MPCN-RP: A Routing Protocol for Blockchain-Based Multi-Charge Payment Channel Networks

Yanjiao Chen, *Senior Member, IEEE*, Yuyang Ran, Jingyue Zhou, Jian Zhang, and Xueluan Gong

***Abstract*—Blockchain-based cryptocurrencies are severely limited in transaction throughput and latency due to the need to seek consensus among all peers of the network. A promising solution to this issue is payment channels, which allow unlimited numbers of atomic and trust-free payments between two peers without exhausting the resources of the blockchain. A linked payment channel network enables payments between two peers without direct channels through a series of intermediate nodes that forward and charge for the transactions. However, the charging strategies of intermediate nodes vary with different payment channel networks. Existing works do not yet have a complete routing algorithm to provide the most economical path for users in a multi-charge payment channel network. In this work, we propose MPCN-RP, a general routing protocol for payment channel networks with multiple charges. Our extensive experimental results on both simulated and real payment channel networks show that MPCN-RP significantly outperforms the baseline algorithms in terms of time and fees.**

***Index Terms*—Blockchain, payment channel network, routing.**

I. Introduction

SINCE Bitcoin [1] became the first decentralized cryptocurrency in 2009, many similar cryptocurrencies such as Ethereum [2] and Ripple [3] have sprung up. Unlike traditional currencies that rely on centralized regulatory systems, cryptocurrencies depend on a decentralized ledger that seeks unified consensus among mutually mistrusting peers in the network. However, to reach an overall consensus, the network needs to broadcast every transaction to all peers in the network, which results in huge storage and communication costs. The average size of a transaction in Bitcoin is about 500 bytes, and a turnover of 60,000 transactions per minute would require 56 GB of additional disk capacity per day [4]. After a transaction is packaged into a block, this block needs to be synchronized to all nodes around the world before packaging the next block, which requires a huge network bandwidth. Moreover, the speed of each node's processing block is also different. The above factors lead to the limited throughput of Bitcoin. Besides, the Bitcoin blockchain can only process up to 7 transactions per second (tps), which is far from people's daily needs. Thus, this has sparked worldwide concern about the throughput of cryptocurrencies.

To address the throughput limitation of decentralized cryptocurrency, many methods have been proposed [5], [6], [7], [8]. A promising approach is to move the transactions off-chain, while preserving the system's decentralized structure and strong integrity guarantees. The most widely known off-chain projects that increase the throughput of Bitcoin and Ethereum are Lightning Network [9] (Bitcoin), Raiden [10] (Ethereum), Plasma [11] (Ethereum). Payment channels are the foundation of off-chain expansion solutions, but they are not enough as a user cannot create a channel with everyone he wants to trade. To address this problem, Lightning Network introduced Hashed Time-Lock Contract (HTLC), which connects two nodes that are not directly connected through a series of end-to-end payment channels. HTLC allows the sender and the recipient to handle the payments through a series of intermediate nodes (INs) that charge fees for forwarding the money. A network containing all peer nodes and payment channels is called a Payment Channel Network (PCN) [9].

Existing commercial PCNs have various charging policies, among which time-related transaction fee is an important term. For example, the charging policy of the well-known Lightning Network includes a fee that is determined by the time value of the locked funds in payment channels [9], which is related to the number of hops between the relay node and the destination node. The time value fee is reasonable since the longer the funds are locked for future transaction, the higher the opportunity cost is for the relay node since the funds may be used for gaining interests in the current period. Time-related fees have often been considered in economics [12], [13]. Finding the path with the lowest cost is beneficial for users. There are some works on routing in payment channel networks [14], [15], [16], [17], [18]. However, most of these works only focus on security but not the cost of the path, and some miss important restrictions, such as channel capacity and time restrictions [14], [19], [20], [21]. CheaPay [22] is concerned with the cheapest routing problem with time lock restrictions but ignore the time-related fee.

In this paper, we present MPCN-RP, a general framework for routing in payment channel networks that aims to find

Manuscript received April 12, 2021; revised September 16, 2021; accepted December 24, 2021. Date of publication December 28, 2021; date of current version June 10, 2022. This work was supported in part by the NSFC under Grant 61972296, and in part by the Wuhan Applied Fundamental Research under Project 2019010701011419. The associate editor coordinating the review of this article and approving it for publication was B. Stiller. *(Corresponding authors: Jian Zhang; Xueluan Gong.)*

The authors are with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: jzhang@whu.edu.cn; xueluangong@whu.edu.cn).

Digital Object Identifier 10.1109/TNSM.2021.3139019

the path with the minimum cost for users. We consider four types of transactions fees, namely fixed fee, proportional fee, imbalance fee and time value fee, which cover most charging policies of payment channel networks. Our routing framework can be applied to various payment channel networks by adapting the transaction fee model. For example, to apply the framework to the Lightning Network, we can delete the imbalance fee and preserve fixed fees, proportional fees, and time-related fees. For Raiden [10], we can preserve the fixed fee, the proportional fee, and the imbalance fee. We design an improved directed Dijkstra algorithm to find the optimal path. In particular, we add restrictions to the path-finding algorithm to ensure sufficient channel capacity for payment.

We hereby summarize our major contributions.

- We establish a sophisticated transaction fee model covering fixed fees, proportional fees, balanced fees, and time fees, which can be adapted to a wide range of payment channel network, such as Lightning Network and Raiden.
- We design an improved directed Dijkstra algorithm to find the path with the lowest cost. To ensure that each edge can cater to the capacity requirements, we impose a capacity restriction during the path-finding process.
- We implement MPCN-RPand evaluate its performance on both simulated network and Lightning Network. The experiment results show that MPCN-RPachieved high efficiency in the overall performance of time and economic benefits.

II. Related Work

Blockchain and payment channel network: Satoshi Nakamoto first proposed Bitcoin in 2008 [1]. Afterwards, Ethereum [23] is built that allows smart contracts to express more feature-rich transactions. Other proposals, such as zcash [24] were proposed to enhance the security of transactions on Bitcoin. Bonneau *et al.* [25] provided an introduction to cryptocurrency. As the functions of the smart contract language become more abundant, it is proved that smart contracts have fatal security risks [26]. Schnorr signatures [19] have been recently proposed to be added to Bitcoin to enhance its privacy and security. Payment channels initially took shape in the Bitcoin community [6]. The Lightning Network [9] has become the most outstanding proposal of the Bitcoin payment channel network and has the largest number of nodes. Other payment channel networks such as Thunder [27] and Eclair [28] for Bitcoin and Raiden [10] for Ethereum have been proposed with only slight differences from the Lightning Network.

Routing in Payment Channel Network: In 2016, Flare [15] was proposed as one of the earliest distributed routing solutions for payment channel networks. In this distributed routing algorithm, nodes are divided into ordinary nodes and beacon nodes. Each node maintains a routing table, which stores nearby nodes and paths to multiple beacon nodes. Later, various routing algorithms for PCNs have been proposed [22], [29], [30], [31]. Zhang *et al.* [22] proposed a cheapest routing scheme for PCNs but made the simplified assumption that each node charge a fixed fee. Malavolta *et al.* [14] proposed SilentWhisper, a landmark-based routing solution [32] that addresses the privacy and security issues in routing. However, all paths must pass through landmarks, which leads to unnecessary long paths. SpeedyMurmurs [18] uses an embedded path [33] to overcome these shortcomings. But landmark routing assumes that a small group of trusted clients control the whole routing process. Rohrer *et al.* [17] regarded payment as a process and found multiple paths for payment. However, its security is hard to guarantee. Based on this idea, Yu *et al.* [16] proposed a distributed routing algorithm to improve payment success rate and reduce computational overhead. However, it ignores the important role of some constraints, such as transaction fees. Engelmann *et al.* [34] considered a simplified model in which transaction fees only depend on the payment amount, but ignore the fees of intermediate nodes. Wang *et al.* [31] used the transaction characteristics of the off-chain network to balance path optimality and detection overhead in dynamic multi-path routing. However, inferring payment size is difficult in onion-routed networks, such as Lightning. Sivaraman *et al.* [35] proposed a routing solution that packetizes multiple transactions and uses a multi-path transport protocol to achieve high throughput. However, many small payments follow the same path, which incurs a lot of overhead. Bagaria *et al.* [30] proposed a complementary method of splitting on the path, but routing higher payment values will increase locked collateral. Eckey *et al.* [36] proposed a Bitcoin-compatible protocol that allows intermediaries to split payments on the path, thereby reducing overhead, besides [36] can avoid the increase in locked collateral. However, the authors left the question of how to integrate the fee into the algorithm. Schnorr [19] proposed a public key signature scheme and corresponding authentication scheme to promote the security of PCN network. Zhang and Yang [37] modified the original HTLC protocol to make it a robust payment routing protocol.

Many of the previous works on routing schemes for PCNs mainly focus on security issues. Existing works that aim to find the lowest-cost path in PCNs adopt an over-simplified fee model. In this paper, we present MPCN-RP, a routing protocol that considers all kinds of costs specifically associated with payment relaying in PCNs. We design an efficient algorithm to find a path with the lowest cost within a short time to enable frequent transactions in PCNs.

III. System Model

A. Background

1) Payment Channel Network: Cryptocurrencies such as Bitcoin, Ethereum, and Ripple, are based on blockchain, essentially a decentralized ledger [38] that is shared, replicated, and synchronized among distrusted network members. A decentralized ledger records transactions between network participants, greatly reducing the expense incurred by mediating different ledgers. It is shown that PoW is an excellent and reliable consensus mechanism for a public blockchain like Bitcoin and is able to prevent a Sybil Attack. However, PoW requires peers to propagate and store all the transaction information, thus resulting in huge storage costs and

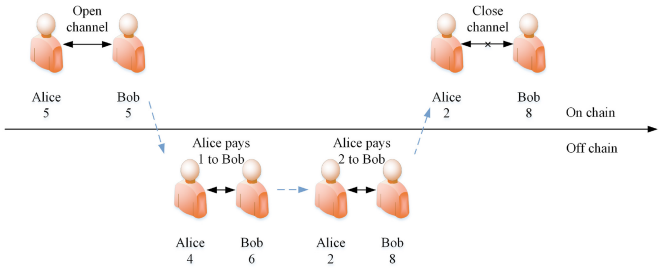

Fig. 1. An illustrative example of payment channels. Firstly, a channel is created between Alice and Bob. Next, Alice pays 1 to Bob. After that, Alice pays 2 to Bob. Finally, we close the channel between Alice and Bob.

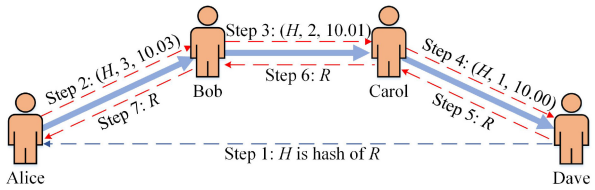

Fig. 2. An illustrative example of HTLC. Alice sends a payment of 10 to the recipient Dave via Bob and Carol (thick blue line). Assume that the transaction fees charged by Bob and Carol are 0.02 and 0.01, respectively. The HTLC (*H*, 2, 10.01) (red dashed line) Bob sent to Carol in step 3 means that Bob tells Carol that if Carol can provide the preimage of *H* within 2 days (step 6), Carol can get a payment of 10.01 from Bob.

communication overhead, which strictly limits the throughput of cryptocurrency systems. Bitcoin confirms only 7 transactions per second on average while Ethereum only confirms about 7-15 transactions per second, which is far from enough to meet the needs of frequent financial transactions.

To improve the throughput of decentralized ledgers, payment channels, an off-chain transaction paradigm, have been proposed. A payment channel allows multiple payments between two users without the need to announce each transaction to the blockchain. Two peers establish a payment channel by putting a certain amount of coins into a joint account and adding this double-signed transaction to the blockchain [39]. In the illustrative example depicted in Fig. 1, Alice and Bob open a payment channel for both parties' initial funds of 5 coins. Next, there can be multiple transactions between Alice and Bob. For example, Alice transfers money to Bob twice with the amount of 1 and 2, respectively. After some payments, a transaction can be added to the blockchain to update the channel balance agreed upon by both Alice and Bob. If the channel is no longer needed or one party's funds are exhausted, the funds are allocated to both parties according to the final state of the channel and the transaction that closes the channel is sent to the blockchain. The deposit will be returned to each user according to the most recent balances [40]. A payment channel network (PCN) is formed by a large number of payment channels among peers. In PCN, even if there is no direct channel between two peers, payment can still be made through a path formed by multiple end-to-end payment channels.

2) Hashed Time Lock Contract: There are some problems with path-based payments. For example, how to ensure that the sender's money can be recovered when the intermediate node refuses to forward the payment? Hashed Time Lock Contract (HTLC) mechanism has been introduced to guarantee path-based payments. HTLC enables payments to be made across two or more payment channels. It assumes that two parties agree on a financial arrangement in which one party will pay the other party a certain amount of cryptocurrency, e.g., Bitcoin. Since these contracts are time-locked, the receiver only has a certain time to accept the payment, otherwise the money will be returned to the sender. The hash time lock helps eliminate the need for a third party to enforce the payment. This is because the sender can create a conditional payment, and then the receiver can agree to, receive, and verify the transaction.

We present an example to explain how HTLC works, as shown in Fig. 2. Firstly, the recipient randomly generates a string *R* and sends its hash value *H* to the sender. Then, the sender and any intermediate peers put *H* in the double-signed transaction contract. Thereafter, the recipient receives payment only when the confidential *R* is provided to the sender. For each transaction, if the sender does not receive *R* within the HTLC tolerance, the transferred funds will be refunded. For an on-chain transaction, the HTLC tolerance relies on the worst on-time constraint that we express in the number of days. Each peer in the path sets a tolerance for HTLC, and this tolerance is smaller in the expenditure channel than in the inward channel. This ensures that if a peer's payment is withdrawn by the peer after it, the peer can withdraw the payment from the peer before it. Apart from the payment to the receiver, the intermediate peers charge transaction fees for forwarding payment, which, however, are significantly lower than blockchain transaction fees and are affected by many other factors.

B. Network Model

We consider a PCN $G = (V, E)$, where V represents the set of nodes and E represents the payment channels in the network. For each pair of nodes v_i and v_j, if there is a payment channel $[v_i, v_j] \in E$ connecting them, we use $b_{i,j}$ and $b_{j,i}$ to denote the balances of account $v_i \rightarrow v_j$ and $v_j \rightarrow v_i$ respectively. $b_{i,j}$ is the maximum payment that v_i can make to v_j, and $b_{j,i}$ is the maximum payment that v_j can make to v_i. $b_{i,j}$ and $b_{j,i}$ are not necessarily the same, and $b_{i,j}$ and $b_{j,i}$ add up to the channel capacity of $[v_i, v_j]$. For simplicity, we assume that there is at most one channel between a specific pair of nodes. Every node knows all about the channels in the network since the transaction of building a payment channel will be broadcast to the blockchain. However, for privacy, the channel balance is only transparent to the channel owners. The balances of the channel will only be broadcast to the blockchain when opening and closing transactions. To ensure the atomicity of transactions, we do not consider using multiple paths for one transaction.

In this paper, we consider a centralized routing protocol for PCNs, where the PCN operator manages the entire network and calculates the optimal path for users. This model conforms to the practice of the Lightning Network, where the sender sends a payment request to the service operator [9]. If the request is reasonable, the operator will find the best payment path and return it to the sender. Distributed routing may be

more privacy-preserving but the communication overhead is high. Currently, there are some works dedicated to solving the privacy issues regarding centralized routing in PCNs [41]. In our future work, we aim to design efficient routing protocols that will help protect user privacy.

A payment request is denoted as $R(v_s, v_r, \alpha)$, where v_s and v_r are the sender and the recipient respectively, and α is the amount of the payment to be transferred. A path $P(v_s, v_r)$ through which the sender v_s can pay to the recipient v_s is expressed as

$$P(v_s, v_r) = (v_0, \ldots, v_{n+1}) \tag{1}$$

where $v_0 = v_s$, $v_{n+1} = v_r$, $v_1, \ldots, v_n \in V$, $[v_i, v_{i+1}] \in E$, $\forall i \in [0, n]$. All payment channels in a path are called delivery channels (DCs), and all nodes in the path except the sender and receiver are called *intermediate nodes*. There are totally four kinds of fees for every intermediate nodes.

Fixed fee: For each transaction forwarded by the *i*-th intermediate nodes, it will charge a fixed fee f_i^{fix}. For a path $P(v_s, v_r) = (v_0, \ldots, v_{n+1})$, the total fixed fee is

$$F^{fix} = \sum_{i=1}^{n} f_i^{fix}. \tag{2}$$

Proportional fee: For each transaction forwarded by the *i*-th intermediate nodes, it will charge a fee proportional to the transaction amount. For example, if a payment of α is forwarded, the *i*-th intermediate nodes will charge a proportional fee $f_i^{prop} = k_{p_i} \times \alpha$, where k_{p_i} is the proportional rate. For a path $P(v_s, \ldots, v_r) = (v_0, \ldots, v_{n+1})$, the total proportional fee is

$$F^{prop} = \sum_{i=1}^{n} f_i^{prop}. \tag{3}$$

Time value fee: The sender compensates intermediate nodes for locking up the money in the channel for a predetermined maximum period of time [9]. The time value fee depends on the length of the path from the intermediate node to the recipient node. The intermediate node at the start of the path is farther away from the recipient and needs to lock the money for a longer period of time. Therefore, it will charge a higher time value fee.

For a path $P(v_s, v_r) = (v_0, \ldots, v_{n+1})$, the time value fee of v_i is

$$f_i^{tv} = (n + 1 - i)k_{t_i}\alpha, \tag{4}$$

where k_{t_i} is the time value rate of the *i*-th intermediate node. The total time value fee of path *P* is

$$F^{tv} = \sum_{i=1}^{n} f_i^{tv}. \tag{5}$$

Imbalance fee: Since the total capacity of a payment channel is fixed, transferring the payment via a payment channel will increase the balance of one user and decrease the balance of the other, which may lead to imbalance. However, once the funds of one party in the payment channel are used up due to too many unidirectional transactions, the channel will need to be closed, which will result in expensive on-chain transactions. Thus intermediate nodes will charge for unbalancing the channel, determined by the Imbalance Penalty (IP) function. The IP function describes how much the intermediate nodes are willing to pay to bring a payment channel into the most-preferred state. Some PCNs allow each node to define their customized IP functions. In our implementation, we adopt a quadratic function as the IP function, as shown in Fig. 3. Note that the IP function can take any form, and we adopt the quadratic form since it imposes a larger penalty on a more severe imbalance. Let $\delta_{i,j} = |b_{i,j} - b_{j,i}|$ denote the imbalance degree of the payment channel (v_i, v_j). The IP function is

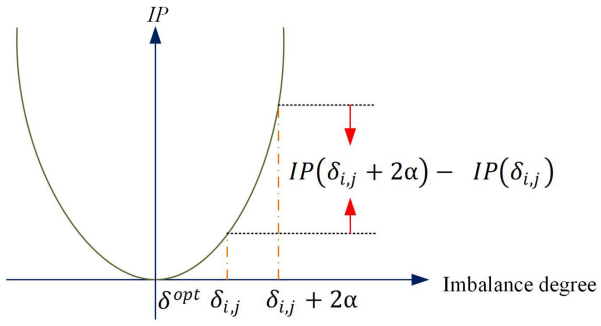

Fig. 3. The quadratic imbalance penalty (IP) function.

$$IP(\delta_{i,j}) = \beta(\delta_{i,j} - \delta^{opt})^2, \tag{6}$$

where β is a constant parameter, and δ^{opt} is the most-preferred state. The *i*-th intermediate node will charge an imbalance fee of $f_i^{im} = IP(\delta_{i,i+1} + 2\alpha) - IP(\delta_{i,i+1})$, where $\delta_{i,j} + 2\alpha$ is the imbalance degree after payment transfer.

For a path $P(v_s, v_r) = (v_0, \ldots, v_{n+1})$, the total imbalance fee is

$$F^{im} = \sum_{i=1}^{n} f_i^{im}, \tag{7}$$

For a path $P(v_s, v_r) = (v_0, \ldots, v_{n+1})$, the total fee is

$$F = F^{fix} + F^{prop} + F^{tv} + F^{im}. \tag{8}$$

C. Problem Formulation

Given a payment request $R(v_s, v_t, \alpha)$, our objective is to find a path *P* with the minimum fee subject to the capacity constraint (CC).

A path $P(v_s, v_r) = (v_0, \ldots, v_{n+1})$ satisfies the capacity constraint, if each intermediate node has sufficient balance to forward the payment. Specifically, the balance $b_{i,i+1}$ of (v_i, v_{i+1}) should be no less than the sum of α and the fees paid to the subsequent intermediate nodes on the path. Thus we have:

$$b_{i,i+1} \geq \alpha + \sum_{j=i}^{n} \left[f_j^{fix} + f_j^{prop} + f_j^{tv} + f_j^{im} \right]. \tag{9}$$

Apparently, more than one path from the sender to the recipient may exist that satisfies the capacity constraint. Our goal is to find the path with the minimum fee. For the single fixed fee, proportional fee, and imbalance fee, finding the path with the minimum fee is a minimum cost routing problem. However,

Algorithm 1 The Process of Payment Request

Payment Request.
Input: The recipient's address A_r.
OutInput: The state of the request, *Reqstate*.
1. The sender requests the recipient's address.
2. If the recipient's address is invalid or the recipient does not accept payments, do not send the payment request and set the request state, *Reqstate* $= 2$.
3. If balances of all channels of the sender are less than the payment amount α then cancel this payment and set the request state, *Reqstate* $= 3$.
4. The sender encapsulates its own address, the recipient's address and the payment amount into a payment request $R(s, r, \alpha)$,then it sends it to the path-finding unit. Set the request state, *Reqstate* $= 4$
5 The sender wait for the path *p* from path-finding unit
6 If the sender receive a path *p* from path-finding unit, and set the request state, *Reqstate* $= 1$, turn to **Payment Execution**.
7 End.

Algorithm 2 The Process of Path-Finding

Path-finding.
Input: Addresses of the sender A_s and the recipient A_r.
OutInput: The path *p*.
1. Set the recipient as the starting node v_s and the sender as the destination node v_r.
2. Initialize the distance from v_s, and the hop number of all nodes. If $[v_i,v_s]\in E$, $dis_i = 0$, $hop_i = 0$. Else, $dis_i = +\infty$, $hop_i = 0$
3. Initialize the set *VIS* of nodes which the shortest path has been found and the set *UVIS* of nodes which the shortest path has not been found.
   $VIS = \phi$, $UVIS = V$
4. Move the node with minimum *dis* v_m from *UVIS* to *VIS*.
5. For every node $v_i \in$ *VIS*, if $\exists v_u$, $[v_i,v_u]\in E$ and $dis_u > dis_i + f_u$, then
   update $dis_u = dis_i + f_u$.
   update the hop number and previous node of node v_u, hop_u++, $prep_u = v_i$.
7 If $VIS! = V$ and $v_s \notin VIS$, then return to step 4.
8 Initialize the path *p* and current node of the path.
   $p = \{v_s\}$, $v_{cur} = v_s$.
9 Set the previous node of the current node as the current node, $v_{cur} = pre_{cur}$, add pre_{cur} to p.
10 If $pre_i! = v_r$ and $pre_i! = null$, then send *p* to the sender. Else, return to Step 9.
11 End.

for the time value fee, finding the path with the minimum fee is a minimum hop problem. Due to the existence of the time value fee that is dependent on the length of the path from an intermediate node to the destination, the optimization problem cannot be solved by traditional shortest path algorithms where the weight of a link is fixed and does not depend on the path length. Therefore, we design MPCN-RP, which can find the optimal path for transactions in PCNs.

IV. Detail Construction

MPCN-RP can be integrated into existing payment channel networks such as the Lightning Network as a routing service. Our solution consists of four steps, namely *payment request*, *path finding*, *payment execution*, and *update*. In the *payment request* stage, the sender of the payment constructs a payment request and sends it to the MPCN-RP service provider (e.g., the payment channel network operator). In the *path finding* stage, the service provider finds the optimal payment path according to the payment request and returns the result to the payment sender. In the *payment execution* stage, the sender executes the payment through HTLC. The network topology is periodically updated in the *update* stage. We summarize key symbols in Table I.

A. Algorithm Design of MPCN-RP

1) Payment Request: As shown in Algorithm I, in the payment request stage, the sender first asks for the recipient's address. If the recipient's address is invalid, the payment request will be denied. If the balance of any channels of the sender is less than the payment amount, the payment will be canceled. Otherwise, the sender encapsulates its address, the recipient's address, and the payment amount into a payment request $R(v_s, v_R, \alpha)$, and sends it to the path-finding unit (PFU).

2) Path Finding: The detailed process of path finding is summarized in Algorithm II. First of all, we reverse the recipient node as the starting node v_s and the sender node as the destination node v_r. This is because we need to know the number of hops to the recipient node in order to estimate the time value fee. We build two sets *VIS* and *UVIS*, where *VIS* contains the nodes whose shortest path to the source node has been found, and *UVIS* contains the remaining nodes. In the initialization stage, $VIS = \varnothing$ while *UVIS* $= V$. We initialize the distance from all nodes to the starting node v_s and the number of hops between each node and the starting node as follows. If there is an edge between node v_i and v_s, the distance of v_i is initialized as 0, and the number of hops of v_i is set as 0. Otherwise, the distance and the number of hops of v_i is initialized as $+\infty$. Note that if the balance of a node is not enough to pay, its distance is always set as $+\infty$. After that, we iteratively finds the node v_m that has the minimum distance from the starting node, and transfer v_m from *UVIS* to *VIS*. Then, we update the distance of the neighbors of v_m if a shorter path is available through v_m and update their numbers

Algorithm 3 The Process of Payment Execution

Payment Execution.
Input: Addresses of the sender A_s and the recipient A_r.
OutInput: The state of the payment *Paystate*.
1. Confirm the address of the recipient and set the current node $v_{cur} = v_r$. Set the state of the payment, *Paystate* $= 0$.
2. Calculate the fee f_{cur-1} that will be charged from the previous node to the current node.
3. If the current node publishes H(R) to the previous node before the end of the time lock.
   $b_{cur-1,cur}$-α-$f_{cur-1,cur}$, $b_{cur,cur-1}$+α+$f_{cur-1,cur}$. Set the previous node of the current node as the current node, v_{cur}=pre_{cur}. Return to Step 2.
4. Else, the payment fails. Set the state of the payment, *Paystate* $= 2$. Go to Step 6.
5. Set the state of the payment, *Paystate* $= 1$.
6. End.

TABLE I
KEY NOTATIONS

Notation	Description
G	The payment channel network
V	Node set in G
E	Edge set in G
v	The i-th node of G
v_s	The sender of the payment
v_r	The recipient of the payment
v_n	The n-th node of the path P
$b_{i,j}$	The balance of node v_i on the channel $[v_i, v_j]$
$e_{i,j}$	Used to mark whether there is a channel between nodes i and j
α	The payment amount of the payment
ff_i	The fixed fee of the i-th node
pf_i	The proportional fee of the i-th node
k_p	k_p is the proportional parameter of all IN
tvf_i	The time value fee of the i-th node
k_t	k_t is the proportional parameter of all IN
if_i	The imbalance fee of the i-th node
f_i	The total fee of the i-th node
dis_i	Distance from the i-th node to the source node
hop_i	The number of hops from the i-th node to the source node
pre_i	The previous node of node i
cur	Current node when calculating the path p
$C_{i,j}$	The message to close the channel between node i and j
$O_{i,j}$	The message to open the channel between node i and j
$U_{j,i}$	The message to update the channel between node i and j

of hops accordingly. Note that if the balance of a neighboring node of v_m is insufficient, we do not make any changes to its distance. Once the destination node is in *VIS*, we stop the iteration process. As a result, we find a shortest path from the destination node to the source node.

The path-finding algorithm will terminate since there is a limited number of nodes in the PCN. At each iteration, one node is inserted into *VIS*, thus there is a limited number of iterations. The major difference between MPCN-RPand Dijkstra algorithm is that Dijkstra assumes that the cost of a path is the sum of weight of all edges on the path. In our problem, the cost of a path also includes the number of hops to the destination node. Therefore, the path found by Dijkstra may not have the lowest cost if the path contains many hops even if the total distance is short. In contrast, our algorithm intends to find a path that has both short length and total weight.

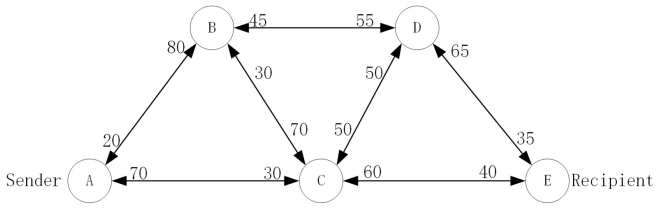

Fig. 4. Balances of nodes in G'.

3) Payment Execution: Payment is executed along path *P* through HTLC. Algorithm III shows the detailed process of payment execution. Suppose the nodes in the path *P* are $v_1, v_2 \cdots v_n$. Firstly $v_{(n-1)}$ pays to v_n, then $v_{(n-2)}$ pays to $v_{(n-1)}$. The above process is repeated until the node that pays is the starting node.

4) Update: The path-finding unit needs to update its knowledge of network topology periodically in order to find a satisfactory path. Algorithm IV shows the detailed process of update. If a channel $[v_i, v_j]$ is closed, node v_i and v_j send $C_{i,j}$ and $C_{j,i}$to the PFU. After receiving $C_{i,j}$ and $C_{j,i}$ the PFU deletes $[v_i, v_j]$ from *E* and changes the flags as $e_{i,j} = 0$ and $e_{j,i} = 0$. If a channel between $[v_i, v_j]$ is been built, node v_i and v_j send $O_{i,j}$ and $O_{j,i}$ to the PFU. After receiving $O_{i,j}$ and $O_{j,i}$, the PFU adds $[v_i, v_j]$ to *E* and changes the flags as $e_{i,j} = 1$ and $e_{j,i} = 1$, then requests $b_{i,j}$ and $b_{j,i}$. If the balances of the channel $[v_i, v_j]$ is changed, node v_i and v_j send $U_{i,j}$ and $U_{j,i}$ to the PFU. After receiving $U_{i,j}$ and $U_{j,i}$, the PFU requests for new balances $b_{i,j}$ and $b_{j,i}$ from the two nodes.

B. A Toy Example of MPCN-RP

To make the path-finding procedure more clear, we a concise example to illustrate the path-finding phase. We assume that the entire network topology of the payment channel is G', as shown in Fig. 4. A double arrow connecting line represents a channel, and the number close to the node represents the balance of it on this channel.

Let's assume that node A wants to transfer 20 to node E. Since the amount that node A wants to transfer to node E is fixed, the proportional fees of all intermediate nodes remain unchanged during the path-finding phase. In addition, when a transfer process does not start, the balances between nodes will not change, so the imbalance fees of all nodes will not change. In summary, all of the fixed fees, proportional fees, and imbalance fees of the intermediate nodes will not change

Algorithm 4 The Process of Update

Update.
Input: The channel state in the network *Chnstate*.
OutInput: End time of the update t_{update}.
1. For all $v_i \in v, v_j \in V$, if the channel $[v_i, v_j]$ between them has been closed, v_i and v_j send $C_{i,j}$ and $C_{j,i}$to the path-finding unit.
2. After receiving $C_{i,j}$ and $C_{j,i}$ the path-finding unit delete $[v_i, v_j]$ from E and change the marks.
   $e_{i,j} = 0$, $e_{j,i} = 0$.
3. For all $v_i \in v, v_j \in V$, if there is a channel between $[v_i, v_j]$ has been built, v_i and v_j send $O_{i,j}$ and $O_{j,i}$to the path-finding unit.
4. After receiving $O_{i,j}$ and $O_{j,i}$ the path-finding unit add $[v_i, v_j]$ to E and change the marks.
   $e_{i,j} = 1$, $e_{j,i} = 1$, request $b_{i,j}$,$b_{j,i}$.
5. For all $v_i \in v, v_j \in V$, if the balances of the channel has been changed then v_i and v_j send $U_{i,j}$ and $U_{j,i}$to the path-finding unit.
6. After receiving $U_{i,j}$ and $U_{j,i}$ the path-finding unit request new balances from the two nodes.
   Request new $b_{i,j}$,$b_{j,i}$.
7. End.

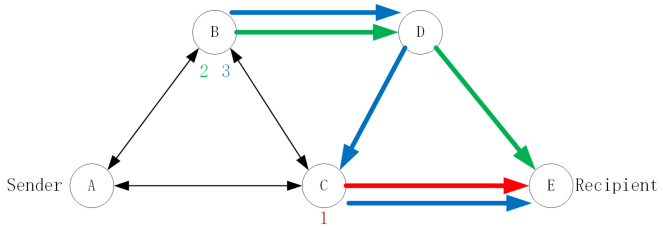

Fig. 5. Illustrative example of time value fee.

during a payment. We assume that the sum of these three fees is $\bar{f}$ and the $\bar{f}$ of all intermediate nodes can be found in Fig. 6. For example, on the channel between node A and node B, the number near A is 3, and the number near B is 4. It means that if node A acts as an intermediate node to transfer money to B, the constant fee $\bar{f}$ is 3.

We assume that the coefficient k_t of time value fee is 0.05, and $\alpha = 20$, then we have $k_t \times \alpha = 1$. That means the time value fee is 1 for nodes that are one hop away from the starting node and 2 for nodes that are 2 hops away from the recipient. The time value fee of a node is shown in Fig. 5. The time value fee of a node is related to the path it is on. For example, for node B, if it only transfers the payment to E through node D, the time value fee of B is 2. If node B passes the payment to E through nodes D and C, then the time value fee of node B is 3.

Now we begin to find the best path for node A to pay to node E. Firstly, we set the receiver as the source node and the sender as the destination node. Next, we initialize the distance from all nodes to the source node. We assume that the set of nodes whose shortest distance to the source node is known as *VIS*, and the set of other nodes is *UVIS*. Since the $\bar{f}$ of nodes C, D are 10 and 3 respectively, and their hops to E are all 1, the distances from nodes C, D to E are $10 + 1$, $3 + 1$, respectively, where 10 and 3 are the $\bar{f}$ of node D and C respectively, the time value fee of node D and C are both 1. The previous node of nodes C and D is E, represented by the letters in parentheses. The distance from nodes A, B to E is unknown, represented by INF, the infinite distance. Here we should note that if the node is connected to the source node, but its balance is not enough to pay, the distance should be INF. Row 2 of Table II shows the state of all nodes after initialization.

Next, we carry out the following iterations. Fig. 6 shows the whole process of the iteration. The green node indicates

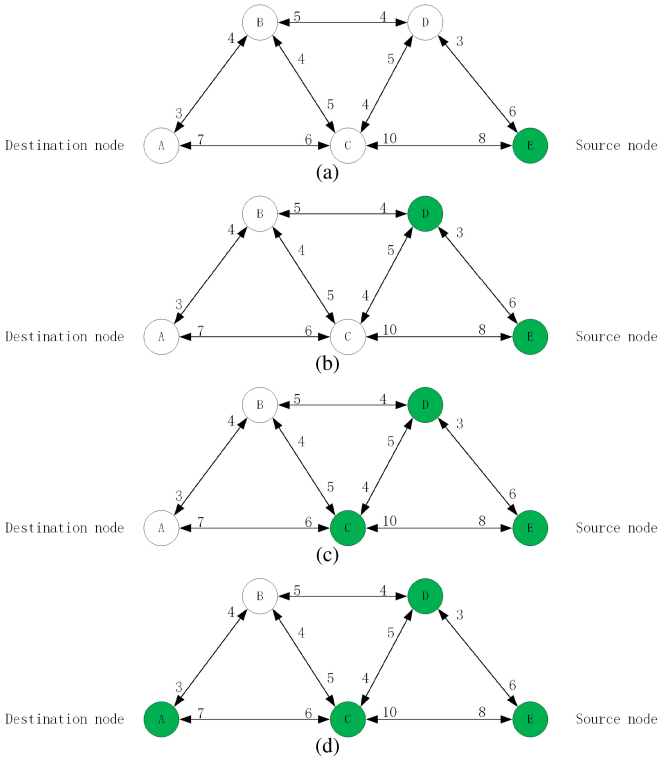

Fig. 6. (a) The state of G' after initialization. (b) The state after the first iteration. (c) The state after the second iteration. (d) The state after the third iteration.

that the node has been added to *VIS*. The third row of Table II shows the result of the second iteration. From the initial state, we can see that the shortest distance to node E is node D. So we add node D to *VIS*. Next, we find all nodes adjacent to node D, except for E. We find that nodes B, C are connected to D. For node B, if we connect through intermediate nodes D, the fees will be $3 + 5 + 1 + 2$, where 3 and 5 are the $\bar{f}$ of node D and B respectively, and 1 and 2 are the time value fee of D and B respectively. For node C, if we connect through intermediate nodes D, the fees will be $3 + 4 + 1 + 2$, where 3 and 4 are the $\bar{f}$ of node D and C respectively, and 1 and 2 are the time value fee of D and C respectively. Since the old distance of C is $10 + 1$ and is greater than the distance through D $3 + 4 + 1 + 2$, we update the distance of C to $3 + 4 + 1 + 2$, and change the previous node of C to D.

TABLE II
THE ITERATION PROCESS

Number of iterations	Nodes that join VIS at each iteration	VIS	Distance to start node				
			A	B	C	D	E
Initial state	-	∅	INF	INF	10+1(E)	3+1(E)	0
1	D	D	INF	3+5+1+2(D)	3+4+1+2(D)	3+1(E)	0
2	C	DC	3+4+1+2(C)	3+5+1+2(D)	3+4+1+2(D)	3+1(E)	0
3	A	DCA	3+4+1+2(C)	3+5+1+2(D)	3+4+1+2(D)	3+1(E)	0

TABLE III
THE ITERATION PROCESS WHEN C'S BALANCE IS NOT ENOUGH TO PAY

Number of iterations	Nodes that join VIS at each iteration	VIS	Distance to start node				
			A	B	C	D	E
Initial state	-	∅	INF	INF	10+1(E)	3+1(E)	0
1	D	D	INF	3+5+1+2(D)	10+1(E)	3+1(E)	0
2	C	DC	10+7+1+2(C)	3+5+1+2(D)	10+1(E)	3+1(E)	0
3	B	DCB	3+5+3+1+2+3(B)	3+5+1+2(D)	10+1(E)	3+1(E)	0
4	A	DCBA	3+5+3+1+2+3(B)	3+5+1+2(D)	10+1(E)	3+1(E)	0

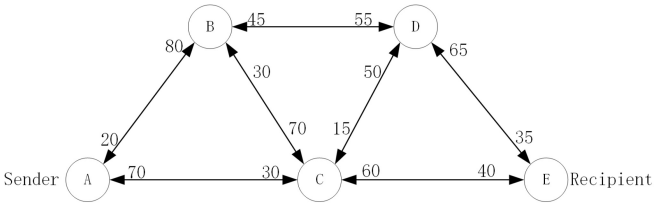

Fig. 7. Node C's balance is not enough to pay.

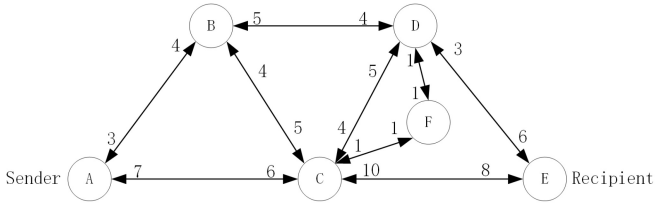

Fig. 8. G' after node F is added.

Then we start the second round of iteration. The fourth row of Table II shows the results of the second iteration. Now the shortest distance to node E is node C. So we add node C to *VIS*. Next, we find all nodes adjacent to node C, except for its previous node D. We find that nodes A and B are connected to C. For node A, if we connect through intermediate nodes C, the fees will be $3 + 4 + 1 + 2$, where 3 and 4 are the $\bar{f}$ of node D and C respectively, and 1 and 2 are the time value fee of D and C respectively. Since A is the destination node, we do not calculate A's fees. Next, we set the previous node of A to C. For node B, if we connect through intermediate nodes C, the fees will be $3 + 4 + 4 + 1 + 2 + 3$, where 3,4, and are the $\bar{f}$ of node D, C, and B respectively, and 1, 2, and 3 are the time value fee of D, C, and B respectively. Since the old distance of C is $10 + 1$ and is smaller than the distance through C, we don't update the distance of B.

Next, we start the third iteration. We find that A is the closest node to E in *UVIS*, so we add A to *VIS*. Now that the destination node is already in *VIS*, we stop iterating.

We assume that the balance of C on the channel between node C and node D is less than 20, as shown in Fig. 7. The iteration process is shown in Table III. In the first iteration, we added D to *VIS*. When calculating the distance of node C, we find that the balance of C is not enough to pay to node D, so we do not update the distance of node C.

Now that the destination node is already in *VIS*, we stop iterating. Now we find the shortest path from the destination node to the source node. We start from node A, the previous node of A is C, the previous node of C is D, and the previous node of D is E. So our path is (A, C, D, E). In the process of finding a path, the most critical step is to set the receiver as the source node. In this way, we can add time value fee based on the number of hops from the source node in the process of calculating the fee. If we still look for the shortest path from the sender as in the existing method, we don't know the number of hops from the current node to the receiver, so we cannot calculate the time value fee. Now we give an example to illustrate the necessity of setting the receiver as the source node. Since we don't know the number of hops from the recipient, we can only use $\bar{f}$ as the full charge for the time being, and add time value fee after finding the shortest path. We assume that there is another node F, whose location and charges are shown in Fig. 8. If we set the sending node as the source node, the iteration results are shown in Table IV, the shortest path found is (A, C, F, D, E). If the receiving node is set as the source node, the iteration results are shown in Table V. The shortest path is then (A, C, D, E). Therefore, since the number of hops from the node to the receiving node is unknown, starting from the sending node to find the shortest path may have confusing results.

V. EVALUATION

In this section, we comprehensively evaluate the performance of MPCN-RP.

A. Experiment Setup

We implement a fully-functional prototype of MPCN-RP on the Linux server with 4 CPU (Intel Xeon CPU E5-2429 V2

TABLE IV
THE ITERATIVE PROCESS OF SETTING THE SENDER AS THE STARTING NODE

Number of iterations	Nodes that join VIS at each iteration	VIS	Distance to start node					
			A	B	C	D	E	F
Initial state	-	∅	0	0	0	INF	INF	INF
1	BC	BC	0	(A)	0(A)	4(C)	10(C)	1(C)
2	F	BCF	0	0(A)	0(A)	1+1(F)	10(C)	1(C)
3	D	BCFD	0	0(A)	0(A)	1+1(F)	1+1+3(D)	1(C)
4	E	BCFDE	0	0(A)	0(A)	1+1(F)	1+1+2(D)	1(C)

TABLE V
THE ITERATIVE PROCESS OF SETTING THE RECEIVER AS THE STARTING NODE

No. of iterations	Nodes joining VIS	VIS	Distance to start node					
			A	B	C	D	E	F
Initial state	-	∅	INF	INF	10+1(E)	3+1(E)	0	INF
1	D	D	INF	3+5+1+2(D)	3+4+1+2(D)	3+1(E)	0	3+1+1+2(D)
2	F	DF	INF	3+5+1+2(D)	3+4+1+2(D)	3+1(E)	0	3+1+1+2(D)
3	C	DFC	3+4+1+2(C)	3+5+1+2(D)	3+4+1+2(D)	3+1(E)	0	3+1+1+2(D)
4	A	DFCA	3+4+1+2(C)	3+5+1+2(D)	3+4+1+2(D)	3+1(E)	0	3+1+1+2(D)

@2,20GHz) and 15,532MB of memory RAM. In the experiment, all charging strategies of the payments remain the same. We compare MPCN-RP with the following four baseline algorithms.

- Dijkstra-original: MPCN-RP is developed based on the classic Dijkstra algorithm. However, the original Dijkstra algorithm only considers one path weight, and it can only solve the problem that the weight on the path is not related to the hop count of the path. Therefore, it is necessary for us to compare the traditional Dijkstra algorithm with MPCN-RP to evaluate the influence of our solution on the weights related to the number of hops in the final result.
- Dijkstra-LN: The Lightning Network, one of the most widely known payment channel networks, uses an improved Dijkstra algorithm to calculate the fees of intermediate nodes. In the Lightning Network payment path, the penultimate node pays first to the recipient, followed by the penultimate node pays to the second node. So in Dijkstra-LN, the sender and receiver are exchanged when looking for the shortest path, making it simple to calculate the weight. However, Dijkstra-LN simply sets the receiver as the starting node and only considers the simplified maximum lock time fees in worst case, let alone fees related to hop number.Therefore, it is necessary to compare the difference in terms of running time and fees between MYPBT and LN.
- MPCN-RP-cut: In order to verify the effectiveness of reversing the sender and the recipient in MPCN-RP, we implement an algorithm that does not perform the reversion, referred to as MPCN-RP-cut. Since the proportional fee is zero for the payment channel between the sender and the first IN, we execute the pathfinding algorithm from every node adjacent to the sender. Then we choose the cheapest path that satisfies the capacity constraint.
- DFS: Depth First Search (DFS) can find the path with a minimum fee by traversing all the paths. However, for large-scale networks, traversing all paths will be computationally expensive. Therefore, it is necessary to compare the difference in running time between MYPBT and DFS. We set a limit that DFS will find no more than 30 paths.

B. Simulated Networks

We evaluated the performance of MPCN-RP on different simulated virtual networks with different scales of networks by varying the number of network nodes (from 100 to 1,900). Then, we changed the average number of channels owned by each node (from 1 to 9), referred to as network density. The channel of each node is randomly connected to another node and the capacity is a random value between 0 and the maximum value in Lighting Network. Furthermore, we measured the effect of the payment amount on the performance of the algorithms. In particular, it should be noted that we set our parameters according to the previous simulation experiments in actual scenes [16], [22]. It can be found that our parameters are similar to their parameter settings, at least an order of magnitude.

Fig. 9 (a), Fig. 10 (a), Fig. 11 (a) and Fig. 12 (a) show the impact of network size on different algorithms. We assume that each node has about 4 to 8 channels. To better compare the differences between the algorithms, we performed up to 500 payments for each algorithm. As shown in Fig. 9 (a), Fig. 10 (a), with the increase of network size, the number of hops in the path between the sender and the recipient increases, resulting in higher transaction fees. However, the number of hops and the fee of the original Dijkstra and the Dijkstra-LN are significantly higher than the other three algorithms. The original Dijkstra performs slightly worse than Dijkstra-LN since the former does not deal with the path-dependent time value fee. There is not much difference in transaction fees between MPCN-RP, MPCN-RP-cut, and DFS, but MPCN-RP-cut and DFS have longer running time than MPCN-RP. Fig. 11 (a) shows the average running time of the 5 algorithms. It can be seen that DFS takes a much longer time than

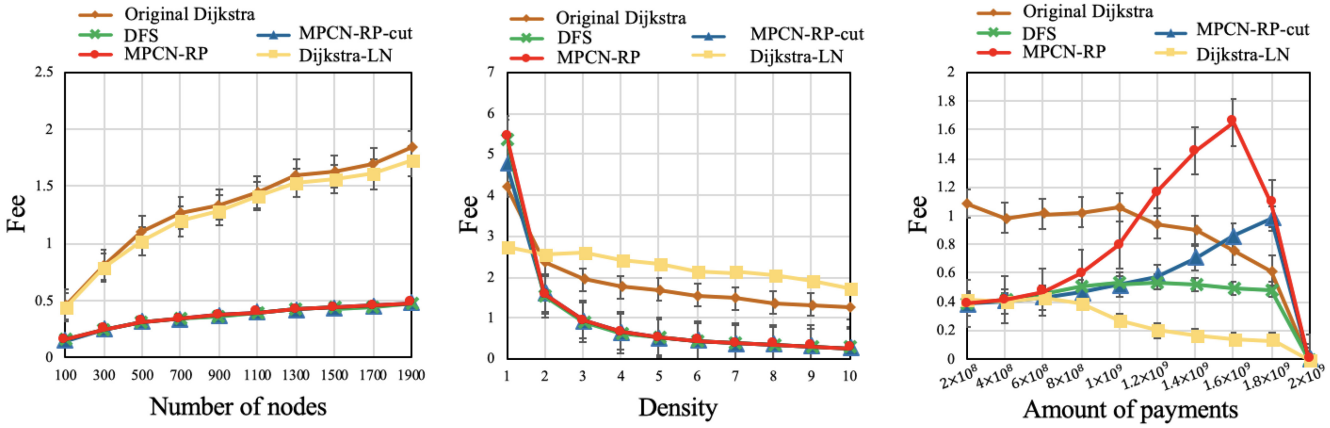

Fig. 9. Experiment results in simulated networks. Fee versus (a) network size (b) channel density (c) payment.

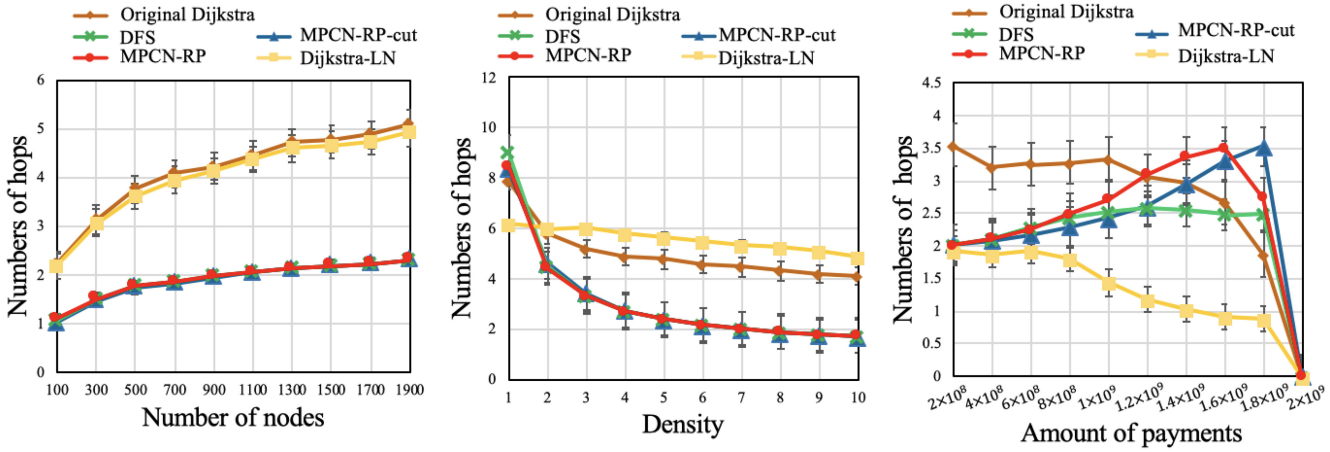

Fig. 10. Experiment results in simulated networks. Number of hops versus (a) network size (b) channel density (c) payment.

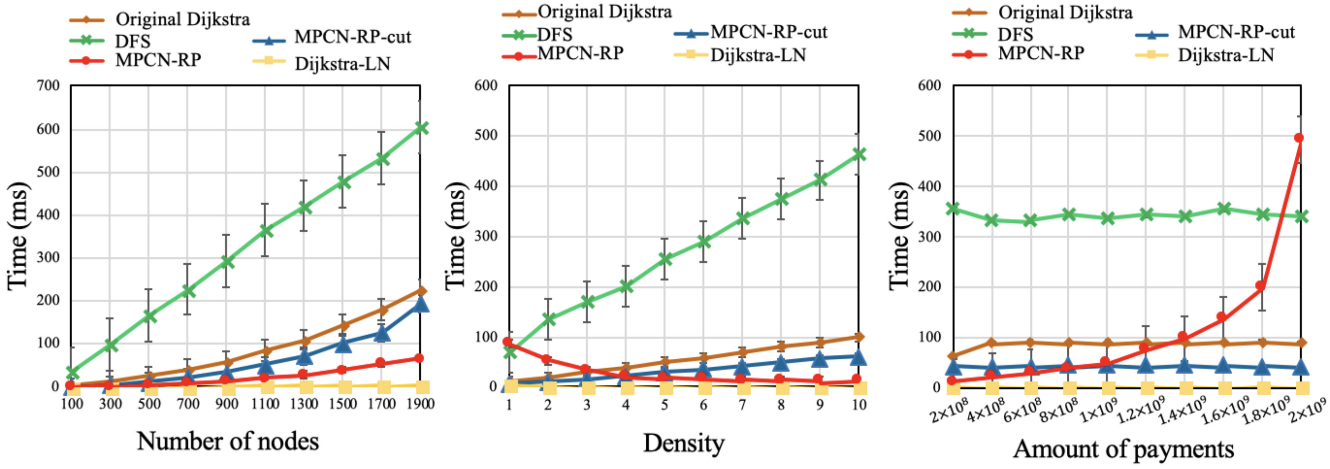

Fig. 11. Experiment results in simulated networks. Time versus (a) network size (b) channel density (c) payment.

the other four algorithms. That is because DFS exhausts all possible paths, which is extremely time-consuming. The time consumption of MPCN-RP is the second shortest, only slightly longer than Dijkstra-LN. The average time for finding a path for each transaction is within 100ms, which is quite fast for real applications. If an algorithm fails to find a path that satisfies the capacity constraint, the payment will fail. We quantify the success ratio as the number of failed payments to the total number of payments. Fig. 12 (a) shows the success ratio of the five algorithms. The success ratio of these algorithms is close to 1 except for the original Dijkstra. In conclusion, MPCN-RP can achieve a minimum fee that is close to the optimal results of DFS with a much shorter time. Since the payment channel network is dynamic with a frequent update of payment channels, MPCN-RP is able to find a valid cost-effective path in time.

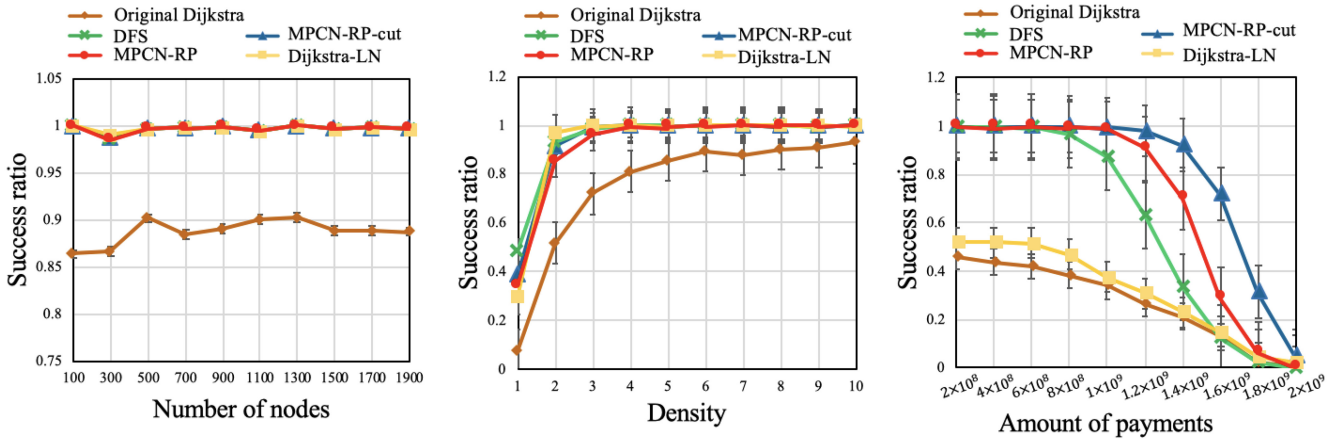

Fig. 12. Experiment results in simulated networks. Success ratio versus (a) network size (b) channel density (c) payment.

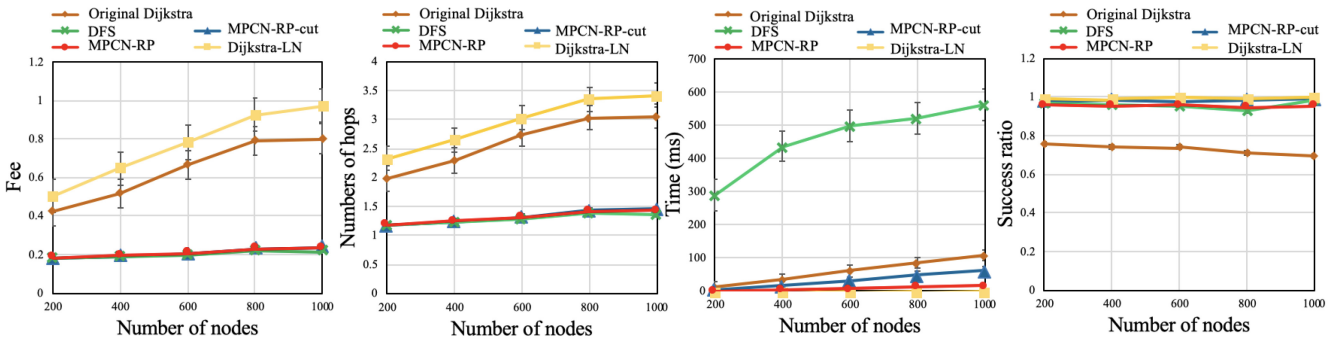

Fig. 13. Experiment results in real networks. (a) fee (b) number of hops (c) time costs (d) success ratio.

Fig. 9 (b), Fig. 10 (b), Fig. 11 (b) and Fig. 12 (b) show the effect of channel density on the performance of different algorithms. We construct a simulated network with 1,000 nodes and vary the number of channels. Similarly, for each algorithm, we executed 500 payments. As shown in Fig. 9 (b) and Fig. 10 (b), with the increase of channels in the network, users are more connected, and there are more choices for paths. Thus the number of hops between the sender and the recipient decreases, resulting in reduced fees. More channels in the network make it more likely that there is a shorter path between the sender and the recipient. The number of hops and the fee of the original Dijkstra and Dijkstra-LN are obviously higher than the other three algorithms. MPCN-RP, MPCN-RP-cut, and DFS are similar with the lowest fees. Fig. 11 (b) shows the average running time of the five algorithms. It is shown that DFS still takes the most time, which is significantly higher than the other four algorithms. The time of MPCN-RP decreases with the increase of density. Fig. 12 (b) shows the success ratio of the five algorithms. When the number of channels in the network is very small, the success ratio of all five algorithms is very small. The success ratio of MPCN-RP increases to 1 as the channel density exceeds 4, which suits the real payment channel network topology.

Fig. 9 (c), Fig. 10 (c), Fig. 11 (c) and Fig. 12 (c) show the effect of payment amount on the five algorithms. We use a simulated network with 1,000 nodes, and the average number of channels per node is 6. As shown in Fig. 9 (c) and Fig. 10 (c), with the increase of payment amount, the fee of the original Dijkstra algorithm and Dijkstra-LN shows a downward trend. The fees of MPCN-RP MPCN-RP-cut, and DFS first increase, but when the payment amount reaches a certain threshold, the fees decrease. As can be seen from Fig. 12 (c), the success ratio tends to 0 when the payment amount is too high. This is because when the payment amount exceeds the capacity of the payment channel network, there will be no path in the network to fulfill the payment, resulting in no fee and zero success ratio.

C. Real Network

We evaluate the performance of MPCN-RP with a real node data set extracted from the Lightning Network. Particularly, it is the complete network data from November 2019 to May 2020. In order to evaluate the influence of network size, we extracted induced subgraphs composed of $200, 400, \ldots, 1,000$ nodes, respectively. Fig. 13 shows the impact of network size on the five algorithms in a real network. Similarly, for each algorithm, we experimented with 500 payments. It can be seen from Fig. 13 (a) and (b) that with the increase of network nodes, the number of path hops between the two parties in the network also increases, increasing the fee. Like the simulated virtual networks, the number of hops and the fee of the original Dijkstra and Dijkstra-LN are significantly higher than the other three algorithms. MPCN-RP, MPCN-RP-cut, and DFS have no significant difference and perform the best. Fig. 13 (c) shows the average time of the payments. We can see that DFS takes

TABLE VI
THE PROCESS OF UPDATE

Algorithm	Time complexity.	Space complexity.
MPCN-RP	$O(N^2)$	$O(N)$
Dijkstra-original	$O(N^2)$	$O(N)$
Dijkstra-LN	$O(N^2)$	$O(N)$
MPCN-RP-cut	$O(N^2)$	$O(N)$
DFS	$O(N^3)$	$O(N^2)$

TABLE VII
EXPERIMENT RESULTS IN REAL NETWORKS WITH 24,988 NODES ON THE LIGHTNING NETWORK

Baselines	Fee	Number of hops	Success ratio	Times (ms)
Original Dijkstra	2.266263	7.8297640	0.5874630	2512.9886
DFS	0.607157	3.8825478	0.9536470	5427.8525
MYPCN-RP-cut	1.028731	4.4357850	0.9746370	1854.6350
Dijkstra-LN	5.496832	10.463675	0.9892537	22.643425
MYPCN-RP	0.697253	2.0316478	0.9474216	264.86346

the most time, which is significantly higher than the other four algorithms.

We also extracted the topological structure data composed of 24988 nodes on the Lightning Network. We tried 500 payments for each algorithm. The result is shown in Table VII. As shown in Table VII, the performance of MPCN-RP in such real network is also consistent with that in the simulated network. The differences between the four algorithms are more obvious due to the sharp increase in nodes. For fee and number of hops, MPCN-RT, MPCN-RT-cut, and DFS have a relatively small difference in fee and number of hops, while Dijkstra-LN and Original Dijkstra are significantly higher than the other three algorithms. For the average payment time, DFS takes the most time, which is much higher than the other algorithms. It can be seen that MPCN-RT achieves the best balance between fees, the number of hops, and average payment time.

D. Complexity Analysis and System Discussion

1) Complexity Analysis: In Table VI we compare the algorithm complexity of MPCN-RP and these baseline algorithms, where *N* represents the number of nodes. In MPCN-RP, we use the adjacency matrix to store the entire network. After we find the closest node to the starting node among N nodes, we need to traverse *N* nodes to update their distances. So the time complexity of MPCN-RP is $O(N^2)$. We need an array of length *N* to store the distance of each node, so the space complexity of MYPBT is *O*(*N*). Since the entire network is a relatively sparse network, we can use adjacency lists and heaps to optimize the algorithm and reduce the time complexity to $O((n+m)log_n)$ when the number of network nodes increases dramatically. We can easily find that other baseline algorithms based on Dijkstra's have the same complexity as MPCN-RP. The time complexity of DFS to find one path is $O(N^2)$. However, we need to find all paths to find the shortest path, so the time complexity of DFS is $O(N^3)$. What's more, since we need to save all paths, the space complexity is $O(N^2)$.

2) System Discussion: MPCN-RP is compatible with Bitcoin and Ethereum and can generally be applicable to payment channel networks similar to Lightning Network and Raiden based on HTLC. Moreover, the MPCN-RP protocol provides non-blocking progress due to its convergence of the iterative process and independent operation of each stage. Specifically, MPCN-RP guarantees that every payment can find a path as soon as possible and consume fewer fees. However, if the nodes in the network are in different geographical locations, it may cause a communication delay, which may increase the path-finding time of MPCN-RP. Since the real geographic location is difficult to achieve and the network delay only affects the communication speed but not the calculation speed, we didn't simulate network delay in this paper.

VI. DISCUSSION

A. Decentralized Routing

Due to the relatively small network scale, most existing payment channel networks use centralized routing algorithms. However, as the network scale grows, the advantages of distributed routing algorithms can be realized. In decentralized routing algorithms, each node has a routing table consisting of the neighborhood of nearby nodes and paths to several beacon nodes. Flare [15] was proposed as one of the first decentralized routing algorithms for payment channel networks. The speed of routing table establishment and the upper limit of the routing scale of the decentralized routing algorithm have obvious advantages over the traditional algorithm. Therefore, our future work may focus on the distributed routing algorithm based on multiple charges.

B. Multi-Path Routing

Multi-path in a payment channel network is similar to multicast technology in a communication network. As a point-to-multipoint communication, multicast is one of the effective methods to save network bandwidth. In the payment channel network, however, the shortest or cheapest path may not be used by users due to capacity constraints. If payment is decomposed into several different paths, the fee can be reduced to a certain extent, and the capacity on the path can also be maximized. However, there are also many challenges with multi-path payment when maintaining the atomicity of the decomposed payment. How to find multiple paths with less fee is also a tricky issue because it involves whether the paths intersect. To realize multi-path payment, it is required that both the path-finding unit and the INs between the sender and the recipient must support multi-path payment. Our future work may focus on finding multiple paths to implement payment and solve the atomicity problem of decentralized payment.

VII. CONCLUSION

The decentralized ledger for peer-to-peer transactions faces the problem of low throughput issues in meeting growing users and payments. One of the most mature methods to overcome this challenge is payment channel networks, such as

the Lightning Network in Bitcoin or Raiden in Ethereum. However, the existing payment channel networks have different charging strategies, and so far, there is no general charging standardization. How to manage multiple charges of INs and find cheaper paths has become a concern for users.

In this work, we have presented MPCN-RP, a general routing protocol based on multi-charge blockchain payment channel networks. MPCN-RP allows nodes to freely add or delete charging items in a payment channel network with multiple charges. What is more, MPCN-RP achieved to find the path as few charges as possible with very little time consumption. We have implemented MPCN-RP on both simulated and real payment channel networks and evaluated the performance of MPCN-RP on networks of different sizes. Experiments show that MPCN-RP is better than the baseline algorithm in terms of time and economic benefits.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin Project, Manubot, Rep., 2019.

[2] "Ethereum." [Online]. Available: https://ethereum.org/ (Accessed: Dec. 29, 2021).

[3] "Ripple." [Online]. Available: https://ripple.com/? (Accessed: Dec. 29, 2021).

[4] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Proc. Symp. Self-Stabilizing Syst.*, 2015, pp. 3–18.

[5] "Bitcoin Wiki: Bitcoin Scalability Faq." [Online]. Available: https://en.bitcoin.it/wiki/Scalability_FAQ (Accessed: Dec. 29, 2021).

[6] "Bitcoin Wiki: Bitcoin Contract." [Online]. Available: https://en.bitcoin.it/wiki/Contract (Accessed: Dec. 29, 2021).

[7] "Bitcoin Wiki: Block Size Limit Controversy." [Online]. Available: https://en.bitcoin.it/wiki/Block_size_limit_controversy (Accessed: Dec. 29, 2021).

[8] W. Hao *et al.*, "Towards a trust-enhanced blockchain P2P topology for enabling fast and reliable broadcast," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 904–917, Jun. 2020.

[9] "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments." [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf (Accessed: Dec. 29, 2021).

[10] "The Raiden Network." [Online]. Available: https://raiden.network (Accessed: Dec. 29, 2021).

[11] "Plasma." [Online]. Available: https://github.com/ethereum-plasma/plasma (Accessed: Dec. 29, 2021).

[12] C. Doll and A. Schaffer, "Economic impact of the introduction of the german HGV toll system," *Transp. Policy*, vol. 14, no. 1, pp. 49–58, 2007.

[13] L. Andronis, M. Maredza, and S. Petrou, "Measuring, valuing and including forgone childhood education and leisure time costs in economic evaluation: Methods, challenges and the way forward," *Social Sci. Med.*, vol. 237, Sep. 2019, Art. no. 112475.

[14] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "SilentWhispers: Enforcing security and privacy in decentralized credit networks," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2017, pp. 1–15.

[15] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An approach to routing in lightning network," Bitfury Group Limited, Amsterdam, The Netherlands, White Paper, 2016.

[16] R. Yu, G. Xue, V. T. Kilari, D. Yang, and J. Tang, "CoinExpress: A fast payment routing mechanism in blockchain-based payment channel networks," in *Proc. IEEE Int. Conf. Comput. Commun. Netw.*, 2018, pp. 1–9.

[17] E. Rohrer, J.-F. Laß, and F. Tschorsch, "Towards a concurrent and distributed route selection for payment channel networks," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Cham, Switzerland: Springer, 2017, pp. 411–419.

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

[19] C.-P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.

[20] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2536–2549, Dec. 2020.

[21] J. An, H. Yang, X. Gui, W. Zhang, R. Gui, and J. Kang, "TCNS: Node selection with privacy protection in crowdsensing based on twice consensuses of blockchain," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 3, pp. 1255–1267, Sep. 2019.

[22] Y. Zhang, D. Yang, and G. Xue, "Cheapay: An optimal algorithm for fee minimization in blockchain-based payment channel networks," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[23] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Zug, Switzerland, Ethereum, Yellow Paper, 2014.

[24] E. B. Sasson *et al.*, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. IEEE Symp. Security Privacy*, 2014, pp. 459–474.

[25] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for Bitcoin and cryptocurrencies," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 104–121.

[26] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (SoK)," in *Proc. Int. Conf. Principles Security Trust*, 2017, pp. 164–186.

[27] "Thunder Network: Off-Chain Bitcoin Payments Using Smart Contracts." [Online]. Available: https://github.com/blockchain/thunder (Accessed: Dec. 29, 2021).

[28] "Eclair: A Scala Implementation of the Lightning Network." [Online]. Available: https://github.com/ACINQ/eclair (Accessed: Dec. 29, 2021).

[29] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. Fanti, and P. Viswanath, "Routing cryptocurrency with the spider network," in *Proc. ACM Workshop Hot Top. Netw.*, 2018, pp. 29–35.

[30] V. Bagaria, J. Neu, and D. Tse, "Boomerang: Redundancy improves latency and throughput in payment-channel networks," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2020, pp. 304–324.

[31] P. Wang, H. Xu, X. Jin, and T. Wang, "Flash: Efficient dynamic routing for offchain networks," in *Proc. Int. Conf. Emerg. Netw. Exp. Technol.*, 2019, pp. 370–381.

[32] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 35–42, 1988.

[33] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theor. Comput. Sci.*, vol. 344, no. 1, pp. 3–14, 2005.

[34] F. Engelmann, H. Kopp, F. Kargl, F. Glaser, and C. Weinhardt, "Towards an economic analysis of routing in payment channel networks," in *Proc. ACM Workshop Scalable Resilient Infrastruct. Distrib. Ledgers*, 2017, pp. 1–6.

[35] V. Sivaraman *et al.*, "High throughput cryptocurrency routing in payment channel networks," in *Proc. USENIX Symp. Netw. Syst. Design Implement.*, 2020, pp. 777–796.

[36] L. Eckey, S. Faust, K. Hostáková, and S. Roos, "Splitting payments locally while routing interdimensionally," IACR, Lyon, France, Rep. 555/2020, 2020.

[37] Y. Zhang and D. Yang, "RobustPay: Robust payment routing protocol in blockchain-based payment channel networks," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, 2019, pp. 1–4.

[38] I. Alqassem and D. Svetinovic, "Towards reference architecture for cryptocurrencies: Bitcoin architectural analysis," in *Proc. IEEE Int. Conf. Internet Things IEEE Green Comput. Commun. IEEE Cyber Phys. Social Comput.*, 2014, pp. 436–443.

[39] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards bitcoin payment networks," in *Proc. Aust. Conf. Inf. Security Privacy*, 2016, pp. 57–76.

[40] "Reaching the Ground with Lightning." [Online]. Available: https:https://ozlabs.org/ rusty/ln-deploy-draft-01.pdf (Accessed: Dec. 29, 2021).

[41] C. Grunspan and R. Pérez-Marco, "Ant routing algorithm for the lightning network," 2018, *arXiv:1807.00151*.

Yanjiao Chen (Senior Member, IEEE) received the B.E. degree in electronic engineering from Tsinghua University in 2010, and the Ph.D. degree in computer science and engineering from the Hong Kong University of Science and Technology in 2015. She is currently a Professor with Wuhan University, China. Her research interests include spectrum management for femtocell networks, network economics, network security, and quality of experience of multimedia delivery/distribution.

Yuyang Ran received the B.E. degree in computer science from Wuhan University, Wuhan, China, in 2018, where she is currently pursuing the master's degree with the computer school. Her research interests include blockchain and spectrum auction.

Jingyue Zhou is currently pursuing the degree in computer science with the Computer School, Wuhan University. Her interests include blockchain, cyber security, and computer networks.

Jian Zhang received the B.S., M.S., and Ph.D. degrees in computer architecture from Wuhan University in 1998, 2001, and 2006, respectively. He is currently a Professor with Computer School, Wuhan University. His research interests include cloud computing, Internet of Things, and big data.

Xueluan Gong received the B.S. degree in computer science and electronic engineering from Hunan University in 2018. She is currently pursuing the Ph.D. degree with the School of Computer Science, Wuhan University. Her research interests include network security and AI security.