big transaction fees are a problem for bitcoin but there could be a solution, 2017, (http://cnbc.com/2017/12/19/big-transactions-fees-are-a-problem-for-bitcoin.html).
[7] J. Poon, T. Dryja, The bitcoin lightning network: scalable off-chain instant payments, Technical Report (Draft), 2015.
[8] TrustNodes, Lightning network ddos sends 20% of nodes down, http://trustnodes.com/2018/03/21/lightning-network-ddos-sends-20-nodes
[9] DIAR, Lightning Strikes, But Select Hubs Dominate Network Funds, 2018, (http://diar.co/volume-2-issue-25).
[10] A. Haghani, S.-C. Oh, Formulation and solution of a multi-commodity, multi--modal network flow model for disaster relief operations, Transp. Res. Part A 30 (3) (1996) 231–250.
[11] S. Even, A. Itai, A. Shamir, On the complexity of time table and multi-commodity flow problems, in: 16th Annual Symposium on Foundations of Computer Science (sfcs 1975), IEEE, 1975, pp. 184–193.
[12] Bitcoin wiki, Bitcoin contract, (http://en.bitcoin.it/wiki/Contract).
[13] C. Burchert, C. Decker, R. Wattenhofer, Scalable funding of bitcoin micropayment channel networks, R. Soc. Open Sci. 5 (8) (2018).
[14] Raiden, 2018, (http://raiden.network/).
[15] S. Thomas, E. Schwartz, A protocol for interledger payments, 2015, (http://interledger.org/interledger.pdf).
[16] L.N. Team, Atomic cross-chain trading.
[17] I.A. Seres, L. Gulyás, D.A. Nagy, P. Burcsi, Topological analysis of bitcoin's lightning network, arXiv:1901.04972 (2019).
[18] S. Martinazzi, The evolution of lightning network's topology during its first year and the influence over its core values, arXiv:1902.07307 (2019).
[19] G. Karakostas, Faster approximation schemes for fractional multicommodity flow problems, ACM Trans. Algorithms 4 (1) (2008).
[20] R. Pass, et al., Micropayments for decentralized currencies, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 207–218.
[21] B. Wiki, Contracts: Example 7: Rapidly-adjusted (micro) payments to a predetermined party.
[22] B. Wiki, Hashed time-lock contracts, 2018.
[23] Lightning RFC, Bolt 4: Onion routing protocol, 2016, (http://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md).
[24] StackExchange, How does lnd's autopilot feature work?, 2018, (https://bitcoin.stackexchange.com/questions/75097/how-does-lnds-autopilot-feature-work).
[25] C.H. Papadimitriou, On the complexity of integer programming, J. ACM (JACM) 28 (4) (1981) 765–768.
[26] L.G. Khachiyan, Polynomial algorithms in linear programming, Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki 20 (1) (1980).
[27] L.R. Ford, D.R. Fulkerson, Maximal flow through a network, Can. J. Math. 8 (3) (1956) 399–404.
[28] E.A. Dinic, Algorithm for solution of a problem of maximum flow in networks with power estimation, in: Soviet Math. Doklady, 11, 1970, pp. 1277–1280.
[29] K. Asanovic, et al., The landscape of parallel computing research: a view from berkeley, Technical Report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.
[30] E. Erdin, M. Cebe, K. Akkaya, S. Solak, E. Bulut, S. Uluagac, Building a private bitcoin-based payment network among electric vehicles and charging stations, in: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2018, pp. 1609–1615.

Enes Erdin is a Ph.D. student in the Department of Electrical and Computer Engineering at Florida International University and is an NSF CyberCorps Fellow. He conducts research in the areas of hardware security, blockchain technology, and cyber-physical systems.

Mumin Cebe is a Ph.D. student in the Department of Electrical and Computer Engineering at Florida International University. He works at the Advanced Wireless and Security Lab (ADWISE). 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.

Kemal Akkaya (A'08M'08SM'15) received the Ph.D. degree in computer science from the University of Maryland at Baltimore County, Baltimore County, MD, USA, in 2005.,He is a Professor with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA. He joined the Department of Computer Science, Southern Illinois University, Carbondale, IL, USA, as an Assistant Professor, 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 has authored or co-authored over 150 papers in peer-reviewed journal and conferences. His current research interests include security and privacy, energy-aware routing, topology control, and quality of service issues in a variety of wireless networks, such as sensor networks, multimedia sensor networks, smart grid communication networks, and vehicular networks.,Dr. Akkaya was a recipient of the Top Cited Article Award from Elsevier in 2010. He is an Area Editor of the Elsevier Ad Hoc Network Journal and serves on the Editorial Board of the IEEE Communication Surveys and Tutorials. He has served as a Guest Editor for the Journal of High-Speed Networks, the Computer Communications Journal (Elsevier), and the Ad Hoc Networks Journal, and on the TPC of several leading wireless networking conferences, including IEEE ICC, Globecom, LCN, and WCNC.(Based on document published on 21 November 2018).

Senay Solak received the B.S. degree in electrical engineering from the United States Naval Academy, Annapolis, MD, in 1997 and the M.S. and Ph.D. degrees in industrial engineering from the Georgia Institute of Technology, Atlanta, in 2002 and 2007, respectively.,He is currently an Assistant Professor of operations management with the Isenberg School of Management, University of Massachusetts, Amherst, where he teaches and conducts research in the areas of optimization and simulation. His research focuses on applications of stochastic optimization in aviation, particularly in traffic-flow management and airspace capacity modeling.

Eyuphan Bulut is an Assistant Professor at Computer Science Department of Virginia Commonwealth University (VCU). Before he joined VCU, he was working for Cisco Systems as a senior engineer in Mobile Internet Technology Group (MITG). He received his Ph.D. degree in computer science department of Rensselaer Polytechnic Institute (RPI) in 2011. His recent research interests include UAV networks, mobile computing, network security and privacy, crowdsensing.

Selcuk Uluagac leads the Cyber-Physical Systems Security Lab at Florida International University, focusing on security and privacy of the Internet of Things and cyber-physical systems. He has Ph.D. and M.S. degrees from Georgia Institute of Technology, and an M.S. from Carnegie Mellon University. In 2015, he received the U.S. National Science Foundation CAREER award and the U.S. Air Force Office of Sponsored Research's Summer Faculty Fellowship, and in 2016, a Summer Faculty Fellowship from the University of Padova, Italy.