Real-Time Recursive Routing in Payment Channel Network: A Bidding-based Design

Jiayuan Liu*, Canhui Chen*, Lulu Zhou† and Zhixuan Fang*‡
*IIIS, Tsinghua University, Beijing, China
†Department of Computer Science, Duke University, USA
‡Shanghai Qi Zhi Institute, Shanghai, China

***Abstract*—Payment Channel Network (PCN) is proposed as a promising layer-two solution to tackle the scalability problem of current blockchain systems, which allows the two transacting parties to perform off-chain transactions through their established payment channel. For the transacting parties who are not directly connected, PCN allows them to route the transaction through some intermediate nodes with sufficient balance. Designing an efficient routing protocol is one of the most important and challenging problems in improving the performance of PCN. To tackle this challenge, we propose Real-Time Recursive Routing (RTRR), an efficient routing algorithm that can achieve a short routing time with strong privacy protection and high flexibility in the dynamic scenario. In addition, we investigate the bidding process in RTRR and derive the equilibrium strategy, which implies that the proposed protocol prefers to route the transaction through the nodes with a higher success rate, contributing to a better performance. Both the theoretical analyses and the empirical experiment results demonstrate the high efficiency of RTRR.**

***Index Terms*—payment channel network, routing, pricing**

I. Introduction

Blockchain systems are widely used in cryptocurrencies, Internet of things, and other applications in that they are decentralized, hard to tamper with, and highly reliable [1]. However, with their increasing popularity, the scalability problem becomes a primary concern ([2], [3]). Current solutions for blockchain scalability can be classified into two directions (see [4] and references therein). The first direction focuses on layer-one on-chain optimization methods such as segregated witness [5], block compression [6], and sharding [7], [8]. Layer-one methods usually improve the system performance by redesigning the protocol on the main chain, which may sometimes be hindered by the complexity of design, as well as protocol compatibilities. The other direction focuses on layer-two off-chain methods such as off-chain verification [9], off-chain smart contract [10], and cross-chain technology [11].

As a fundamental building block of many layer-two applications, the mechanism of payment channel enables safe and frequent transaction transfers between two users without broadcasting to the blockchain. Instead of establishing channels between every two users, a more efficient solution is to establish a Payment Channel Network (PCN) among payment channel users so that funds can be transferred from the sender to the receiver through intermediate nodes on the network. Therefore, a problem arises: how to select intermediate nodes to ensure successful and efficient transaction transfer, i.e., routing.

The Lightning Network [3] and the Raiden network [12] are among the most famous and widely-used payment channel networks in the current blockchain systems, both of which are based on source routing [13]. In source routing, the sender specifies the relay route for each transaction according to the locally-cached global information, including the network topology, the capacity of each channel in the network, etc. Recent work on Spider [14] and Flash [15] is also based on the source routing method.

One of the key concerns in source routing is the requirement of global knowledge for all nodes, which leads to two major problems. First, since global information of the network can be acquired by every node in the network, the privacy of nodes cannot be well preserved. Second, source routing may not perform well in practical scenarios such as Internet of things (IoT) or sensor network. In these scenarios, the communication capacity is limited; the nodes (e.g., IoT devices) have limited computing resources and may join and leave the network all the time, leading to a fragile dynamic network of frequently changing network topology and channel balance ([16]–[18]). In this case, the request for consistent updates of global network information imposes much pressure on network communication and a huge processing load on the nodes in the network. It would be impractical for all nodes to keep track of all changes in the network. Thus, source routing could induce a dilemma for these dynamic networks: *The outdated global information could lead to multiple retries of routing, degenerating the system performance, while the frequent update of global network information among nodes could lead to congestion and a direct performance degeneration.*

To tackle this problem, we propose a routing scheme, Real-Time Recursive Routing (RTRR), where each node only needs to interact with its neighboring nodes without accessing global information. Different from the source routing, where the transaction initiator is totally responsible for finding a path and relaying the transaction, RTRR adopts a more flexible and decentralized routing approach, where the subsequent node is chosen recursively by outsourcing during the relay process. Furthermore, we develop an auction model to investigate the dynamic pricing in the outsourcing process. We propose

Jiayuan Liu and Canhui Chen contribute equally.
Corresponding author: Zhixuan Fang at IIIS, Tsinghua University (Email: zfang@mail.tsinghua.edu.cn).

TABLE I
COMPARISON WITH OTHER ROUTING SCHEMES.

		RTRR	Traditional Source Routing[a]	Spider	Landmark-Routing	DHT
Privacy	Local information only	✔	✗	✗	✔	✔
	Broadcast-free	✔	✗	✗	✗	✔
	Decentralized routing	✔	✔	✔	✗	✔
Performance	Dynamic pricing	✔	✗	✗	✗	✗
	Routing time	Fast	Medium	Medium	Fast	Fast
	Cost for topology change	Low	High	High	Low	High
	Healthy balance	Medium	Low	High	Low	Low

[a]includes Lightning Network, Raiden Network, shortest-path, min-cost, water-filling, etc.

a variant of HTLC protocol [3], the HTLC-bid protocol, that incorporates a bidding process into the payment channel transfer to facilitate the outsourcing auction. Theoretical analysis shows that the proposed mechanism adapts to network changes with dynamic pricing and helps to improve the system performance. To validate the proposed routing scheme, we also conduct intensive experiments, on both synthetic and real-world network topologies. Experiment results demonstrate the high performance of RTRR.

The main contributions are listed as follows.

- We propose a routing scheme in PCN, Real-Time Recursive Routing (RTRR), which simultaneously supports distributed routing and dynamic pricing with only local information and achieves strong privacy protection. In addition, to facilitate a secure auction-based routing process, we design the HTLC-bid protocol, a variant from the current HTLC in PCN.
- We characterize the routing process in PCN and show that RTRR can achieve a shorter transaction routing time than the source routing algorithm. In addition, we investigate the bidding process in RTRR and derive the equilibrium strategy, which implies that the RTRR prefers to route the transaction through the nodes with a higher success rate, contributing to a higher performance of RTRR.
- We conduct intensive experiments using the data from Lightning Network to show the performance advantage of RTRR. Experiment results show that RTRR can route a transaction with less time and a lower transaction fee compared with the existing protocols.

In the following sections, we first formally give our model of PCN, routing scheme, and contract design with security guarantee in Section III. In Section V, we analyze the performance of our scheme. In Section IV, we construct a model for self-interested bidding strategy. In Section VI, some experiment settings and results are presented showing the performance of our method compared with several existing ones. Finally, there are some discussions and conclusions.

II. Related Work

Prior work has proposed several routing schemes that are used in some existing payment channel networks (PCNs). For example, Lightning Network [3], a PCN in Bitcoin [19], and Raiden Network [12], an off-chain network in Ethereum [20], both of which serve as tentative solutions to the scalability problem of cryptocurrency. Both Lightning Networks and Raiden Networks use the source routing mechanism, i.e., the transaction initiator is responsible for finding the complete transaction route, and specifying the exact intermediate nodes for the relay. Such source-based routing scheme is efficient when information such as network topology is accurate and up to date. However, in a dynamic network environment, nodes may go offline at any time, and the payment channel may not be available due to insufficient balance.

Recently, Spider [14] is proposed as another source routing scheme in PCN. It divides transactions into small packages, and routes in a package switching network according to the solution to a local optimization problem that maximizes the network throughput and maintains the channels at a balanced state. Another source routing scheme, Celer [21], also tries to keep the channels balanced by adopting a network flow routing scheme cRoute where each node runs the optimization algorithm to find the route according to the network congestion condition in its vicinity. However, [4] points out that these schemes may suffer from scalability issue due to the complex local computation overhead, especially when the network is changing frequently.

In addition to source routing, there are many new schemes based on local information. Flare [22] utilizes Distributed Hash Table (DHT) to find routes in PCN, but it does not support network topology changes [4]. SilentWhispers [23] adopts the landmark routing transaction routing scheme, where there are several supernodes dedicated as landmarks through which transactions are relayed. To reduce the computational overheads of SilentWhispers, SpeedyMurmurs [24] adopts an embedded routing scheme and uses the spanning-tree structure to maintain the network topology. However, SpeedyMurmurs does not consider the balance on the payment channels, which also causes inefficiency in the dynamic PCNs [4].

Previous routing schemes usually rely on one of the following assumptions, e.g., updated information, static network topology, or centralized source routing. As a comparison, we show in Table I that our proposed Real-Time Recursive Routing (RTRR) simultaneously supports distributed routing and dynamic pricing with only the local information, achieves strong privacy protection, and well adapts to topology changes.

TABLE II
DESCRIPTION OF COMMONLY-USED NOTATIONS.

Variable	Description
G	the PCN modeled by a graph
E	the set of edges in graph G
V	the set of nodes in graph G
s	the source node of a transaction
t	the destination node of a transaction
μ	the value of a transaction
f	the transaction fee in RTRR
d	the security deposit in RTRR
τ	the auction phase timeout in RTRR
T_b	the bidding lock time in RTRR
T_t	the transfer lock time in RTRR
$p_u[t]$	the success rate of transactions through node u to node t

III. Real-Time Recursive Routing

In this section, we first introduce the PCN model, then describe the way RTRR routes transactions, and finally explain how security is guaranteed in this routing scheme.

A. PCN Model

Payment Channel Network (PCN) builds upon bidirectional payment channels and each payment channel connects two parties. Both parties of a payment channel put a certain amount of money in escrow. If one party wants to send money to another, then the two parties reassign the distribution of the total escrow by cryptographically signing a message establishing a joint agreement of the new balances on the two sides.

A PCN can be modeled by a directed graph $G(V, E)$ where V is the set of nodes, i.e. participants of PCN, and E is the set of directed edges representing the payment channels. Note that each bidirectional payment channel between two nodes u, v is represented by two directed edges in G, edge (u, v) and edge (v, u). Each edge $e = (u, v)$ is associated with a value b_e representing the remaining balance on the channel from u to v, i.e., at most b_e amount of value can be relayed from u to v. The deposited value b_e will be secured in the channel with the signatures of both parties.

To successfully relay a transaction $tx(s, t, \mu)$ from s to t with value μ, a routing scheme needs to find a route in PCN from s to t where each node on the route is currently online and each edge on the route has enough remaining balance for value μ. During the routing process, the nodes on the routing path will deposit the transaction value and the corresponding transaction fee on the Hash Time Lock Contracts (HTLCs) [3] of their related channels, these values will be locked until the transaction succeeds or timeout T_t expires. If a transaction succeeds, then for each edge $e = (u, v)$ on the selected relay route, its balance b_e will be updated to $b_e - \mu$ and the balance $b_{e'}$ on the reverse direction edge $e' = (v, u)$ will be updated to $b_{e'} + \mu$. Otherwise, if timeout T_t expires, the deposit value locked in HTLCs will be unlocked and can then be withdrawn by the nodes.

B. Real-time Recursive Routing and HTLC-bid

Unlike the source routing where the source node is responsible for specifying a path to relay the transaction, we utilize out-sourcing in RTRR where each node on the transaction route selects one of its neighbors as the next node to relay, and the selection process goes on recursively from the source to the destination. In RTRR, each node is only required to communicate with its neighbors, i.e., the nodes it has established payment channels with. As a result, information about the local network is enough for our protocol, which substantially protects the privacy of each node in the network.

To ensure security, we design a modified version of HTLC (HTLC-bid), which enables the implementation of both the bidding lock and the transfer lock, as well as the security deposit, i.e. the amount of compensation the bidder needs to pay if it is selected to relay the transaction but fails.

In RTRR, each node v has the following three actions.

1) ***Notification****:* In this method, node v works as an auctioneer which starts an auction and requests all its neighbors to bid a transaction fee before a timeout τ for the current transaction tx with destination t and value μ. Specially, if the destination node t is one of v's neighbors, then the routing process successfully ends and t directly sends back to v the preimage R of a hash value $hash(R)$ as a proof of success. The hash value is set in advance by the source node and its preimage R is revealed only to the destination node before routing starts [3].

2) ***Bidding****:* When receiving a fee-bidding invitation from its neighboring node u, node v decides whether to participate in bidding or not. If it decides to participate, it proposes a bid (i.e. the required transaction fee) f_v based on its local information. It then constructs an HTLC-bid for u containing the transfer lock time T_t and value $\mu + d_v + f_v$ that will be locked in HTLC if signed by both parties. Here, d_v is the security deposit set to be a fixed portion of f_v, which will be paid to u as compensation if v fails to relay tx to the destination. Node v signs on the contract and sends it to the preceding node, agreeing to lock the amount of value d_v on its side for bidding lock time T_b where $\tau \leq T_b \ll T_t$. This means the contract is valid only if selected and signed by u within T_b, and otherwise, the value d_v is unlocked after T_b passes and can immediately be used to bid for other transactions. If node v decides not to bid, v would respond NO-ACK to the principal, for example when it is unprofitable for v to relay this transaction or v has already been found impassable for the current transaction in previously failed relays.

3) ***Outsourcing****:* Having received at least one responding contract from its neighbors, v chooses the node w with the smallest bid to relay the transaction by accepting the corresponding contract sent by w. It agrees to lock the transaction value μ and w's required fee f_w into the HTLC for a transfer lock time T_t, and meanwhile the security deposit d_w continues to be locked on w's side for T_t. Node v sends the contract signed by both parties to w, it is then w's responsibility to complete the remaining routing of tx by calling its *Notification*

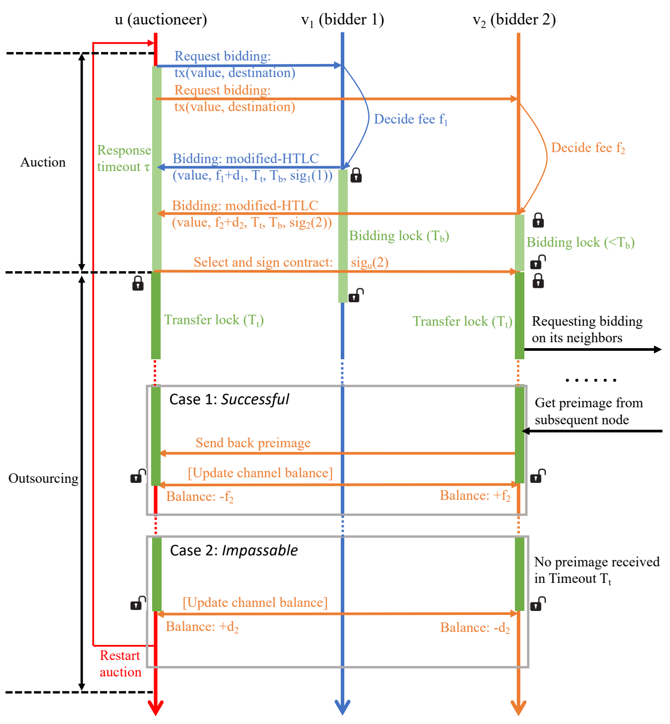

Fig. 1. Flow chart of RTRR.

method, repeating the bidding process with its neighboring nodes.

- *Successful*: Node v relays the transaction successfully if it receives the preimage from subsequent node within transfer timeout T_t. Node v then sends the preimage to its preceding node and gets all deposit value in HTLC $\mu + d_v + f_v$. As a result, the preceding node pays fee f_v to relay value μ forward to its selected bidder v, while node v withdraws its security deposit d_v and earns f_v.
- *Impassable*: Node v is impassable for transaction tx if either auction timeout τ expires but none of v's neighbors responds to the bidding request, or transfer lock time T_t expires but v has not received preimage from the subsequent node. In these cases, v's preceding node gets all deposit value in HTLC $\mu + d_v + f_v$, unlocking $\mu + f_v$ on its side and meanwhile earning d_v from v as compensation for the failure. The preceding node will then restart a bidding process by calling its *Notification* method. Specially, if v is the source node, the transaction tx fails immediately.

RTRR starts routing by source node's calling its *Notification* method, and one iteration of RTRR is shown in Fig. 1.

IV. Bidding and Pricing Strategy

In this section, we demonstrate that RTRR enables dynamic pricing and shows how PCN participants can leverage this property to strategically bid for different transactions.

We model the nodes as rational agents and analyze their bidding behaviors. In each iteration of RTRR, the auctioneer node selects a neighbor with the least fee to outsource the transaction after receiving bids from the neighbors. Bidder v proposes its fee according to the empirical success rate for transactions sending through v to destination node t based on v's local history. The bidding value incorporates the risk of compensation in case the transaction fails. Moreover, each self-interested bidder will bid strategically, taking into consideration the fee its competitors may propose because only the lowest bid will be selected. If it proposes a bid higher than any one of its competitors', then it will not be chosen, while if the bid is too cheap, the node may risk losing money from relaying this transaction.

We formalize an auction model to characterize this bidding process. Consider the situation where a transaction tx with destination node t reaches node u. Then, node u acts as the auctioneer which calls upon its neighbors to bid for the transaction. We consider symmetric bidding strategy for all K neighbors of u, i.e., $v_1, ...v_K$. Each bidder has a private signal $p_{v_i}[t]$ indicating the probability that it can successfully deliver the transaction to the destination. Assume that all $p_{v_i}[t]$'s are i.i.d. random variables following the same continuous distribution $F(\cdot)$ and assume that the distribution is common knowledge among $v_1, ...v_K$. Bidder v_i bids $f_{v_i}[t]$ as its fee for this transaction tx with destination node t. Define the security deposit to be $\alpha f_{v_i}[t]$ which v_i needs to pay as compensation if it is selected to relay but fails. Auctioneer u aims to select the bidder with the least $f_{v_i}[t]$. This process is similar to a procurement auction [25], where multiple sellers bid their fees to satisfy the demand of one buyer.

Theorem 1. *A rational node v_i will bid*

$$f_{v_i}[t] \propto (1 - F(p_{v_i}[t]))^{K-1}$$

where K is the number of neighbors in our model, $p_{v_i}[t]$ is the success probability for the current transaction from node v_i to destination node t, and $F(\cdot)$ is the distribution for $p_{v_i}[t]$.

Proof. According to the RTRR, when neighbor v_i is selected by auctioneer u, its expected utility is

$$\mathbb{E}[U_i^{(\text{selected})}] = p_{v_i}[t] f_{v_i}[t] + (1 - p_{v_i}[t])(-\alpha f_{v_i}[t])$$

in which $f_{v_i}[t]$ is the fee v_i earns if the transaction succeeds (with probability $p_{v_i}[t]$ it succeeds), and $\alpha f_{v_i}[t]$ is the compensation v_i has to pay to u if the transaction fails (with probability $1 - p_{v_i}[t]$ it fails). v_i's expected utility equals 0 if it is not selected by u. Thus, v_i's overall expected utility is

$$\begin{aligned}\mathbb{E}[U_i] =& \mathbb{E}[U_i^{(\text{selected})}] \cdot \Pr[v_i \text{ is selected}] \\ =& \Big(p_{v_i}[t] f_{v_i}[t] - (1 - p_{v_i}[t]) \alpha f_{v_i}[t]\Big) \\ & \cdot \Pr[f_{v_j}[t] > f_{v_i}[t], \forall j \neq i].\end{aligned}$$

As we consider symmetric bidding strategy, each neighbor v_i decides its bid $f_{v_i}[t]$ according to its estimated winning probability $p_{v_i}[t]$ by a common function $g(\cdot)$, i.e., $f_{v_i}[t] =$

$g(p_{v_i}[t])$. Further assume function $g(\cdot)$ is differentiable and strictly monotone. Then,

$$\begin{aligned}\mathbb{E}[U_i] =& \Big(p_{v_i}[t]f_{v_i}[t] - (1-p_{v_i}[t])\alpha f_{v_i}[t]\Big) \\ &\cdot \Pr[g(p_j) > g(p_{v_i}[t]), \forall j \neq i] \\ =& \Big((p_{v_i}[t](1+\alpha)-\alpha)f_{v_i}[t]\Big)\Big(1-F(g^{-1}(f_{v_i}[t]))\Big)^{K-1}.\end{aligned}$$

Note that a tie $f_{v_i}[t] = f_{v_j}[t]$ can only occur with infinitesimal probability because g is monotone and $p_j(\forall j)$ can be updated to any real number when the history information accumulates.

Then, we derive the equilibrium strategy in bidding. To reach a maximum of expected utility, we calculate the stationary point of the utility function with respect to the bidding strategy. Its first order derivative is

$$\begin{aligned}\frac{\partial \mathbb{E}[U_i]}{\partial f_{v_i}[t]} =& (p_{v_i}[t](1+\alpha)-\alpha)\Big(1-F(g^{-1}(f_{v_i}[t]))\Big)^{K-1} \\ &+ (p_{v_i}[t](1+\alpha)-\alpha)f_{v_i}[t]\Big(1-F(g^{-1}(f_{v_i}[t]))\Big)^{K-2} \\ &\cdot (K-1)f(g^{-1}(f_{v_i}[t]))\frac{1}{g'(p_{v_i}[t])}\end{aligned}$$

where $f(\cdot)$ is the p.d.f. with respect to c.d.f. $F(\cdot)$ and $g'(\cdot)$ is the derivative of strategy function $g(\cdot)$. Then, $\frac{\partial \mathbb{E}[U_i]}{\partial f_{v_i}[t]} = 0$ is equivalent to

$$\gamma_i(1-F(p_{v_i}[t])) + \gamma_i g(p_{v_i}[t])(K-1)f(p_{v_i}[t])\frac{1}{g'(p_{v_i}[t])} = 0$$

where $\gamma_i \equiv p_{v_i}[t](1+\alpha)-\alpha$. This is in fact a linear differential function with variable coefficients. By solving this differential equation, we can derive the equilibrium bidding strategy for a rational bidder v_i for this auction with a bidder-specific constant $C_{i,\text{auc}}$ as

$$f_{v_i}[t] = g(p_{v_i}[t]) = C_{i,\text{auc}}(1-F(p_{v_i}[t]))^{K-1}.$$

The proof is thus completed. ■

From Theorem 1, if a transaction that goes through u to t succeeds, then node u's estimation for the success rate of transactions with destination t will increase according to Bayes' rule, and from the pricing model, u will bid a lower value next time for transactions with destination t, and vice versa.

Corollary 2. *For a rational bidder v_i, its optimal bidding price $f_{v_i}[t]$ is decreasing in $p_{v_i}[t]$.*

Corollary 2 shows that the bidder with the least proposed transaction fee is exactly the bidder with the highest transaction success rate.

By proposing different prices for each bidding request, the nodes in RTRR are naturally allowed to dynamically adjust their transaction fees according to the updated information. On the contrary, source routing schemes either fix the fee at a certain proportion to the transaction value or require extra commitments, broadcasts, and delays when fee rate changes.

With dynamic pricing, each node in RTRR has greater flexibility in pricing and prefers to relay to a subsequent neighbor with a higher success probability, thus contributing to a higher transaction success rate.

V. Performance Analysis

In this section, we analyze the performance advantage of RTRR over the source routing schemes such as the routing scheme used in Lightning Network [3] by comparing the expected time needed to find a passable path.

Source routing schemes send each transaction tx to its destination node through a path specified by the source node s. However, this proposed path may be unavailable due to containing either channels with insufficient balance or offline nodes. When encountering such situations, the proposed route will fail. Then, tx needs to be resent starting from s on another route again specified by s. This process repeats until a passable path is found or timeout is reached. While in RTRR, every intermediate node on the path can resume the routing process when the previously selected neighbor fails, thus reducing overheads by not having to restart the relay from the source node.

To investigate the routing time of our protocol, we model the network as a K-regular graph. Let N be the total number of nodes in the network. For simplicity, we assume that all routes on PCN for a transaction $tx(s,t,\mu)$ are of the same length L and all nodes have the same degree K.

In RTRR, the time for the auction phase is upper bounded by τ. In practice, we can choose a small τ such that $\tau \ll T_t$, as bidder nodes only send their bids to the auctioneer in the auction phase, while establishing HTLC in the outsourcing phase requires complicated cryptographic signatures and verifications. In addition, the outsourcing phase in RTRR consists of the same transaction relay process and HTLC termination process as in the protocol of existing payment channel networks, such as Lightning Network. Therefore, we assume that the time for the interaction between two nodes in RTRR is equivalent to that of the traditional source-based routing algorithm. Based on this assumption, the performance analysis goes as follows.

The search process of RTRR is similar to a depth-first search (DFS) because it proceeds through one path until an endpoint node if no failure occurs, and retrogrades to the preceding node if fails at some point. Because DFS search trajectory forms a search tree, it is natural to model the network structure by a $(K-1)$-ary tree of depth $L+1$, where each internal node of the tree has exactly K neighbors. The process of RTRR corresponds to a DFS search starting from the root node and going down through edges to one of the leaf nodes of the depth-$(L+1)$ tree, with each edge being available with probability θ. An available edge means that the two ends of the corresponding channel are both online and the remaining balance on it is enough for the transaction. Encountering failure, RTRR retrogrades to the preceding node while source routing restarts from the source node. We prove the performance advantage of RTRR in the following theorem.

Theorem 3. *The transaction routing time in RTRR T_{rec} is at least 100η% shorter than that of the source routing protocols T_{src}, where*

$$\begin{aligned}\eta \equiv& \frac{T_{src}-T_{rec}}{T_{src}} = 1-\frac{T_{rec}}{T_{src}} \\ \geq& 1-\frac{2}{L-1}\Big((1-(K+1)(1-\theta)^K + K(1-\theta)^{K+1}) \\ &-\frac{\theta(1-\theta)^K K}{1-(1-\theta)^K} + \frac{(1-\theta)^K}{(1-(1-\theta)^K)L}\Big). \qquad (1)\end{aligned}$$

The proof is shown in Appendix A in the online technical report [26]. Theorem 3 shows that RTRR is more efficient compared to the source routing algorithm, achieving a shorter transaction routing time.

We analyze the asymptomatic case when the number of participants $N \to \infty$ and the number of neighbors $K \to \infty$.

Corollary 4. *When $N \to \infty$ and $K \to \infty$, the transaction routing time in RTRR is $100\frac{L-3}{L-1}$% shorter than that of the source routing protocols asymptotically,*

$$\lim_{K\to+\infty} \eta \geq \frac{L-3}{L-1}. \qquad (2)$$

Note that $L = 2$ indicates a highly-centralized system and $L = 1$ indicates a fully-connected graph, neither of which is idealized in a distributed payment system. When $L \geq 3$, the lower bound for η in the limit case is non-negative, and when $L \gg 1$ our scheme shows a significant performance advantage.

VI. Experiments

In this section, we first introduce the settings and environments used in our experiments, then show the results and interpretations.

A. Experiment Settings and Environments

We use python SimPy framework [27] to do discrete-event simulation and use NetworkX python library [28] to construct directed graphs to simulate PCNs.

We implement our protocol on two types of network topology: the synthetic structure based on Erdős–Rényi graph [29], and the network topology from Lightning Network [3]. Then, we use the data from Lightning Network [3] to demonstrate the performance advantage of RTRR on typical PCN structures. Note that among all the 1,917 nodes in the Lightning Network, there exist some "supernodes", nodes with degree larger than 100, which are well connected in the network. For example, there exists a node with 858 neighbors, and exists two nodes together connecting more than 1,000 nodes in the network. With an average degree of 424, the top 10 supernodes directly connect with more than 80% of the nodes in the network. These supernodes are serving as trading hubs for PCN transaction relays. The existence of these stable and well-connected nodes could be due to the requirements for watchtowers in the Lightning Network ([30], [31]). Such network topology may not reflect more general decentralized and dynamic networks, e.g., networks of IoT devices or sensors [16]. Thus, we also conduct experiments on a pruned Lightning Network topology without supernodes, which is derived by eliminating the 58 supernodes from the original Lightning Network.

To leverage the property of allowing dynamic pricing, RTRR decides transaction-specific fee by the empirical estimation of success rate based on the previous transactions that went through it. Specifically, in RTRR, each node v maintains local counters $N_{\text{succ},v}[t], N_{\text{fail},v}[t]$ for every other node t in the network, which counts the number of successful and failed relays that goes through the current node v with destination node t, respectively. Node v calculates the empirical success rate for transactions to a certain destination node t by

$$p_v[t] = \frac{N_{\text{succ},v}[t]}{N_{\text{succ},v}[t] + N_{\text{fail},v}[t]}. \qquad (3)$$

Each successful (failed) relay will increment the corresponding counter $N_{\text{succ},v}[t]$ ($N_{\text{fail},v}[t]$) by one.

Apply the result in Theorem 1 and get

$$f_v[t] = g(p_v[t]) = C_{v,t}(1-F(p_v[t]))^{K-1}. \qquad (4)$$

Take differentiate on both sides of equation (4) to get the amount of fee update $\Delta f_v[t]$ with respect to the change of transaction success probability $\Delta p_v[t]$,

$$\Delta f_v[t] = C_{v,t}(K-1)(1-F(p_v[t]))^{K-2}F'(p_v[t])\Delta p_v[t]. \quad (5)$$

From equation (4) and equation (5), we can derive

$$\Delta f_v[t] = f_v[t]\frac{K-1}{1-F(p_v[t])}F'(p_v[t])\Delta p_v[t]. \qquad (6)$$

For each update, we change the fee by $\Delta f_v[t]$ according to equation (6), where $\Delta p_v[t]$ is the difference between the values of $p_v[t]$ after and before the counters $N_{\text{succ},v}[t], N_{\text{fail},v}[t]$ are updated in such iteration. As a result, a succeeded (failed) relay leads to an increase (decrease) in transaction fee.

To model dynamic network topology, which occurs in wireless networks and sensor networks, we consider that for each coming transaction, each node has a fixed probability to be offline and does not participate in relaying such transaction. This probability is set to be 35% in the experiments. In each run of the experiments, we use 2,000 transactions. We assume that the interaction, including the communication and operations on HTLC of the established payment channel, between two directly connected nodes costs one time unit. And the timeout for each transaction is set to be 300 time units.

We compare our proposed method with the following baselines, which are common source routing schemes and also used as baselines in recent works ([14], [15]).

1) Shortest-Path Routing (SP): This scheme sends each transaction to its destination node through a path with the shortest length on its local cached network. Both Lightning Network and Raiden Network use this scheme in relaying transactions.

2) Min-Cost Routing (MC): This scheme sends each transaction to its destination node through a path with the minimum transaction fee on its local cached network.

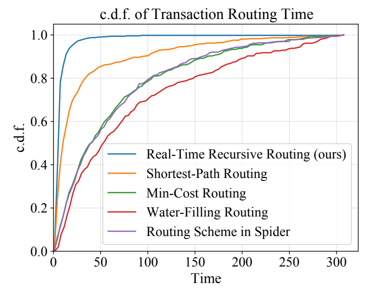

Fig. 2. The c.d.f. of transaction routing time in random graph with 1870 nodes.

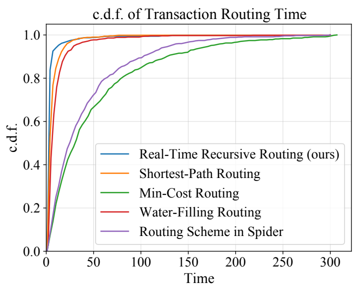

Fig. 3. The c.d.f. of transaction routing time in Lightning Network.

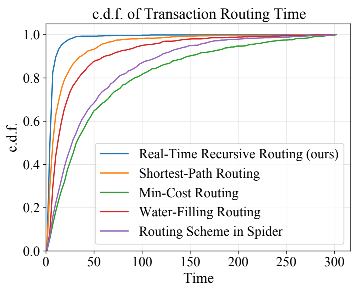

Fig. 4. The c.d.f. of transaction routing time in Lightning Network without supernodes.

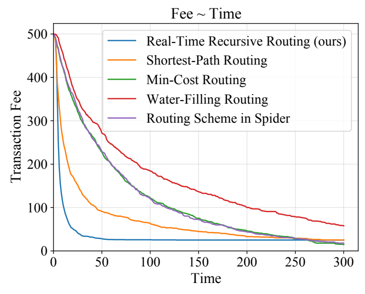

Fig. 5. Average transaction fee over time on random graph.

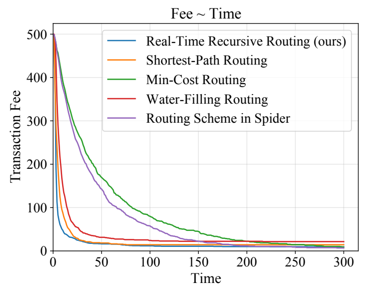

Fig. 6. Average transaction fee over time on Lightning Network.

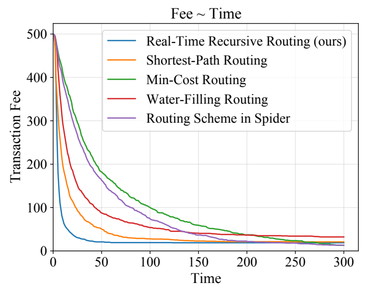

Fig. 7. Average transaction fee over time on Lightning Network without supernodes.

3) Water-Filling Routing (WF): This scheme sends each transaction to its destination node through a path with the maximum bottleneck capacity on its local cached network, where the bottleneck capacity of a route is the minimum capacity among all channels on the route.

4) Routing Scheme in Spider: [1] This scheme characterizes the important channel balancing feature in Spider's routing scheme. To achieve high performance, Spider imposes a higher cost to routes that would cause imbalance or would cause congestion [14] and sends each transaction through an available path with minimum adjusted total cost. This is the most important feature of Spider's routing scheme. As the complete version of Spider also includes transaction division, packetization, channel balancing optimization, etc., we only compare our protocol with this routing scheme of Spider.

B. Experiment for Performance Advantage

Fig. 2, Fig. 3, and Fig. 4 show the cumulative distribution of transaction routing time in random graph, Lightning Network, and Lightning Network without supernodes. These figures depict the performance advantage of RTRR over baselines in terms of the time needed to find an available path.

Fig. 5, Fig .6, and Fig. 7 demonstrate the average transaction fee over time, from which we can find that RTRR can achieve a higher performance, i.e., finding a path with a lower fee in a shorter time compared to the existing protocols.

It is not surprising that in Fig. 3 and Fig. 6, where supernodes exist, our protocol only outperforms baselines at a moderated level since all baselines prefer to route through the supernodes, which is already nearly optimal. Fig. 2, Fig. 4, Fig. 5, and Fig. 7 depict more remarkable performance advantage of RTRR in more decentralized network structures. The Min-Cost routing method and the routing scheme in Spider achieve fewer average transaction fees but cost more time compared with RTRR. A lower average fee is because Min-Cost routing and the routing scheme in Spider prefer to find the route with the least cost, while longer routing time is due to encountering more failures on a comparatively longer route in the dynamic network. Summarizing these results, RTRR works well on real PCN structures and performs substantially better on decentralized and dynamic networks.

[1]Original Spider code is not open-source, so we extract the key idea from Spider's routing scheme.

VII. Discussions

A. Privacy Protection

In RTRR, each node only needs to know the local information (the information of its neighbors and its adjacent payment channels) because, in RTRR, each node only needs to interact with its neighbors, without acquiring extra information from other nodes. While in source routing schemes, the source node needs to know the global information such as the fee and capacity of all channels in the network in order to specify the optimal path. Furthermore, to adapt to the dynamic environment, in source routing, announcement messages are encouraged to be broadcast to all nodes in the network when some node becomes offline or changes its fee rate. Thus, compared to the traditional source routing protocol such as the one used in Lightning Network, RTRR achieves stricter privacy protection.

B. Decentralization Tendency

The traditional source routing protocol, such as the routing method based on the shortest path, aims to find an optimal path with less transaction fee and a higher success rate. Thus, the source routing protocol prefers to route through some supernodes that are well connected in the network. Indeed, the current PCNs, such as the Lightning Network, suffer from the centralization problem caused by these supernodes. In RTRR, the outsourcing process prefers to relay the transaction through the nodes with lower bidding fees, which can be proposed by some "small" nodes with fewer neighbors instead of the supernodes. Such a design tends to degrade the centralization of the PCN.

C. Security

RTRR has the same guarantee of security as the original HTLC because HTLC-bid requires the same commitment procedures between two parties of the contract as the ones in HTLC. Due to the security deposit, malicious behaviors such as defecting in either the bidding stage or relay stage will lead to the punishment, i.e., the defector will forfeit its security deposit.

VIII. Conclusion

In this work, we propose Real-Time Recursive Routing (RTRR), an efficient routing protocol that simultaneously supports distributed routing and dynamic pricing with only local information and achieves strong privacy protection. Both the theoretical analyses and the intensive experiment results show that RTRR can route a transaction in a shorter time with a lower transaction fee compared to the existing protocols. One of the future works is to design a more efficient bidding strategy based on the local information paradigm of RTRR scheme.

References

[1] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*. Ieee, 2017, pp. 557–564.

[2] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer *et al.*, "On scaling decentralized blockchains," in *International conference on financial cryptography and data security*. Springer, 2016, pp. 106–125.

[3] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[4] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Off the chain transactions." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 360, 2019.

[5] E. Lombrozo, J. Lau, and P. Wuille, "Segregated witness (consensus layer)," *Bitcoin Core Develop. Team, Tech. Rep. BIP*, vol. 141, 2015.

[6] D. Ding, X. Jiang, J. Wang, H. Wang, X. Zhang, and Y. Sun, "Txilm: Lossy block compression with salted short hashing," *arXiv preprint arXiv:1906.06500*, 2019.

[7] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 931–948.

[8] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[9] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," *arXiv preprint arXiv:1908.04756*, 2019.

[10] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1353–1370.

[11] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, vol. 21, pp. 2327–4662, 2016.

[12] R. Network, "What is the raiden network?" *URL:https://raiden.network/101.html.*

[13] C. A. Sunshine, "Source routing in computer networks," *ACM SIGCOMM Computer Communication Review*, vol. 7, no. 1, pp. 29–33, 1977.

[14] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 777–796.

[15] P. Wang, H. Xu, X. Jin, and T. Wang, "Flash: efficient dynamic routing for offchain networks," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 370–381.

[16] S. L. Ullo and G. Sinha, "Advances in smart environment monitoring systems using iot and sensors," *Sensors*, vol. 20, no. 11, p. 3113, 2020.

[17] S. Li, L. D. Xu, and S. Zhao, "The internet of things: a survey," *Information systems frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[18] H. J. La, C. W. Park, and S. D. Kim, "A framework for effectively managing dynamism of iot devices," *Journal of KIISE: Software and Applications*, vol. 41, no. 8, pp. 545–556, 2014.

[19] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[20] W. Ethereum, "Ethereum whitepaper," *Ethereum. URL: https://ethereum.org [accessed 2022-05-01]*, 2014.

[21] M. Dong, Q. Liang, X. Li, and J. Liu, "Celer network: Bring internet scale to every blockchain," *arXiv preprint arXiv:1810.00037*, 2018.

[22] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An approach to routing in lightning network," *White Paper*, p. 144, 2016.

[23] P. Moreno-Sanchez, A. Kate, and M. Maffei, "Silentwhispers: Enforcing security and privacy in decentralized credit networks," in *Proc. NDSS*, 2017.

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

[25] R. R. Chen, R. O. Roundy, R. Q. Zhang, and G. Janakiraman, "Efficient auction mechanisms for supply chain procurement," *Management Science*, vol. 51, no. 3, pp. 467–482, 2005.

[26] "Online technical report," *URL: https://cloud.tsinghua.edu.cn/f/e28c3b985ed84dff9836/.*

[27] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "Sympy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017.

[28] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[29] E. Paul and R. Alfréd, "On random graphs i," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.

[30] Z. Avarikioti, O. S. Thyfronitis Litos, and R. Wattenhofer, "Cerberus channels: Incentivizing watchtowers for bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 346–366.

[31] M. Leinweber, M. Grundmann, L. Schönborn, and H. Hartenstein, "Tee-based distributed watchtowers for fraud protection in the lightning network," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2019, pp. 177–194.