Thor: A Virtual Payment Channel Network Construction Protocol over Cryptocurrencies

Qiushi Wei Dejun Yang Ruozhou Yu Guoliang Xue

***Abstract*—Payment Channel Networks (PCNs) have been proposed as a second-layer solution to the scalability issue of blockchain-based cryptocurrencies, most developed systems still lack effective strategies for further scalability solutions. Virtual payment channel (VPC) has been proposed as an off-chain technique that avoids the involvement of intermediaries for payments in a PCN. However, there is no research on how to efficiently construct VPCs while considering the characteristics of the underlying PCN. To fill this void, this paper focuses on the VPC construction in a PCN. More specifically, we propose a metric, Capacity to the Number of Intermediaries Ratio (CNIR), to consider both the capacity of the constructed VPC and the collateral locked by the involved users. We first study the VPC construction problem for a single pair of users and design an efficient algorithm that achieves the optimal CNIR. Based on this, we propose Thor, a protocol that constructs a virtual payment channel network (VPCN) for multiple pairs. Evaluation results show that Thor can efficiently construct a VPCN and outperform baseline algorithms in terms of the CNIR.**

***Index Terms*—Cryptocurrency, payment channel network, virtual payment channel**

I. Introduction

Over the past decades, we have witnessed a flourishing of cryptocurrencies as a decentralized form of currency, such as Bitcoin [1] and Ethereum [2]. However, these digital currencies have struggled to achieve large-scale usage due to significant overhead and demanding storage requirements [3]. The second-layer solution [4] lives on top of the blockchain and conducts payments off-chain. The most popular application is the payment channel network (PCN) implemented in Bitcoin's Lightning Network (LN) [5] and the state channel network (SCN) implemented in Ethereum's Raiden Network [6]. Existing studies [7]–[16] have focused on addressing issues related to routing, rebalancing, and collateral reduction.

When two users in PCN do not have a direct channel but are connected through a path of channels, they transmit payments to each other via this path, subject to the available balance in each channel. For security, a Hash Timelock Contract (HTLC) [17] is utilized, locking the balance on each channel until all channels verify the payment's outcome or until the timelock period expires [5]. Compared to locking funds on each channel through the HTLC, virtual payment channels (VPCs) offer the benefit of processing payments through a single hop, facilitating completely off-chain channel constructions, and presenting broader potential use cases. Current VPC protocols such as Perun [18] provide the smallest example of building a two-party virtual payment channel, which further reduces the interaction with the blockchain and is implemented with full concurrency using smart contracts in Ethereum with its security proven in universally composable (UC) proof [19]. This two-party VPC concept was then extended to general state channels in [20]. Dziembowski also introduces the multi-party virtual state channels, specifically enabling more than two parties to execute contracts off-chain [21].

The construction of VPCs demands collaborative effort from all intermediaries along the path consisting of payment channels. Such collaboration entails not only exchanging messages to agree on the initiation phase but also involves the intermediaries locking funds for a specified duration. Excessive intermediaries make the total collateral requirement burdensome. In addition, current protocols do not examine how to efficiently optimize the number of intermediaries or how to establish the VPCs for multiple pairs. To address the above challenges, we first define a new metric Capacity to the Number of Intermediaries Ratio (CNIR) for evaluating both the number of intermediaries and the capacity of the constructed VPC. Then we propose Thor, a protocol for constructing VPCs that guarantees maximum CNIR in the single-pair case, and efficiency for constructing VPCs in the multi-pair case. Our algorithms can be implemented as a decentralized protocol without trusted central servers. The main contributions of this paper are:

- To the best of our knowledge, we are the first to study the problem of optimal single VPC construction and multiple VPC constructions evaluated by both VPC capacity and the number of intermediaries in PCN, where users can benefit from parallel VPC constructions off-chain.
- We design an optimal VPC construction algorithm for a single pair of users. We then propose Thor, a protocol that can construct a virtual payment channel network with high CNIRs for the multi-pair case.
- Evaluation results demonstrate that Thor accomplishes the maximum CNIR of the constructed VPC in the single-pair case, and yields high CNIRs for the multi-pair case, compared to the baseline algorithms.

Organization. Section II overviews the background related to VPCs. Section III describes the system model and formulates

Wei and Yang are affiliated with Colorado School of Mines, Golden, CO 80401. Yu is affiliated with North Carolina State University, Raleigh, NC 27695. Xue is affiliated with Arizona State University, Tempe, AZ 85287. Email:{qiushiwei, djyang}@mines.edu, ryu5@ncsu.edu, xue@asu.edu. This research was supported in part by NSF grants 2008935, 2045539, and 2007083. The information reported here does not reflect the position or the policy of the federal government.

the problems. Section IV points out the challenges for the protocol design. Section V illustrates the design of protocol Thor. Section VI evaluates the related algorithms. Section VII shows the conclusion.

II. Background Overview

Ledger Payment Channel (LPC): LPCs are second-layer solutions to address the blockchain's scalability issue by conducting micropayments without committing transactions to the blockchain. LPC operates under a multisig smart contract [22] that needs multiple authentic signatures to prevent single-point failures. There are four stages in an LPC: (1) Open: Participants follow a protocol that involves depositing funds into a smart contract that will serve as collateral for the duration of the channel. The initial state is signed by all parties to set the starting point of the channel. (2) Update: Parties exchange messages that are secured with cryptographic signatures, and the states of the channels can then get updated. (3) Dispute: Once the dispute is started, the opposing party is required to respond within the challenge window. A dispute can arise due to unresponsive participants, refusal of sign, or invalid state proposals [23]. (4) Close: Participants are required to submit the channel's final, agreed-upon state on-chain, after which the locked deposits will be returned to the participants.

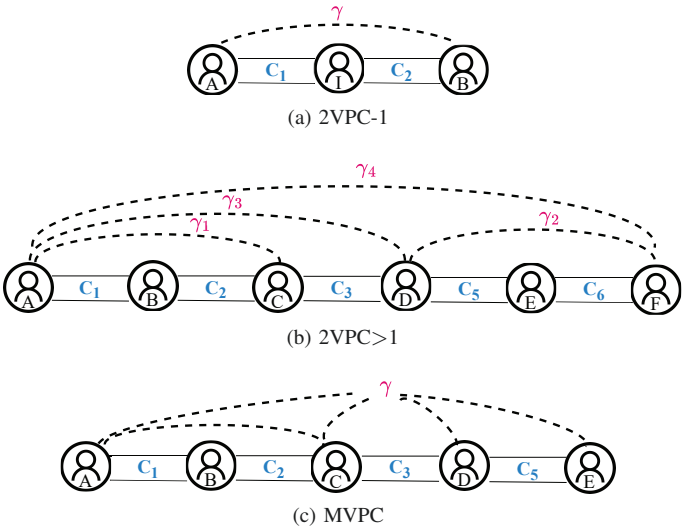


Fig. 1. Three types of VPCs

Virtual Payment Channel (VPC): There are three types of VPCs. (1) Two-party VPC with One Intermediary (2VPC-1): If the two indirectly-connected parties desire to make off-chain payments, they can either open a LPC or VPC [18]. Instead of routing payments through hash-locked transactions across an intermediary, a VPC-1 is constructed off-chain over two LPCs. For example, a VPC-1 γ is constructed with the help of I serving as the *intermediary*, as shown in Fig. 1(a). Here, the two LPCs C_1 and C_2 are the two *subchannels* of γ. A and B need to notify I of their proposal to employ her as an intermediary for γ, which is done symmetrically by A's proposal to open an instance of the special virtual payment channel contract in channel C_1, as well as B's proposal to open another instance in channel C_2. Both new instances can be seen as a copy of the opened γ. If I agrees with both proposals, γ is successfully opened and ready for future transactions between A and B. The funds that the two end users (EUs) A and B intend to put in γ as collateral will be locked from their respective original deposits at C_{AI} and C_{IB} for the duration that the γ remains active. At the same time, I needs to lock up the same collateral to represent both A and B, which means that I is ready to cover the commitments from the transactions in the VPC. (2) Two-party LPC with More Intermediaries (2VPC>1) [20]: A 2VPC>1 is constructed on a path of LPCs between the two EUs recursively regardless of the subchannel types. In Fig. 1(b), γ_1 is first built between A and C, with B as the intermediary. Then γ_2 is built on top of the two LPCs C_5 and C_6, with E serving as the intermediary. Next, γ_3 is built on γ_1 and the LPC C_3. Finally, γ_4 is built on both γ_3 and γ_2. When more intermediaries are involved, all subchannels must be created/closed before the target VPC can be opened/closed. (3) VPC with Multiple End Users (MVPCs): MVPCs [21] are constructed when EUs are more than two. A specific instance of a five-party VPC is illustrated in Fig. 1(c). The five EUs: A, B, C, D, and E, are interconnected via LPCs. B chooses not to participate in γ, so it acts as the intermediary in the 2VPC-1 between A and C. A contract instance is initialized in each subchannel as a replica of γ. The creation of MVPC is also off-chain, but it requires consensus from all EUs.

III. Network Model and Problem Formulation

In this section, we describe the network model and present our problem formulation.

A. Network Model

The payment channel network (PCN) can be modeled as a bidirectional weighted graph $G = (\mathcal{V}, \mathcal{E})$. $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of n nodes, every node v_i in the network represents a user who holds a cryptocurrency account and has at least one LPC with another user. $\mathcal{E} = \{e_{i,j} | v_i, v_j \in \mathcal{V}\}$ is the set of m directional edges, and $i \leftrightarrow j$ represents a bidirectional channel between nodes v_i and v_j. Let $\bar{\delta}_{i,j}$ denote the balance on $e_{i,j}$. Let $\delta_{i,j}$ denote the **maximum locked balance (MLB)** that v_i would offer to be locked for VPC construction on channel $e_{i,j}$. It is reasonable that $\bar{\delta}_{i,j} \geq \delta_{i,j}$, since v_i is unable to lock more balance than the original balance. For those who are not willing to participate in the VPC construction, $\delta_{i,j} = 0$. Note that the MLBs are public data, which will not expose the private balance information. Let $\mathcal{M} = \{\delta_{i,j} | \delta_{i,j} > 0, \forall e_{i,j} \in \mathcal{E}\}$ denote the set of MLBs from those nodes who are willing to participate. In addition, let $\mathcal{N} = \{(s_1, t_1), \ldots, (s_K, t_K)\}$ denote the set of K node pairs requesting VPC construction between them. A **virtual payment channel network (VPCN)** is a network of VPCs between nodes in a PCN. Thus, we leave this topic for our future work.

B. Problem Formulation

Any node pair (s, t) that does not have a LPC connecting them can choose to construct a VPC, where s and t are the

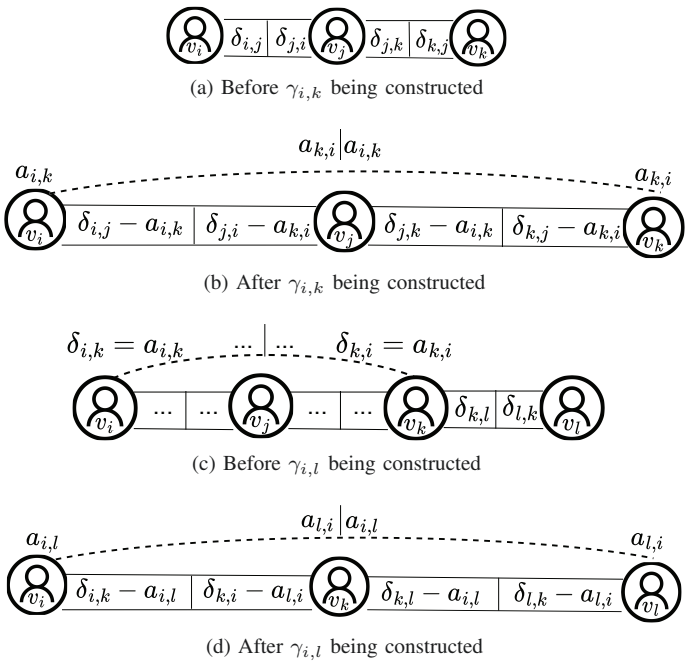
two EUs. The virtual channel type we are interested in is the two-party VPC between a node pair (s, t). Given that the PCN is bidirectional and VPC is constructed on top of a path, an **underlying ledger channel path (ULP)** P is two directed paths traversing the same nodes but in reverse directions between a node pair and it is the bottom structure of the VPCs. Let $P = \{P^{s\to t}, P^{t\to s}\}$ denote a bidirectional path $s \leftrightarrow I_1 \leftrightarrow I_2 \leftrightarrow \ldots \leftrightarrow I_d \leftrightarrow t$, where $\mathcal{I}_P = \{I_1, I_2, \ldots, I_d\}$ is the set of d intermediaries. Here $P^{s\to t}$ is the directional path from s to t: $s \to I_1 \to I_2 \to \ldots \to I_d \to t$, and $P^{t\to s}$ is the other directional path from t to s: $t \to I_d \to \ldots \to I_2 \to I_1 \to s$. We are only interested in simple paths for which the nodes in the sequence are distinct. For K pairs, we have the ULP set $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$.

The **MLB of a path**, denoted by $\delta(P^{s\to t})$ is the minimum of the MLBs on the edges along the path $P^{s\to t}$, i.e., $\delta(P^{s\to t}) = \min(\{\delta_{i,j} : \forall e_{i,j} \in P^{s\to t}\})$. Similarly, $\delta(P^{t\to s})$ denotes the MLB of the path $P^{t\to s}$. In addition, we define the maximum capacity as $c := a_{s,t} + a_{t,s}$ for the constructed VPC γ, where $a_{s,t}$ and $a_{t,s}$ represent the maximum balance that s and t can respectively contribute to γ. We define the new metric **Capacity to the Number of Intermediaries (CNIR)** as $r := \frac{c}{d}$. The rationale of introducing CNIR is two-fold: 1) a larger capacity can support more potential transactions over the constructed VPC; and 2) a fewer number of intermediaries means less collateral will be locked and less fee will be charged to the end users. Thus, it is logical to aim for high CNIR in VPC construction.

A modular recursive approach is applied to construct a VPC on a ULP that has more than one intermediary, where VPCs are constructed recursively on top of VPC or other already constructed VPCs [20]. We extend the concept of *subchannels* to any two channels (VPC or LPC) that constitute a VPC in our paper. We use Fig. 2 as an example to show the relationship between the capacity and MLBs in the recursive VPC construction process. The initial MLB values on $i \leftrightarrow j$ and $j \leftrightarrow k$ is shown in Fig. 2(a). v_i and v_k desire to construct a VPC $\gamma_{i,k}$ between them. v_i, v_j and v_k need to lock a certain amount of balances to construct $\gamma_{i,k}$ from the two subchannels $i \leftrightarrow j$ and $j \leftrightarrow k$. Therefore, the balance distribution in the subchannels changes after $\gamma_{i,k}$ is constructed, as shown in Fig. 2(b). The maximum balance $a_{i,k}$ that node v_i can put on $\gamma_{i,k}$ should be mutually determined by $\delta_{i,j}$ and $\delta_{j,k}$. Intuitively, this amount should not be more than what v_i can maximally offer ($\delta_{i,j}$) and what v_j can lock in represent of v_i ($\delta_{j,k}$). Therefore, in order to reach the maximum capacity of $\gamma_{i,k}$:

$$a_{i,k} = \min(\delta_{i,j}, \delta_{j,k}), a_{k,i} = \min(\delta_{k,j}, \delta_{j,i}) \tag{1}$$

In this case, the two subchannels $i \leftrightarrow j$ and $j \leftrightarrow k$ construct $\gamma_{i,k}$ with the maximum capacity $c = \min(\delta_{i,j}, \delta_{j,k}) + \min(\delta_{k,j}, \delta_{j,i})$. The total MLB $\delta_{i,j} + \delta_{j,i} + \delta_{j,k} + \delta_{k,j}$ in Fig. 2(b) remains the same as in Fig. 2(a). Once $\gamma_{i,k}$ is built, a new bidirectional edge $i \leftrightarrow k$ is added to the VPCN. However, the distribution of the MLB changes: the MLB on the subchannels decreases, while the MLB on the newly built VPC increases.



(a) Before $\gamma_{i,k}$ being constructed

(b) After $\gamma_{i,k}$ being constructed

(c) Before $\gamma_{i,l}$ being constructed

(d) After $\gamma_{i,l}$ being constructed

Fig. 2. Recursive VPC construction model

The two-party VPC in Fig. 2(a) can be seen as a *building unit* for those VPCs on longer ULPs, where the two subchannels can be extended to any channel type as introduced in Section II. They can either be both VPCs, or one VPC and one LPC, or both LPCs. If now v_l wants to build a VPC with v_i, then v_l and v_i need to utilize $\gamma_{i,k}$ and $k \leftrightarrow l$ as the two subchannels as shown in Fig. 2(c). In order to reach the maximum capacity for the constructed VPC, we assign $a_{i,k}$ to the MLB of v_i on the new $e_{i,k}$: $\delta_{i,k} = a_{i,k}$. Symmetrically, we have $\delta_{k,i} = a_{k,i}$. After the $\gamma_{i,l}$ is constructed as shown in Fig. 2(d), the balance distribution pattern is similar to Fig. 2(b). Similarly, we have $a_{i,l} = \min(\delta_{i,k}, \delta_{k,l}) = \min(\min(\delta_{i,j}, \delta_{j,k}), \delta_{k,l})$ and $a_{l,i} = \min(\delta_{l,k}, \delta_{k,i}) = \min(\delta_{l,k}, \min(\delta_{k,j}, \delta_{j,i}))$. By this bottom-up construction from $\gamma_{i,j}$ to $\gamma_{i,l}$, the two-party VPC $\gamma_{i,l}$ with one more intermediary than $\gamma_{i,j}$ is being built. The *construction order* is from $\gamma_{i,j}$ to $\gamma_{i,l}$, and $\gamma_{i,l}$ is considered as a *upper-level* VPC of $\gamma_{i,k}$.

The construction is a recursive bottom-up process that starts from LPCs to the target VPC [20]. It is assumed that the subchannels of the VPCs will be prevented from being closed until these VPCs are closed. There is a special real-time value named "validity" [18] that the two parties of the VPC agree on when the VPC is opened to prevent the EUs from maliciously letting the intermediaries' money be locked forever in their VPCs. The VPC will be closed as the validity of time passes, which differs from the LPCs where the close phase is initiated by the parties of the channel.

In this paper, we focus on the following protocol design problem: given multiple node pairs in a PCN, we aim to design a protocol to construct a VPCN connecting these node pairs with high CNIRs.

IV. Design Rationale and Challenges

A. Design Rationale

When VPCs are created, all involved intermediaries need to lock a certain amount of balance as collateral. Their locked funds become unavailable until the upper-level VPCs are closed. Existing protocols [18], [20] related to the two-party VPCs involving more than one intermediary do not consider the collateral held by intermediaries during the VPC construction. Thus, it is necessary to design a protocol that can strategically select ULPs for the node pairs while considering the resulting capacity of the constructed VPCs and the number of involved intermediaries.

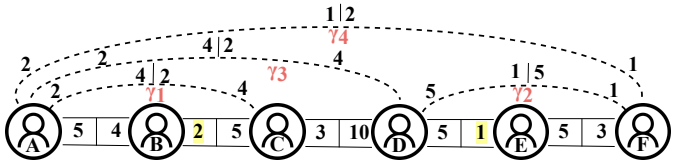
B. Design Challenges

Thor should achieve the VPC construction with maximum CNIR for the single-pair case and efficient construction of VPCs with high CNIR for the multi-pair case. The challenges are as follows:

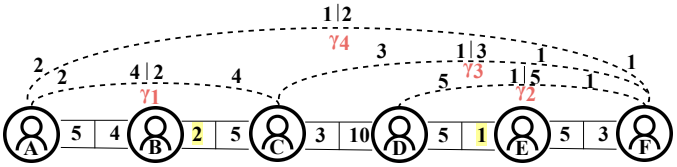
- Each node pair can try all of its paths and pick the one that gives it maximum CNIR. However, it may take exponential time as the number of paths of the node pair may not be polynomially bounded.
- Various subchannels and construction orders exist even when forming the target VPC on top of the same ULP. Before the node pair finds a ULP with to construct their VPC, we need to figure out whether the order they construct the subchannels affects the capacity of the VPC.
- Breadth-First Search (BFS) [24] can be used to search a BFS tree data structure. The resulting BFS tree can be utilized to discover paths and exclusively identify the shortest path connecting a pair of nodes. Even though Breadth-First Search ensures the minimum number of intermediaries, it does not ensure the highest CNIR.
- A solution to the problem of the single-pair case cannot be directly applied to the multi-pair case due to the mutual influence and competence between those ULPs sharing common LPCs. In addition, each path's allocated MLB can be computed after considering all path selections. Thus how to compute the available MLB on each edge before path selection has not been solved yet.

V. Design of Thor

Thor is based on three algorithms: Optimal ULP Finding for Single Pair (Algorithm 1), MLB computation for ULPs (Algorithm 2), and Efficient ULPs Finding for Multiple Pairs (Algorithm 3). Before introducing our algorithms, we first prove that the construction order does not affect the maximum capacity of the constructed VPC, then we address the rest challenges by introducing Algorithm 1 for finding the optimal ULP with maximum CNIR in the single-pair case. Finally, we use Algorithm 2 and Algorithm 3 to solve the problem of the multi-pair case based on Algorithm 1.



(a) First topology of the VPC between node A and F



(b) Second topology of the VPC between node A and F

Fig. 3. We present two examples to demonstrate that different subchannel topology of γ_4 does not impact the maximum capacity of γ_4.

A. Capacity is Independent of the Construction Order

Theorem 1. *The maximum capacity c of the virtual state channel γ for the node pair (s,t) on a ULP $P=\{P^{s\to t},P^{t\to s}\}$ is the sum of both $\delta(P^{s\to t})$ and $\delta(P^{t\to s})$ and the construction order of the subchannels does not affect the maximum capacity of γ:*

$$a_{s,t}=\delta(P^{s\to t}),a_{t,s}=\delta(P^{t\to s}) \tag{2}$$

$$c=a_{s,t}+a_{t,s}=\delta(P^{s\to t})+\delta(P^{t\to s}) \tag{3}$$

Proof. The capacity c consists of two portions $a_{s,t}$ and $a_{t,s}$ as shown in Eq. (1). We will show that both $a_{s,t}$ and $a_{t,s}$ come from multiple min-comparisons of the MLB values all the way from its P. Suppose there is a number of d intermediaries in $P=s\leftrightarrow I_1\leftrightarrow I_2\leftrightarrow\ldots\leftrightarrow I_d\leftrightarrow t$. Given the bottom-up construction approach, we consider the sequence in which the subchannels are constructed: starting from γ_{s,I_1}, then proceeding to γ_{s,I_2}, $\gamma_{s,I_3}\ldots$, until reaching the γ. Each two consecutive constructed VPCs differs by the length of one intermediary. Since P is bidirectional, the two maximum amounts $a_{s,t}$ and $a_{t,s}$ should follow:

$$a_{s,t}=\min(\ldots(\min(\delta_{s,I_1},\delta_{I_1,I_2}),\ldots),\delta_{I_d,t}) \tag{4}$$

$$a_{t,s}=\min(\ldots(\min(\delta_{I_1,s},\delta_{I_2,I_1}),\ldots),\delta_{s,I_1}) \tag{5}$$

Then we have $a_{s,t}=\delta(P^{s\to t})$ and $a_{t,s}=\delta(P^{t\to s})$ based on the definition of the MLB of a path. Thus, we prove Eq. (2) based on Eq. (4) and Eq. (5) for the above construction order. Following the definition of capacity, we have Eq. (3). Additionally, all MLBs in Eq. (4) and Eq. (5) originate from the bottom LPC channels. Instead of nesting multiple $\min$-operations, directly extracting the smallest MLB from the set of MLBs on P gives the same result (for example, $\min(\min(\min(a,b),c),d)=\min(a,b,c,d)$). In other words, the MLB order in the nested $\min$-operations in Eq. (4) and Eq. (5) is indeed the construction order of the subchannels. This order does not affect $\delta(P^{s\to t})$ and $\delta(P^{t\to s})$ since the minimum value can be directly extracted. In return, the construction order does not affect the maximum capacity of γ. □

Fig. 3 demonstrates this unique characteristic described in Theorem 1. The examples showcase two different subchannel

Algorithm 1: Optimal ULP Finding for Single Pair

Input: network $G = (\mathcal{V}, \mathcal{E})$, MLB set $\mathcal{M}$ and node pair (s, t)

Output: maximum CNIR r and ULP $\{P^{s\to t}, P^{t\to s}\}$

1 $r_{max} \leftarrow 0, P_{max} \leftarrow \{\}$;
2 **for** $t_1 \in \mathcal{M}$ and $t_2 \in \mathcal{M}$ **do**
3 Find a shortest bidirectional path $P = \{P^{s\to t}, P^{t\to s}\}$ from s to t, such that $\delta(P^{s\to t}) \geq t_1$ and $\delta(P^{t\to s}) \geq t_2$;
4 Compute r using Eq. (6);
5 **if** $r > r_{max}$ **then** $r_{max} \leftarrow r$, $P_{max} \leftarrow P$;
6 **end**
7 **return** r_{max} and P_{max}

topologies for the same target γ_4. Both ways of construction follow a bottom-up approach, meaning that the construction is as follows: $\gamma_1 \to \gamma_2 \to \gamma_3 \to \gamma_4$. For γ_1 in Fig. 3(a), the intermediary is B and the two EUs are A and C. The maximum amount that A can lock on γ_1 is $a_{A,C} = \min(5, 2) = 2$. Similarly, we have $a_{C,A} = \min(4, 5) = 4$. Then we have $a_{A,C} = a_{B,C} = 2$ and $a_{C,A} = a_{B,A} = 4$. Similar min-operations are then done recursively for γ_2, γ_3, and γ_4. Finally, we have $a_{A,F} = 2$ and $a_{F,A} = 1$. Another construction for γ_4 on the same ULP but with different subchannel constructions is shown in Fig. 3(b). However, the maximum capacity of γ_4 is $2 + 1 = 3$ in both Fig. 3(a) and Fig. 3(b). The capacity is from the two highlighted MLBs: 2 is the MLB of the path from A to F and 1 is the MLB of the path from F to A.

Based on Theorem 1, the problem of finding a ULP with maximum CNIR can be restated as to find such a ULP for the node pair (s, t), such that the sum of the two MLBs $\delta(P^{s\to t})$ and $\delta(P^{t\to s})$ over the number of intermediaries d on this ULP is the maximum.

$$r = \frac{\delta(P^{s\to t}) + \delta(P^{t\to s})}{d} \tag{6}$$

However, the original BFS does not support finding such an optimal ULP with the maximum CNIR in polynomial time. We need to modify BFS carefully.

B. ULP Finding for the Single-pair Case

We first designed the optimal algorithm for a single pair to find the ULP with maximum CNIR shown in Algorithm 1. As we aim to identify ULPs with the maximum CNIR of the target virtual channel capacity to the number of intermediate parties for a node pair, we first fix the two thresholds t_1 and t_2 for $s \to t$ and $t \to s$ directions, respectively (Line 2). The two threshold ranges are bounded by the set of MLB values $\mathcal{M}$. In total, there are $|\mathcal{M}|^2$ such iterations. Only when the two MLBs satisfy $\delta_{u,v} \geq t_1$ for the edge $e_{u,v}$ on $s \to t$ direction and $\delta_{v,u} \geq t_2$ for the edge $e_{v,u}$ on $t \to s$ direction, the neighbor v of u can be added to the ULP P. The r_{max} should have two corresponding ULPs $P^{s\to t}$ and $P^{t\to s}$. In the end, the maximum CNIR r and its P_{max} are returned. To summarize, the complexity of Algorithm 1 should be $O(m^2(n+m))$ due to the outer for-loop $O(m^2)$ and the inner BFS $O(n+m)$.

Lemma 1. *When the two thresholds are at $t_1 = t_1'$ and $t_2 = t_2'$, the ULP that Algorithm 1 returns under the iteration combination of t_1' and t_2' for node pair (s, t) satisfies that $\delta(P^{s\to t}) \geq t_1'$ and $\delta(P^{t\to s}) \geq t_2'$.*

Proof. In Algorithm 1, for a node pair (s, t), every bidirectional edge $u \leftrightarrow v$ added to the BFS tree under each iteration combination of t_1 and t_2 always guarantees that $\delta_{u,v} \geq t_1', \forall(u, v) \in P^{s\to t}$, and similar for $\delta_{v,u}$: $\delta_{v,u} \geq t_2', \forall(v, u) \in P^{t\to s}$. Since the MLB of every bidirectional edge added to the BFS tree has the above property, then the MLB should also have this property: $\delta(P^{s\to t}) \geq t_1'$, symmetrically we have $\delta(P^{t\to s}) \geq t_2'$. Thus we have proved the lemma. □

Theorem 2. *Algorithm 1 returns the optimal ULP with maximum CNIR for node pair (s, t) in a bidirectional connected network $G(\mathcal{V}, \mathcal{E})$.*

Proof. First, if the network is connected and the two nodes of the pair are not directly connected, there must exist at least one ULP between them, otherwise, the network becomes disconnected. We also need to prove that the maximum CNIR ULP must exist in at least one iteration of the combination of the two thresholds. We prove it by contradiction.

Suppose there exists a ULP with the maximum CNIR $r' = \frac{t_1' + t_2'}{d'}$ under two MLBs t_1' and t_2' with a number of d' intermediaries, but it is not returned in any combination of $t_1 \in \mathcal{M}$ and $t_2 \in \mathcal{M}$. Obviously, such ULP cannot be found when $t_1 > t_1'$ and $t_2 > t_2'$ (according to Lemma 1). But what if $t_1 \leq t_1'$ and $t_2 \leq t_2'$?

We first consider when $t_1 = t_1'$ and $t_2 = t_2'$. Follow our consumption for contradiction proof, Algorithm 1 should return a ULP with a smaller CNIR $r'' = \frac{t_1'' + t_2''}{d''}$ than r', given that $d'' \leq d'$ (shortest path is always a by-product of BFS [24]), $t_1'' \geq t_1'$ and $t_2'' \geq t_2'$ (Lemma 1). Then we have the CNIR for this ULP $r'' = \frac{t_1'' + t_2''}{d''} \geq \frac{t_1' + t_2'}{d'}$, which contradicts the hypothesis that $\frac{t_1' + t_2'}{d'}$ should be the highest ratio, thus it is already enough to prove Theorem 2.

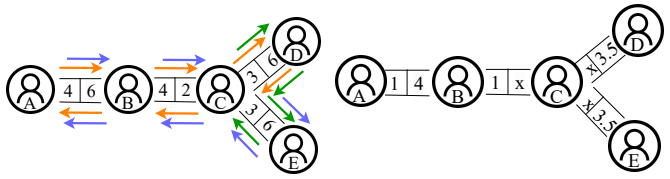
Regarding $t_1 < t_1'$ and $t_2 < t_2'$, we know that the shortest ULP is always returned, but it is not guaranteed that the maximum CNIR ULP is returned. Consequently, we have also demonstrated that if there is a minimum of one optimal ULP between node s and t with two MLBs t_1' and t_2', it ought to be discovered at least when $t_1 = t_1'$ and $t_2 = t_2'$. Hence, we have proved Theorem 2. □

C. ULPs Finding for the Multi-pair Case

In this section, we aim to design an algorithm to find ULPs with the maximum CNIR for the multi-pair case, which is the core of the Thor protocol. To address this problem, we design our algorithm that is inspired by the game theory-based algorithm in [25]. The network routing problem in [25] was formulated as a game, where each source-destination pair is a player and the objective of each player is to find a path with the maximum bandwidth under the max-min fair bandwidth allocation. Their algorithm converges to a set

of *best response* paths for all players, where no player can increase its bandwidth by unilaterally changing its path.

Since multiple ULPs may share a common edge, it is necessary to decide how to allocate MLBs among these ULPs. Algorithm 2 is designed to compute the MLB of each ULP given a set of ULPs under the max-min fair allocation principle proposed in [25]. Let $\Delta = (\delta(P_1^{s\to t}), \delta(P_1^{t\to s}), \ldots, \delta(P_K^{s\to t}), \delta(P_K^{t\to s}))$ denote the MLB vector and let $\mathbf{r} = (r_1, r_2, \ldots, r_K)$ denote the CNIR vector. The primary idea of Algorithm 2 is to calculate the *global MLB bottleneck* $b_g = argmin_{e_{i,j}\in\mathcal{E}} \frac{\delta_{i,j}}{w_{i,j}}$ iteratively, where b_g is defined as the edge possessing the least even share of the MLB and $w_{i,j}$ is the number of ULPs that utilizes $e_{i,j}$ (Line 5). Any ULP in the directional ULP set that uses b_g gets an equal share of the MLB on b_g and then is removed from the network. The MLBs are decreased by the MLB consumed from the removed ULPs. The above procedures are repeated until the directional ULP set becomes empty (Lines 5 to 14). Finally, we can use Δ to calculate the CNIR for every node pair (Line 16). We use Fig. 4(a) as an example to demonstrate Algorithm 2. In the first iteration, $e_{C,B}$ is selected as b_g as $\frac{\delta_{C,B}}{w_{C,B}} = 1$, then we get $\delta(P_2) = \delta(P_4) = 1$. In the second iteration, $\delta(P_3) = \delta(P_5) = 1.5$ and $b_g = e_{C,E}$. In the final iteration, $\delta(P_1) = \delta(P_6) = 1.5$ and $b_g = e_{C,D}$, the resulting network is shown in Fig. 4(b). The CNIR for a pair can only be calculated after both MLBs of $P^{s\to t}$ and $P^{t\to s}$ are returned. Finally, we can get the three ratios for the node pairs: $r_1 = \frac{\delta(P_1)+\delta(P_2)}{2} = 1.25$, $r_2 = \frac{\delta(P_3)+\delta(P_4)}{2} = 1.25$ and $r_3 = \frac{\delta(P_5)+\delta(P_6)}{1} = 3$. Algorithm 2 has a complexity of $O(Km)$.



(a) Before running Algorithm 2 (b) After running Algorithm 2

Fig. 4. An example with six ULPs for three node pairs (A, D), (A, E) and (D, E): $P_1 = A \to B \to C \to D$, $P_2 = D \to C \to B \to A$, $P_3 = A \to B \to C \to E$, $P_4 = E \to C \to B \to A$, $P_5 = D \to C \to E$, $P_6 = E \to C \to D$. Removed edges are denoted with "x".

Let us introduce more notations. When the ULP of the node pair (s_k, t_k) P_k is not in the network, let $\mathcal{P}_{-k} = \{P_1, \cdots, P_{k-1}, P_{k+1}, \cdots, P_K\}$ denote the set of ULPs. Let $\delta_{-k}(P_x)$ denote the MLB of the ULP P_x. Let Δ_{-k} denote the MLB vector. Let $W_{-k}^{i,j}$ denote the ULPs of the node pairs that utilize edge $e_{i,j}$. Let $w_{-k}^{i,j} = |W_{-k}^{i,j}|$ denote the number pairs sharing edge $e_{i,j}$. After the node pair (s_k, t_k) changes its ULP to P_k', let $\mathcal{P}' = \{P_1', P_2', \ldots, P_K'\}$ denote the ULP set for all node pairs, where $P_x' = P_x$ if $x \neq k$.

Definition 1. *[Best Response ULP] The best response ULP for the node pair (s_k, t_k) is a ULP P_k given $\mathcal{P}_{-k}$, such that the CNIR r_k is maximized.*

Next, we introduce the concept of **observed available MLB** followed by the *observed available bandwidth* in [25], which

Algorithm 2: MLB computation for ULPs

Input: network G, MLB set $\mathcal{M}$ and ULP set $\mathcal{P}$
Output: MLB vector Δ and CNIR vector $\mathbf{r}$

1 $\mathcal{E}^c \leftarrow \mathcal{E}, \mathcal{P}^c \leftarrow \emptyset, \delta_{i,j}^c \leftarrow \delta_{i,j}, \forall e_{i,j} \in \mathcal{E}$;
2 **foreach** $k \in [1, K]$ **do** $\mathcal{P}^c \leftarrow \mathcal{P}^c \cup \{P_k^{s\to t}\} \cup \{P_k^{t\to s}\}$;
3 $\delta(P) \leftarrow 0, \forall P \in \mathcal{P}^c$;
4 **while** $\mathcal{P}^c \neq \emptyset$ **do**
5 $\quad b_g \leftarrow argmin_{e_{i,j}\in\mathcal{E}} \frac{\delta_{i,j}^c}{w_{i,j}}$;
6 $\quad \delta_{temp} \leftarrow \frac{\delta_{b_g}^c}{w_{b_g}}$;
7 $\quad$ **for** $P \in \mathcal{P}^c$ and $b_g \in P$ **do**
8 $\quad\quad \delta(P) \leftarrow \delta_{temp}$;
9 $\quad\quad$ **forall** $e_{i,j} \in P$ **do**
10 $\quad\quad\quad \delta_{i,j}^c \leftarrow \delta_{i,j}^c - \delta(P); w_{i,j} \leftarrow w_{i,j} - 1$;
11 $\quad\quad\quad$ **if** $\delta_{i,j}^c = 0$ **then** $\mathcal{E}^c \leftarrow \mathcal{E}^c \setminus e_{i,j}$;
12 $\quad\quad$ **end**
13 $\quad\quad \mathcal{P}^c \leftarrow \mathcal{P}^c \setminus \{P\}$;
14 $\quad$ **end**
15 **end**
16 **for** $k \in [1, K]$ **do** $r_k \leftarrow \frac{\delta(P_k^{s\to t})+\delta(P_k^{t\to s})}{len(P_k^{s\to t})-1}$;
17 **return** Δ and $\boldsymbol{r}$

will be helpful in explaining Algorithm 3. We leave the proof of the correctness of observed available MLB in [25]. The observed available MLB $\delta_{i,j}^\circ$ of edge $e_{i,j} \in \mathcal{E}$ are calculated based on the following equations:

$$\hat{\mathcal{W}}_{-k}^{i,j}(x) = \{z \mid z \in \mathcal{W}_{-k}^{i,j} \wedge \delta_{-k}(P_z) < \delta_{-k}(P_x)\} \tag{7}$$

$$\overline{\mathcal{W}}_{-k}^{i,j} = \{x \mid x \in \mathcal{W}_{-k}^{i,j} \wedge \delta_{-k}(P_x) \geq \frac{\delta_{i,j} - \sum_{z\in\hat{\mathcal{W}}_{-k}^{i,j}(x)} \delta_{-k}(P_z)}{w_{-k}^{i,j} - |\hat{\mathcal{W}}_{-k}^{i,j}(x)| + 1}\} \tag{8}$$

$$\delta_{i,j}^\circ = \frac{\delta_{i,j} - \sum_{x\in\mathcal{W}_{-k}^{i,j}\setminus\overline{\mathcal{W}}_{-k}^{i,j}} \delta_{-k}(P_x)}{\left|\overline{\mathcal{W}}_{-k}^{i,j}\right| + 1} \tag{9}$$

$\hat{\mathcal{W}}_{-k}^{i,j}(x)$ is the set of pairs whose current assigned MLB is less than the pair x on edge $e_{i,j}$. $\overline{\mathcal{W}}_{-i}^{i,j}$ is the set of pairs such that the newly assigned MLB is at least as large as the MLB of any other pairs. The observed available MLB on all edges can be computed in $O(K \log K + mK)$ time for K pairs [25].

Now we introduce Algorithm 3 that is designed to find a set of ULPs for the K node pairs, such that no pair can increase its CNIR by unilaterally changing its ULP. We abbreviate the Algorithm 1 as $Ag1$ and Algorithm 2 as $Ag2$ in Algorithm 3 to save space. Algorithm 3 proceeds in a round-robin fashion with only one node pair changing its ULP at each state. Algorithm 3 starts with calculating the initial ULP set (Lines 2 to 5). Then when a node pair plans to change its ULP, it runs the following steps until no ULP is changed in the ULP set:

1) Calculate the MLB vector Δ and the CNIR vector $\mathbf{r}$ for all ULPs using Algorithm 1 (Lines 8).
2) Calculate the MLB vector Δ_{-k} for the set of ULPs without P_k using Algorithm 1(Line 9).

Algorithm 3: Efficient ULPs Finding for Multiple Pairs

Input: network $G = (\mathcal{V}, \mathcal{E})$ and MLB set $\mathcal{M}$
Output: ULP set $\mathcal{P}$

1 $\mathcal{P} \leftarrow \emptyset$;
2 **foreach** $k \in [1, K]$ **do**
3 Run `Ag1` $(G, \mathcal{M}, (s_k, t_k))$ to obtain P_k;
4 $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_k\}$;
5 **end**
6 **repeat**
7 **foreach** $k \in [1, K]$ **do**
8 Run `Ag2` $(G, \mathcal{M}, \mathcal{P})$ to obtain Δ and $\mathbf{r}$;
9 Run `Ag2` $(G, \mathcal{M}, \mathcal{P} \setminus \{P_k\})$ to obtain Δ_{-k};
10 Compute $\mathcal{M}^o$ using Eq. (9);
11 Run `Ag1` $(G, \mathcal{M}^o, (s_k, t_k))$ to obtain r'_k and P'_k;
12 **if** $r'_k > r_k$ **then** $P_k \leftarrow P'_k$;
13 $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_k\}$;
14 **end**
15 **until** *no ULP is changed* ;
16 **return** $\mathcal{P}$.

3) Calculate the observed available MLBs $\mathcal{M}^o$ for all edges using Algorithm 2 (Line 10).
4) Re-select a ULP for pair (s_k, t_k) based on the resulting network using Algorithm 1 (Line 11), if the new CNIR is higher than the old one, then the old ULP is replaced, otherwise the old ULP is kept (Line 12).
5) Add the updated ULP to the ULP set (Line 13).

Now we prove the convergence of Algorithm 3.

Lemma 2. *Assume that the node pair (s_k, t_k) changes its ULP from P_k to P'_k. Define $\mathcal{W} = \{1, 2, \ldots, K\}$, $\mathcal{W}^{=} = \{x \in \mathcal{W} \mid r_x = r'_x\}$, $\mathcal{W}^{\uparrow} = \{x \in \mathcal{W} \mid r_x < r'_x\}$, and $\mathcal{W}^{\downarrow} = \{x \in \mathcal{W} \mid r_x > r'_x\}$, then we have $\min_{x \in \mathcal{W}_{\downarrow} \cup \mathcal{W}_{\uparrow}} r'_x > \min_{x \in \mathcal{W}_{\downarrow} \cup \mathcal{W}_{\uparrow}} r_x$.*

Proof. The Lemma 4.1 in [25] has proved that the minimum bandwidth among the paths whose bandwidths (in our case, MLB of the path) change increases strictly after a path selfishly changes its path. First, it is clear that $k \in \mathcal{W}_{\uparrow}$, otherwise, the node pair (s_k, t_k) has no incentive to change its ULP with a less CNIR. In addition, the ULP lengths for pairs in $\mathcal{W} \setminus \{k\}$ are fixed before and after k changes its ULP. Thus the CNIRs of these ULPs will strictly change with respect to the changing pattern of the MLB of the paths (Eq. (6)). This indicates that the minimum CNIR of $\mathcal{W}_{\downarrow} \cup \mathcal{W}_{\uparrow}$ increases as well. We have proved that the minimum CNIR among the ULPs whose ratios change also increases strictly. □

Theorem 3. *Algorithm 3 converges in $O((Knm)^K(m^2(n+m)))$ time, where m is the number of edges, n is the number of nodes and K is the number of node pairs.*

Proof. Theorem 4.1 in [25] proves that the vector of the bandwidth increases lexicographically (in our case, MLB of the path). Similarly, we conclude that whenever a node pair changes its ULP, the CNIR vector $\mathbf{r}$ also increases lexicographically (Lemma 2). We know that there is a finite number of ULPs for each pair. Thus the number of ULP configurations is finite as well. Now we prove an upper bound on the number of times the ordering of CNIR can increase. Lemma 6.1 in [25] proves that the MLB value on a global bottleneck must be equally shared by all ULPs using it. Given that the CNIR of a ULP is decided by both the MLB and the number of intermediaries, thus the number of possible values of the minimum CNIR is bounded by $O(Knm)$. There are at most K node pairs whose ULPs are at this value. Repeating the same analysis for all the pairs, we conclude that the upper bound is $O((Knm)^K)$ for K pairs. Therefore, the time complexity of Algorithm 3 is $O((Knm)^K(m^2(n+m)))$. □

D. VPC Construction Protocol

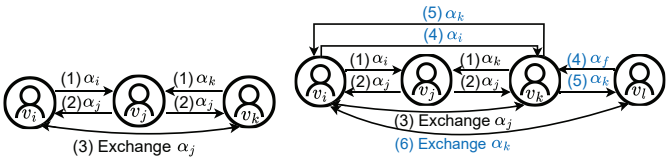
In Thor, time is divided into discrete time slots. At the beginning of each time slot, each pair that desire to construct a VPC between them will broadcast its intent. After receiving the broadcast intents from other pairs, each pair first obtains the network $G = (\mathcal{V}, \mathcal{E})$ and $\mathcal{M}$, which are periodically announced by users who are interested in joining as intermediaries. Each node pair then follow Algorithm 3 to select its ULP and calculate its MLBs. We now prove that all pairs will follow the Thor protocol in Theorem 4 assuming there is no collusion.

Theorem 4. *Each pair has no incentive to not follow the protocol unilaterally if all other pairs follow.*

Proof. Assume pair (s_k, t_k) does not follow the protocol, while others follow. Let $\mathcal{P}_{-k}$ denote the set of ULPs of other pairs. Let P'_k denote the ULP selected by the pair (s_k, t_k). Let P_k denote the ULP of pair (s_k, t_k) if it followed the protocol. Let r' and r denote the CNIRs of pair (s_k, t_k) with these two ULPs, respectively. By the definition of the best response ULP, $r' \leq r$. Therefore, pair (s_k, t_k) cannot increase its CNIR by not following the Thor protocol. □

Each node pair then starts building its own VPC on the selected ULP concurrently. The computed MLBs are the actual balances that will be locked on the VPC. Now we introduce how the recursive process is done on a protocol level using the computed MLBs from our Algorithm 3. We follow the same balanced construction as in [20]. Balanced construction means that s and t construct the subchannels concurrently with their next closest intermediary from the two ends of the ULP toward the center of the ULP. Furthermore, the two VPCs used to build an upper-level VPC have approximately the same length. Let β_s denote the number of the constructed subchannels that have s as one of their EUs. Let β_t denote the number of the constructed subchannels that have t as one of their EUs. We have the following relationship between β_s and β_t when the number of intermediaries is d:

$$\begin{cases} \beta_s = 1, \beta_t = 0 & \text{if } d = 1 \\ \beta_s = \beta_t = \frac{d-1}{2} & \text{if } d \text{ is odd and } d > 1 \\ \beta_s = \frac{d}{2}, \beta_t = \frac{d}{2} - 1 & \text{if } d \text{ is even and } d > 1 \end{cases}$$



(a) Basis step (b) Recursion step

Fig. 5. A bottom-up recursive VPC construction protocol

We use Fig. 5(a) to illustrate the basis case when the two subchannels are both LPCs. In this figure, v_i and v_k are the two EUs, v_j is the intermediary, and the two LPC channels are $i \leftrightarrow j$ and $j \leftrightarrow k$. $\gamma_{i,k}$ is the basis VPC being constructed. α_i, α_j and α_k are the construction approvals. A construction approval is a tuple with four attributes. For EUs, the first is the VPC that needs to be constructed, the second is the amount of balance that the approval sender needs to lock, the third is the amount of balance that the other EU needs to lock and the fourth is the signature of the sender. Therefore, $\alpha_i := (\gamma_{i,k}, \delta(P^{s\to t}), \delta(P^{t\to s}), \sigma_i)$ and $\alpha_k := (\gamma_{i,k}, \delta(P^{t\to s}), \delta(P^{s\to t}), \sigma_k)$, respectively. For the intermediary, the difference is that the second and the third attribute are both the amount of balance that the intermediary needs to lock on behalf of the two EUs, respectively. Therefore, $\alpha_j := (\gamma_{i,k}, \delta(P^{s\to t}), \delta(P^{t\to s}), \sigma_j)$.

The construction of a basis VPC involves three steps [18]:

1) v_i and v_k send their approvals α_i and α_k to v_j.
2) If v_j agrees on both α_i and α_k, it sends its own approval α_j to v_i and v_k.
3) v_i and v_k exchange the α_j they received.

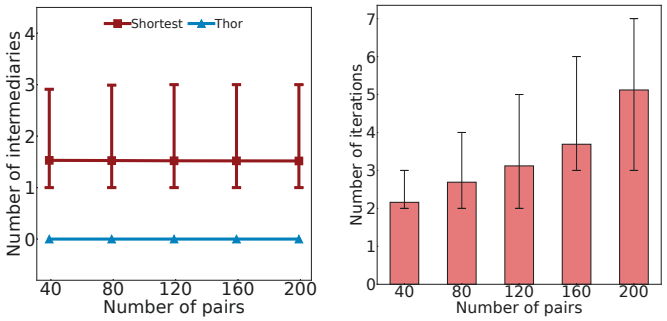
Once the above steps are performed, v_i locks $\delta(P^{s\to t})$ and v_k locks $\delta(P^{t\to s})$ from their respective LPCs to construct $\gamma_{i,k}$. The same processes can be performed recursively for any VPC that utilizes a ULP with more than one intermediary. The recursive step is shown in Fig. 5(b) for any subchannels that are not both LPCs. v_i and v_l construct the $\gamma_{i,l}$ on the LPC $k \leftrightarrow l$ and the VPC $\gamma_{i,k}$. The above three steps of exchanging messages will be repeated among v_i, v_k, and v_l. To apply the recursive process in a more general case, now consider such a ULP $P = v_i \leftrightarrow I_1 \leftrightarrow I_2 \leftrightarrow \ldots \leftrightarrow I_d \leftrightarrow v_q$ with d intermediaries $\{I_1, I_2, \ldots, I_d\}$. If d is odd, then the constructed subchannels will be perfectly balanced. v_i constructs the first VPC with I_2, then constructs the second VPC with I_3, until it constructs with $I_{\frac{d-1}{2}}$. Symmetrically, v_q constructs the first VPC with I_{d-1}, then constructs the second VPC with I_{d-2}, until it constructs with $I_{\frac{d-1}{2}}$. Finally, v_i and v_q construct the $\gamma_{i,q}$ with $I_{\frac{d-1}{2}}$ served as the intermediary. If d is even, then v_i needs to construct one more VPC than v_q. Both v_i and v_q start the same subchannel building process concurrently until both v_i and v_q construct the VPC with $I_{\frac{d}{2}+1}$. Finally, v_i and v_q construct $\gamma_{i,q}$ with $I_{\frac{d}{2}+1}$ served as the intermediary.

VI. Performance Evaluation

A. Evaluation Setup

We used a core network with 298 nodes and 2093 edges from the Bitcoin LN on Jan. 25, 2022 [26]. A pre-processing of the whole network was operated to obtain a connected subgraph from the original topology. Assume the top 4% with the highest capacity are participating in constructing the VPC. In addition, all isolated nodes and nodes with degrees that are less or equal to 3 were removed to get the largest connected component. After prepossessing, the minimum, maximum, and mean capacities were \$4212, \$366240, and \$13803, respectively. The number of node pairs was set in the range of 40, 80, 120, 160, and 200 for the VPC constructions. We ran each setting for 50 times with different seeds to average out random noise. We evaluated the following evaluation metrics: the number of hops needed in the transaction phase, CNIR, and the capacity of the VPCs. The current payment channel construction is a nascent development without recent algorithms for direct comparison. Therefore, we compare Thor with the following baseline algorithms:

1) K-path union (KPU): We first use Algorithm 1 to compute the ULP for each pair, then use Algorithm 2 to compute the capacity and the CNIR of each ULP without any further iterations in the multi-pair case.
2) Shortest path (Shortest): Dijkstra algorithm [27] is deployed for finding the shortest ULP. For the single-pair case, we compare Shortest with Algorithm 1. For the multi-pair case, we first use Shortest to find ULPs, then calculate the CNIR and capacity of each ULP using Algorithm 2. Note that the current PCN protocol also uses the shortest path algorithm in routing (PCN-shortest), thus we also compare Thor with Shortest for measuring the number of intermediaries.
3) Widest path (Widest): The widest path algorithm [28] is deployed for finding ULPs that maximize the capacities of the ULPs. For the single-pair case, we compare Widest with Algorithm 1. For the multi-pair case, we first use Widest to find ULPs, then calculate the CNIR and capacity of each ULP using Algorithm 2.

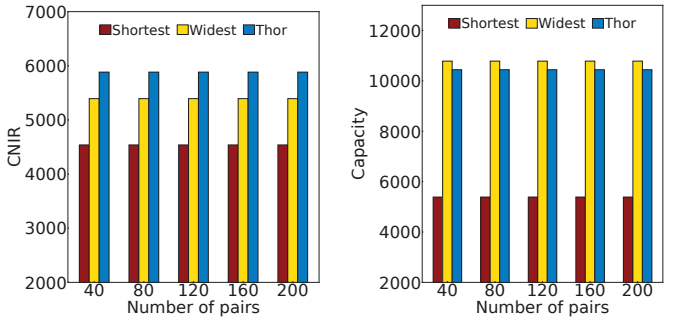


(a) Number of pairs vs. number of intermediaries (b) Number of pairs vs. number of iterations

Fig. 6. Number of intermediary results and convergence results

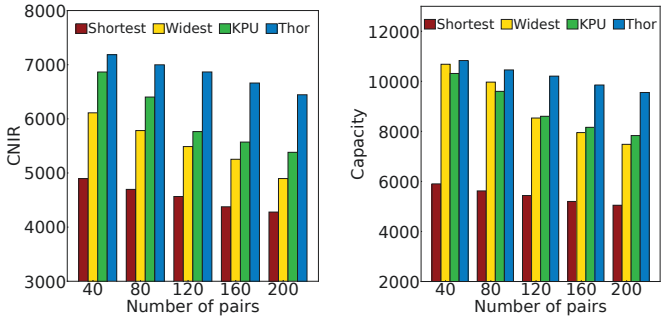
B. Results

Fig. 6(a) shows the number of intermediaries involved in Shortest and Thor. Shortest shows higher maximum, minimum, and average of the intermediaries compared to Thor. This is due to the HTLC routing protocol used in a PCN, where routers are needed during the transaction phase. However, the



(a) Number of pairs vs. CNIR (b) Number of pairs vs. capacity

Fig. 7. Comparison results for the single-pair case



(a) Number of pairs vs. CNIR (b) Number of pairs vs. capacity

Fig. 8. Comparison results for the multi-pair case

pair of users make transactions through the VPC between them in a VPCN, thus only one hop is needed. This is beneficial for those pairs who desire to make transactions without going through routers. The convergence results are shown in Fig. 6(b), where one iteration is from Line 6 to Line 15 in Algorithm 3. The average number of iterations increases with the number of node pairs. We also observe that the average number of iterations is less than 10 in all cases studied, which is far less than the theoretic bound in Theorem 3.

Fig. 7 shows the performance of the three algorithms in CNIR and capacity with respect to the number of pairs in the single-pair case. Each pair runs Algorithm 1 individually and obtains their CNIR and capacity. Fig. 7(a) and Fig. 7(b) show the average CNIR and average capacity for the three algorithms Shortest, Widest and Thor. In Fig. 7(a), Thor outperforms the Shortest and Widest algorithms in terms of CNIR. This is because Thor is optimal in finding the ULP with the maximum CNIR for the single-pair case (Theorem 2). In Fig. 7(b), the Shortest, which considers only the shortest ULP, inherently can only achieve lower capacity than Thor and Widest, respectively. Widest always selects a ULP with the highest capacity, thus it has the best performance in capacity compared to the other two algorithms. Thor considers CNIR and can reach a relatively bigger capacity than Shortest, but it does not guarantee the maximum capacity.

Fig. 8 shows the performance of the four algorithms with the increase of the number of pairs in terms of CNIR and capacity for the multi-pair case. Fig. 8(a) shows the CNIR achieved by Thor, Shortest, Widest, and KPU algorithms in the multi-pair case. It is observed that Thor always outperforms the other three algorithms in CNIR at any number of pairs. A factor contributing to the higher CNIR is that Thor lets the node pair select the best response ULP. Widest shows better CNIR, this is because Widest still considers the capacity while Shortest does not. For KPU, it uses Algorithm 1 to find the ULPs and then calculate the CNIR for all pairs using Algorithm 2. Even though KPU selects the ULPs based on the maximum CNIR in the single-pair case, the returned ULP set is not the best response path set in the multi-pair case. Thus, the actual CNIR achieved by KPU is less than Thor in the multi-pair case.

Fig. 8(b) shows the capacity achieved by the four algorithms when the number of pairs ranges from 40 to 200. Shortest selects ULPs based on the least number of intermediaries without considering the capacity of the ULP, hence it is expected to have the least average capacity. Widest chooses ULPs based on the maximum capacity of the ULP for each pair in the multi-pair case. However, the actual allocation of the MLB for those ULPs is computed mutually after all ULPs are selected by Widest in Algorithm 2. Thus, even though Widest initially finds the ULPs with the maximum capacity, the actual capacities of these selected ULPs should be determined by the allocated MLBs, which are not guaranteed to be the maximum capacities for all pairs. In addition, since there is no iteration performed and no node pair can select ULP with better CNIR in KPU, KPU shows less capacity compared to Thor.

Both CNIR and capacity achieved by all algorithms decrease as the number of node pairs increases in Fig. 8. This is expected due to the competence of MLB on the shared common edges. A decrease in MLB negatively impacts the CNIR and capacity for the multi-pair case.

To summarize, our evaluation results show the importance of constructing VPC in help reduce the number of intermediaries for transactions. In addition, for the single-pair case, the results show that Algorithm 1 can find ULPs with the maximum CNIR. For the multi-pair case, the results show that Algorithm 3 converges to the best response ULPs rapidly and achieves very good CNIR as well as capacity.

VII. Conclusion

In this paper, we introduced the new metric CNIR and investigated the virtual payment channel construction protocol Thor. We mainly studied the problem of how to construct VPCs for the single-pair case and the multi-pair case. We stated a set of design rationales and challenges for the Thor design. We designed our algorithm to find the ULP with maximum CNIR for the single-pair and to efficiently find the ULPs for multiple pairs based on the single-pair case. We also designed the VPC construction protocol for multiple users to construct a VPCN concurrently. We proved that Thor can construct VPC with maximum CNIR for the single-pair case and construct a VPCN for multiple pairs. The evaluation results demonstrate that Thor can find the ULP with maximum CNIR for the single-pair case, and reach high CNIR for the multi-pair case.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21260, 2008.

[2] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "An overview of smart contract: architecture, applications, and future trends," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 108–113.

[3] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2018, pp. 1204–1207.

[4] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE access*, vol. 8, pp. 125 244–125 262, 2020.

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

[6] R. Network-Fast, "cheap, scalable token transfers for ethereum," *Accessed: Jul*, vol. 7, p. 2020, 2018.

[7] Q. Bai, Y. Xu, and X. Wang, "Understanding the benefit of being patient in payment channel networks," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1895–1908, 2022.

[8] 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.

[9] Y. Zhang and D. Yang, "Robustpay: Robust payment routing protocol in blockchain-based payment channel networks," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–4.

[10] 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 *2018 27th international conference on computer communication and networks (ICCCN)*. IEEE, 2018, pp. 1–9.

[11] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and state channels: Payment networks that go faster than lightning," in *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers*. Springer, 2019, pp. 508–526.

[12] T. Close and A. Stewart, "Forcemove: an n-party state channel protocol," *Magmo, White Paper*, 2018.

[13] F. Engelmann, H. Kopp, F. Kargl, F. Glaser, and C. Weinhardt, "Towards an economic analysis of routing in payment channel networks," in *Proceedings of the 1st workshop on scalable and resilient infrastructures for distributed ledgers*, 2017, pp. 1–6.

[14] 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: ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*. Springer, 2017, pp. 411–419.

[15] S. Mazumdar, S. Ruj, R. G. Singh, and A. Pal, "Hushrelay: A privacy-preserving, efficient, and scalable routing algorithm for off-chain payments," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2020, pp. 1–5.

[16] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre, "Routing payments on the lightning network," in *2018 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, 2018, pp. 1161–1170.

[17] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards bitcoin payment networks," in *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I 21*. Springer, 2016, pp. 57–76.

[18] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 106–123.

[19] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.

[20] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 949–966.

[21] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, and K. Hostáková, "Multi-party virtual state channels," in *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*. Springer, 2019, pp. 625–656.

[22] C. D. Clack and C. McGonagle, "Smart derivatives contracts: the isda master agreement and the automation of payments and deliveries," *arXiv preprint arXiv:1904.01461*, 2019.

[23] N. Papadis and L. Tassiulas, "Blockchain-based payment channel networks: Challenges and recent advances," *IEEE Access*, vol. 8, pp. 227 596–227 609, 2020.

[24] A. Bundy and L. Wallen, "Breadth-first search," *Catalogue of artificial intelligence tools*, pp. 13–13, 1984.

[25] D. Yang, G. Xue, X. Fang, S. Misra, and J. Zhang, "Routing in max-min fair networks: A game theoretic approach," in *The 18th IEEE International Conference on Network Protocols*. IEEE, 2010, pp. 1–10.

[26] "The Lightning Network." [Online]. Available: https://lightning.network/

[27] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[28] A. P. Punnen, "A linear time algorithm for the maximum capacity path problem," *European Journal of Operational Research*, vol. 53, no. 3, pp. 402–404, 1991.