Fence: Fee-based Online Balance-aware Routing in Payment Channel Networks

Xiaojian Wang, *Student Member, IEEE,* Ruozhou Yu, *Senior Member, IEEE,* Dejun Yang, *Senior Member, IEEE,* Guoliang Xue, *Fellow, IEEE,* Huayue Gu, *Student Member, IEEE,* Zhouyu Li, *Student Member, IEEE,* and Fangtong Zhou, *Student Member, IEEE*

***Abstract*—Scalability is a critical challenge for blockchain-based cryptocurrencies. Payment channel networks (PCNs) have emerged as a promising solution for this challenge. However, channel balance depletion can significantly limit the capacity and usability of a PCN. Specifically, frequent transactions that result in unbalanced payment flows from two ends of a channel can quickly deplete the balance on one end, thus blocking future payments from that direction. In this paper, we propose Fence, an online balance-aware fee setting algorithm to prevent channel depletion and improve PCN sustainability and long-term throughput. In our algorithm, PCN routers set transaction fees based on the current balance and level of congestion on each channel, in order to incentivize payment senders to utilize paths with more balance and less congestion. Our algorithm is guided by online competitive algorithm design, and achieves an asymptotically tight competitive ratio with constant violation in a unidirectional PCN. We further prove that no online algorithm can achieve a finite competitive ratio in a general PCN. Extensive simulations under a real-world PCN topology show that Fence achieves high throughput and keeps network channels balanced, compared to state-of-the-art PCN routing algorithms.**

***Index Terms*—Blockchain, payment channel network, routing, online algorithm, competitive analysis**

I. Introduction

Thanks to decentralization of the blockchain, cryptocurrencies such as Bitcoin [40] can execute transactions trustlessly. However, compared to payment systems like Visa which can handle $24,000$ transactions per second (tx/s) [7], Bitcoin can only process around 7 tx/s. This scalability issue of the blockchain is due to the global consensus for every transaction [13].

Off-chain payment channels were proposed as a promising solution for this challenge. With only two on-chain transactions to open and close a channel respectively, two nodes can execute many transactions without committing all transactions to the blockchain. A Payment Channel Network (PCN) is a network of payment channels, used to execute off-chain payments between users without a direct channel. A real-world example is the Bitcoin's Lightning Network (LN), which has a capacity of ₿$5,226$ (or \$$109,122,968$) as of January 2023 [6].

Payment routing is a key challenge in a PCN. A payment's success requires all channels on its payment path to have enough balance for forwarding. Yet, the balance on one direction of a channel may deplete due to mismatched transaction flows on both directions [52]. When depletion happens, the depleted end can no longer forward payments until its balance is replenished via opposite-direction payments. Further, senders do not know if channels along the selected payment paths have sufficient balance due to the balance privacy requirement in a PCN [5]. Hence current PCNs only route payments in a trial-and-error manner, leading to a low payment success ratio.

Existing work mostly tries to improve the success ratio in three different ways. Balance probing [59] violates the balance privacy of a PCN [17]; payment slicing and queueing [49] raises payment latency, transaction fee and network overhead; active rebalancing [32] incurs on-chain or forwarding costs.

In this paper, we explore the missed opportunity of using transaction fees to help the network remain balanced. The intuition is to indirectly influence the path selection strategy of senders by dynamically adjusting the transaction fee setting of routers in the network, thus achieving overall network balance and increasing long-term throughput. Specifically, routers can set transaction fees based on their channels' congestion or imbalance levels. Senders are incentivized to pick paths that have lower fees and thus are less congested and more balanced. Inspired by competitive online algorithm design, we specifically propose an *exponential* fee function, such that the transaction fee increases exponentially with the level of congestion or imbalance on a channel. Our fee-based approach allows PCN senders to still pick the minimum-fee path without guessing the balance of each channel. Furthermore, our algorithm can be implemented as a decentralized protocol without landmarks or trusted central servers, and can be easily adjusted to protect the balance privacy of payment channels. Existing PCN routing algorithms mostly lack theoretical analysis and solely rely on empirical performance evaluation, which we propose to address through the competitive analysis framework [14]. Through theoretical analysis, we show that no online algorithm can achieve finite competitiveness on a general bidirectional PCN, and our algorithm can achieve an *asymptotically tight* competitive ratio on a unidirectional PCN. We perform extensive and comprehensive simulation experiments based on real-world PCN topology and transaction data. Results show that our algorithm significantly outperforms state-of-the-art solutions in terms of payment throughput and success ratio, while keeping the network balanced over the long run.

Our main contributions are summarized as follows:

- We propose a novel algorithm for balance-aware, high-

Wang, Yu ({xwang244, ryu5}@ncsu.edu) are with North Carolina State University, Raleigh, NC 27606, USA. Yang (djyang@mines.edu) is with Colorado School of Mines, Golden, CO 80401, USA. Xue (xue@asu.edu) is with Arizona State University, Tempe, AZ 85287, USA. Gu, Li, Zhou ({hgu5, zli85, fzhou}@ncsu.edu) are with North Carolina State University, Raleigh, NC 27606, USA. This research was supported in part by NSF grants 2007083, 2008935 and 2045539.

throughput online payment routing in a PCN, which reacts to channel congestion and imbalance with adaptive fee setting.

- We prove that our algorithm achieves an *asymptotically tight* competitive ratio in a unidirectional PCN, and show that no online algorithm can achieve a finite competitive ratio in a general bidirectional PCN.
- Extensive simulations show that our algorithm can improve the throughput while keeping the network balanced, compared to existing routing and fee setting approaches.

Organization. §II introduces background and related work. §III presents our system model. §IV proposes our online balance-aware fee setting algorithm. §V presents competitive analysis results. §VI explains protocol design details. §VII shows evaluation results. §VIII concludes the paper.

II. Background and Related Work

A. Payment Channel Network

Cryptocurrencies like Bitcoin [40] and Ethereum [56] have put forward an innovative permissionless paradigm based on the blockchain technology. A global consensus protocol is used for global participants to agree on the state of an append-only distributed ledger, which is maintained by all the users of the system. However, global consensus requires high time cost and computational and storage investment because each individual transaction needs to be confirmed by a majority of maintainers of the network. For example, the Bitcoin blockchain can only process about 7 transactions per second, which is several orders of magnitude worse than the mature Visa network [7]. As a result, the blockchain-based cryptocurrencies are not widely used for daily transactions on a large scale.

Some approaches have been put forward to address the scalability issue. These approaches can be divided into two tracks: on-chain scaling (Layer 1) and off-chain scaling (Layer 2). On-chain scaling like sharding [18] improves scalability by dividing the network and the blockchain states into smaller shards, where consensus is reached within each shard. But sharding suffers from reduced security and additional overhead for inter-shard communication, and existing solutions have not sufficiently addressed these issues [28].

Another track of approach is off-chain scaling which improves the transaction processing rate by moving the consensus off-chain and requires only a small number of transactions to be on-chain. One most promising example is the payment channel network (PCN). A payment channel addresses the blockchain scalability issue by consolidating many off-chain transactions into two on-chain transactions [31]. To open a channel, two users as channel owners publish an on-chain transaction to deposit into a multi-signature address controlled by them. Their deposits are then regarded as their initial balances on the channel. To carry out a payment, both owners of the channel agree to update the channel balance distribution based on the payment direction and amount. An infinite number of transactions (subject to bidirectional channel balances) can be conducted before the channel is closed and the final balances committed to the blockchain. For example, suppose S and D open a channel by each depositing ₿10 as shown in

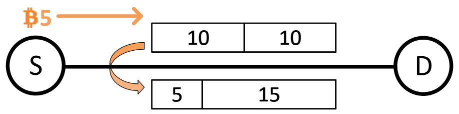


Fig. 1. Off-chain payment through channel balance redistribution.

Fig. 1. If S transfers ₿5 to D, the balance decreases by ₿5 on S's side and increases by ₿5 on D's side, as approved by both parties.

A PCN is a set of users inter-connected by payment channels [31]. When two users do not have a direct channel but there exists a path of channels connecting them, they can send payments to each other along the path, subject to the available balance on all channels along the path. To ensure payment security, a Hash Timelock Contract (HTLC) can be employed in LN, which locks the available balance on each channel until every channel confirms payment success or failure, or when the timelock expires [31]. Raiden network [9] utilizes smart contracts on Ethereum to control state changes of this process. To incentivize owners of intermediate channels for payment forwarding, each channel owner can charge a transaction fee based on the transaction amount when forwarding a payment.

Since invention, PCNs have received significant attention due to its ability to decrease latency, improve throughput and enhance privacy of the blockchain system. For example, as of January 2023, the Bitcoin LN already has $16,041$ nodes and $75,828$ open channels, with a total deposit of ₿$5,226$ (\$$109,122,968$) in the network [6]. To address challenges in PCN such as limited security provision, constrained usage scenario and functionality, and complete path reliance of payment, various innovative PCN architectures have been proposed. For instance, payment hub [27], state channel [38], virtual channel [20], general channel [11], etc.

B. Routing and Rebalancing in PCNs

We first review the categorization of payment routing schemes that focus on improving the payment success ratio. Then we introduce limitations of existing rebalancing schemes. We also list relevant works that control network balance through fee adjustments, and emphasize the differences and improvements offered by our approach compared to these existing solutions.

Payment routing's impact on PCN throughput and success ratio has been extensively studied [54], [60], [63]. Landmark routing is one promising approach in payment routing. Landmark routing picks several landmarks network-wide, and routes a payment via one of them [39], [43]. SpeedyMurmurs [46] extends landmark routing and uses embedding-based path discovery to reduce overhead and increase throughput. Yet, landmark-based routing is vulnerable to denial-of-service attacks due to centralization, and lacks path diversity.

Some proposed congestion control-based approaches to deal with the imbalance problem. Spider [49] was proposed to achieve higher throughput via payment slicing and queueing. However, slicing and/or queuing suffer from high transaction fees and long waiting and settlement times [13], [42]. Furthermore, Atomic Multipath Payment (AMP) [2] is required to ensure payment security, which could incur a huge overhead on routers for keeping per-slice states. APCN [61] mitigates channel balance depletion and enhances the success ratio with per-user congestion control, relying on hardware trusted execution environments to deter user misconduct [55].

Other routing schemes have also been explored. Bailout [24] can re-route ongoing multi-hop payments to allow earlier unlocking, but it does not focus on improving the payment success ratio and incurs additional interactions and overhead. Webflow [62] improves the payment success ratio by achieving high resource utilization, but it needs semi-centralized web servers for support and requires extra overhead for routers.

Some recent works focus on applying deep reinforcement learning to schedule transactions in order to maximize the long-term throughput [35] and deal with the channel imbalance [15], or to dynamically generate fee setting strategies [10]. However, learning-based solutions cannot satisfactorily learn accurate network conditions due to high dynamics and unknown information (such as balances) in PCN.

On-chain or off-chain rebalancing has been proposed to address balance depletion. On-chain rebalancing requires closing and reopening the channel, which involves time-consuming and expensive on-chain operations. Off-chain cycle-based rebalancing tries to find routing cycles to fulfill rebalancing requests [32]. The rebalancing process incurs transaction fees without completing any actual payment, and may make other channels more unbalanced. Also, it occupies balances on involved channels during rebalancing. Hence frequent off-chain rebalancing actually degrades throughput. Some existing work suggests using smart contracts for rebalancing [25], [29]. However, rebalancing through smart contracts requires extra deployment costs and interaction with the blockchain.

Most related to our work are several solutions on mitigating network imbalance with balance-aware fees or routing metrics [16], [19], [23], [33], [34], [45], [52]. However, they rely on heuristic fee functions or metrics without support from a theoretical framework. So they may only work in restricted settings and their efficacy can be inadequate in reality. For instance, Merchant [52] is a balance-aware fee setting scheme with a linear transaction fee function. OptimizedFees [19] applies a variable fee based on payment size and channel imbalance. FixedExpFee [45] introduces a fixed tunable parameter to reflect channel balance status in the fee function. We evaluated them in §VII, which performed inferiorly compared to ours.

Notably, most solutions above do not have any theoretical analysis or guarantee. Spider [49] is the only algorithm with a throughput-optimal guarantee in a restricted special case, but it requires breaking down all payments into unit amounts, which significantly modifies the current PCN architectures and incurs severe router overhead and high transaction fees. In contrast, our algorithm only results in minimal modification to user software, and does not rely on slicing payments to achieve an asymptotically tight competitive ratio in a unidirectional PCN.

III. System model

Table I summarizes notations used in our system model.

A. Network Model

We model a distributed PCN as a directed graph $G=(V,E)$, where V is the set of nodes, and E is the set of channels. A directed channel is denoted as $uv \in E$ with source node or

TABLE I
NOTATION TABLE OF SYSTEM MODEL

Symbol	Definition
G, V, E	Network topology, node set and channel set
$uv, \overline{uv}$	A direct channel from u to v, a bidirectional channel between u and v
$e', \overline{e}$	Opposite-direction and bidirectional channel of e
$c_{\overline{uv}}$	Capacity of channel $\overline{uv}$
$b(T, uv)$	Balance of node u on channel $\overline{uv}$ at time T
$\mathcal{R} = \{R_i\}$	Set of online arriving payments
$R_i = (s_i, d_i, \delta_i, st_i, ed_i)$	Payment i: (sender, recipient, payment amount, arrival time, expected completion time)
n	Maximum hop count of a valid payment path
$\mathcal{P}, \mathcal{P}_i, P_i$	Set of all paths with no more than n hops, set of paths between s_i and d_i, and subset of paths for R_i
p, l_p	A payment path, and the number of hops of the path
$\boldsymbol{\delta}_i, \boldsymbol{\delta}_i(e)$	Payment allocation function over P_i and payment amount processed along a directed channel e
$\mathcal{T}, \mathbb{T}$	Time slots and max payment duration estimation
$\tau(T, i), \kappa(T, i)$	Indicator of payment R_i arrival before time T and payment R_i being active at time T
$\phi_e(y), \varphi(p, \delta)$	Transaction fee function for amount y on channel e, the total transaction fee of amount δ on path p
$\rho_i, C_i, \mathbb{C}$	Valuation for payment R_i, constant coefficient for valuation ρ_i and the upper bound of C_i of all users

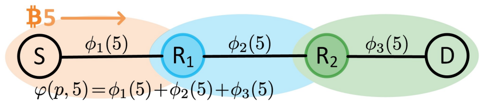
owner u. Two opposite-direction channels $uv, vu \in E$ form a bidirectional channel denoted as $\overline{uv} = \overline{vu} \triangleq \{uv, vu\}$, which is an unordered set. In general, we define e' as the opposite-direction channel of $e \in E$, $\overline{e}$ as the bidirectional channel that e belongs to, and $\overline{E}$ as the set of all bidirectional channels. The capacity of $\overline{uv}$ is $c_{\overline{uv}}$, denoting the total deposit of both owners at channel opening, and shared by both directed channels.

Consider uniform time slots $\mathcal{T} = (0, 1, \ldots, \mathbf{T})$. This uniform time slot assumption is only for simplicity of illustration, and can be trivially extended to systems without or with variable time slots. The instantaneous balance of a channel $uv \in E$ at time $T \in \mathcal{T}$ is $b(T, uv)$, denoting the available balance of node u on channel $\overline{uv}$. Let $b(0, uv)$ be the initial balance of uv upon channel opening. We call $\{b(T, uv), b(T, vu)\}$ a balance distribution of channel $\overline{uv}$. Note that a node's balance may or may not include in-flight transactions that are in processing. For clarity, we define $b(T, uv)$ to *include* all in-flight transactions on $\overline{uv}$. Based on Fig. 1, we have $b(T, uv) + b(T, vu) = c_{\overline{uv}}$ at any time.

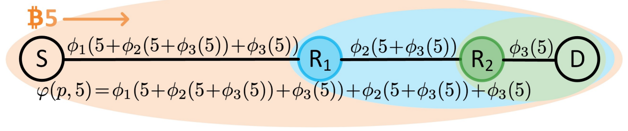
We assume that each PCN node knows the network topology from the blockchain [31], but only knows the balances of its own channels. This complies with real-world PCN implementations such as the Bitcoin LN [31], where the network topology is automatically kept by each user's client based on channel opening and closing transactions on the blockchain, while the real-time balance of each channel is private information and is not disclosed to a non-adjacent node. A node can be a sender/recipient of payments, a router forwarding payments for others, or both simultaneously.

B. Payment Model

Let $\mathcal{R} = \{R_1, \ldots, R_{\mathbf{K}}\}$ be a set of $\mathbf{K}$ online arriving payments, and define $\mathcal{K} = \{1, \ldots, \mathbf{K}\}$. Without loss of generality, we assume that the payments in $\mathcal{R}$ are ordered based on their arrival times. Each payment $R_i \in \mathcal{R}$ is denoted by a 5-tuple: $R_i = (s_i, d_i, \delta_i, st_i, ed_i)$, where s_i and d_i are the sender and recipient respectively, δ_i is the payment amount, and st_i and ed_i are the arrival time and *expected* completion time if the payment succeeds respectively. s_i, d_i, δ_i, st_i are known when



(a) Linear summation fee



₿5
S
$\phi_1(5+\phi_2(5+\phi_3(5))+\phi_3(5))$
R_1
$\phi_2(5+\phi_3(5))$
R_2
$\phi_3(5)$
D
$\varphi(p,5)=\phi_1(5+\phi_2(5+\phi_3(5))+\phi_3(5))+\phi_2(5+\phi_3(5))+\phi_3(5)$

(b) Convolutional fee

Fig. 2. Linear summation fee and convolutional fee setting with final payment amount ₿5. $\phi_i(\cdot)$ denotes the transaction fee function of ith hop. The total transaction fee for the payment is denoted by $\varphi(p,5)$. The shaded part denotes the transaction amount range that each node considers when calculating fees.

a payment is initiated, while ed_i can be estimated based on st_i and the average settlement time in the PCN.

Let n be the maximum hop count of a valid payment path, and let $\mathcal{P}$ be the set of all paths with no more than n hops. $\mathcal{P}_i \subseteq \mathcal{P}$ denotes the set of paths between s_i and d_i. A path is defined as $p \triangleq (e_1, e_2, \dots, e_{l_p})$, where l_p is the number of hops of path p. Let $\mathbb{T} \triangleq \max_{i\in\mathcal{K}}\{ed_i - st_i\}$ be the maximum duration estimation of one payment. We define $\tau(T,i) \triangleq \mathbf{1}_{st_i<T}$ to denote if the payment R_i has arrived before time T, and $\kappa(T,i) \triangleq \mathbf{1}_{st_i\le T\le ed_i}$ to denote if the payment R_i is *active* at time T by estimation, where $\mathbf{1_c}$ is an indicator function of condition $\mathbf{c}$.

Define a transaction fee function $\phi_e(y)\in\mathbb{R}^*$ for forwarding an amount of $y>0$ on channel e; $\mathbb{R}^*$ denotes the non-negative real number set. Let δ be the final payment amount received by the recipient along a path p. There are two ways for computing the transaction fee along the entire path p in the PCN literature:

- **Linear summation fee:** Total transaction fee $\varphi(p,\delta)$ is a linear summation of applying the fee function of each channel on the final payment amount δ along the path:
$$\begin{aligned}\varphi^{\mathsf{lin}}_{e_{l_p}}(p,\delta) &= \phi_{e_{l_p}}(\delta),\\ \varphi_{e_l}(p,\delta) &= \phi_{e_l}(\delta)+\varphi^{\mathsf{lin}}_{e_{l+1}}(p,\delta),\ \forall l<l_p,\\ \varphi^{\mathsf{lin}}(p,\delta) &= \varphi^{\mathsf{lin}}_{e_1}(p,\delta).\end{aligned} \tag{1}$$
- **Convolutional fee:** The transaction fee on channel $e\in p$ is computed by applying the fee function on the sum of the payment amount, plus the *accumulative transaction fee* on all channels after e along the path:
$$\begin{aligned}\varphi^{\mathsf{conv}}_{e_{l_p}}(p,\delta) &= \phi_{e_{l_p}}(\delta),\\ \varphi_{e_l}(p,\delta) &= \phi_{e_l}(\delta+\varphi^{\mathsf{conv}}_{e_{l+1}}(p,\delta))+\varphi^{\mathsf{conv}}_{e_{l+1}}(p,\delta),\ \forall l<l_p,\\ \varphi^{\mathsf{conv}}(p,\delta) &= \varphi^{\mathsf{conv}}_{e_1}(p,\delta).\end{aligned} \tag{2}$$

The *linear summation fee* has been widely used in existing work due to simple computation of end-to-end transaction fees [16], [19], [52]. However, in practical PCNs such as the LN, the *convolutional fee* is used [5], as $\varphi_e(p,\delta)$ would be the *actual transaction amount* that needs to be processed on each channel e, not just the final payment amount δ. We show two examples of fee settings with payment amount ₿5 in Fig. 2. In linear fee setting, each hop only needs to consider the final transaction amount in calculating the transaction that it will charge, as shown in Fig. 2(a). Each hop in the convolutional fee setting instead needs to consider the final transaction amount, as well as the transaction fee of all subsequent hops, as shown in Fig. 2(b). In practice, the first hop of the path usually does not charge the transaction fee, in which case the fee function can be temporarily set as $\phi_{e_1}(\cdot) = 0$ for this hop during routing. Routing based on the linear summation fee may lead to under-estimated transaction fees and hence frequent payment failures. We address both fee forms with a uniform algorithm framework.

When the transaction fee of making a payment through the PCN is too expensive, a sender will be motivated to use an alternative method such as the blockchain itself or a traditional payment method like credit card. In other words, each sender has an internal valuation $\rho_i = C_i\delta_i$ for payment R_i, which is the upper bound of the total transaction fee the sender is willing to spend for payment amount δ_i, where C_i is a constant coefficient. We assume that there is a global upper bound of the transaction fee coefficients of all senders, denoted as $\mathbb{C} \triangleq \max_{i\in\mathcal{K}}\{C_i\}$. It can be computed, for example, by dividing the average on-chain transaction fee and/or the transaction fee of using any traditional payment method by the payment amount. To properly formulate our payment routing problem, we first define a *routing scheme* of a successful payment $R_i \in \mathcal{R}$:

Definition 1. *Given payment $R_i \in \mathcal{R}$, a* routing scheme *for R_i is defined as a tuple $(P_i, \boldsymbol{\delta_i})$, where $P_i \subseteq \mathcal{P}_i$ is a subset of paths for R_i, and $\boldsymbol{\delta_i} : P_i \to \mathbb{R}^*$ is a* payment allocation function *over P_i that represents the payment amount allocated on each path in the set P_i. We then define $\boldsymbol{\delta}_i(e) \triangleq \sum_{p\in P_i: e\in p}\boldsymbol{\delta}_i(p)$ as the payment amount processed along a directed channel $e\in E$, and similarly $\boldsymbol{\delta}_i(\overline{e})$ as the amount along both directions of $\overline{e}\in\overline{E}$.*

IV. Online Routing Design

In this section, we first motivate and formulate the balance-aware network weighted throughput maximization problem. We then design an exponential fee function to instruct each router's transaction fee policy. Following the fee setting policy, we propose a distributed online algorithm for payment routing. We assume that each sender can obtain the real time fee setting policies of each router in this section and relax this assumption in §VI. Table II lists the notations used in this section.

A. Balance-aware Weighted Throughput Maximization

In a PCN, the transaction amount a channel can forward is limited by both its capacity and balance distribution. Capacity limits the maximum in-flight transaction amount on *both* directions of a channel, while balance limits the amount that can be forwarded along *each* direction. Meanwhile, capacity will be resumed when the in-flight transactions settle, while balance on a direction is affected by all transactions settled before a given time, and can only be recharged mostly via opposite-direction payments. Depletion of either resource will prevent a payment from being forwarded. Current PCNs with static fees and minimum-fee routing can lead to frequent depletion of some channels, while leaving others under-utilized.

Fig. 3 gives an example of channel balance depletion and underutilization. Suppose S is sending payments to D, and R_1 and R_2 are two routers that form paths $S\to R_1\to D$ and $S\to R_2\to D$ respectively. R_1 charges a lower fee than R_2. In this case, even if R_2 has more balance to support payments from S to D, S will still use $S\to R_1\to D$ as the default path and deplete

TABLE II
NOTATION TABLE OF ROUTING DESIGN

Symbol	Definition
$\overline{e}$	Chanel e's synthetic channel which ignores transaction fees in updates
β_e, α_e	Balance goal and liquidity parameter of channel e
$X = \{x_i\}, \Phi$	Outcome of routing payments $\mathcal{R}$ w.r.t each payment R_i's acceptance state, the demand allocation over all paths for each R_i
$f_{i,e}(p)$	Transaction amount incurred on $e \in p$ by routing R_i
$\boldsymbol{F} = (F_{st}, \ldots, F_{ed})$	The congestion state on a channel w.r.t. total in-flight amount on both directions of the channel at time $T \in [st, ed]$.
$\lambda_{1,\overline{e}}(T,i), \lambda_{2,e}(i)$	Capacity utilization of channel $\overline{e}$ at the time when the i-th payment arrives, the balance utilization of channel e at the time when the i-th payment arrives
$\sigma_{1,\overline{e}}(T,i), \sigma_{2,e}(i), \sigma_e(i)$	Congestion cost at each time T for any bidirectional $\overline{e}$, the imbalance cost for any directed $e \in E$ and channel e's unit cost upon a payment R_i's arrival
F_1, F_2	Two conservativeness parameters of the router in charge of fee setting
$\mathbb{Z}$	Global factor to match valuation and capacity usage
$\phi_e(\delta, \sigma_e(i))$	Adaptive fee function of channel e with payment amount δ, upon R_i's arrival
$\varphi_i^{\mathsf{lin}}(p,\delta)$, $\varphi_i^{\mathsf{lin}}(P_i, \boldsymbol{\delta}_i)$	Total fee for sending payment amount δ along path p and the total fee of a routing scheme $(P_i, \boldsymbol{\delta}_i)$ using linear fee setting
$\varphi_i^{\mathsf{conv}}(p,\delta)$, $\varphi_i^{\mathsf{conv}}(P_i, \boldsymbol{\delta}_i)$	Total fee for sending payment amount δ along path p and the total fee of a routing scheme $(P_i, \boldsymbol{\delta}_i)$ using convolutional fee setting

the channel between R_1 and D, leaving the channel between R_2 and D underutilized. After depletion, channel $R_1 \rightarrow D$ cannot be used anymore due to the lack of available balance on R_1's side, unless there are payments coming from $D \rightarrow R_1$ or some other rebalancing technique is used.

Given online arriving payments $\mathcal{R}$, a PCN tries to maximize the total amount of payments successfully settled. Motivated by the above example, each directed channel uv has a certain *perfectly balanced state*, which is the desired state of both channel owners. For instance, a channel with both sides having the same balance (equal to half of the channel capacity) can be regarded as perfectly balanced, if both sides expect to send an equal amount of transactions to each other in the long run. For generality, we define a *balance goal* of a channel uv with parameter $\beta_{uv} \in (0,1)$, which defines the balance distribution that is regarded as "perfectly balanced" for this channel. For instance, $\beta_{uv} = 0.5$ means the channel is perfectly balanced when each side has half of the total channel capacity as its balance: $b(T,uv) = b(T,vu) = 0.5c_{\overline{uv}}$. To simplify the description of the problem, we let $\beta_{uv} + \beta_{vu} = 1$. In practice, it is not necessary for the two owners of the channel to reach a consensus on the balance goal. Our approach remains applicable without consensus. In the case of $\beta_{uv} + \beta_{vu} < 1$, the liquidity parameter α_e can be adjusted to align with the balance goals of the channel owners. In the case of $\beta_{uv} + \beta_{vu} > 1$, our proposed fee setting scheme also guarantees that at least one owner's fee setting brings her closer to her desired balance goal. The owners of each channel also have requirements for liquidity. They have the motivation to maintain a certain amount of "reserved fund" on their channel for unexpected or urgent payments of their own. We define *liquidity requirement* of a channel with liquidity parameter $\alpha_e \in [0, \beta_e)$, denoting that the owner would like to keep at least $\alpha_e \cdot c_e$ minimum balance on channel e at any time. The channel owner allows full depletion if $\alpha_e = 0$.

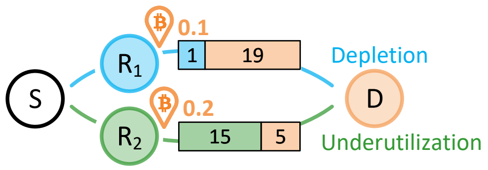


Fig. 3. Depletion and underutilization of payment channels.

Let $X = \{x_i \in \{0,1\} \mid R_i \in \mathcal{R}\}$ be the outcome of routing payments $\mathcal{R}$, where x_i indicates if payment R_i is accepted or not. Let $\Phi = \{\boldsymbol{\delta}_i(p) \in \mathbb{R}^* \mid R_i \in \mathcal{R}, p \in \mathcal{P}_i\}$ be the demand allocation over all paths for each R_i. Define the transaction amount incurred on $e \in p$ by routing R_i along path p as

$$f_{i,e}(p) = \boldsymbol{\delta}_i(p) + \varphi_e(p, \boldsymbol{\delta}_i(p)). \tag{3}$$

$f_{i,e}(p)$ is 0 when R_i is not accepted, i.e., when $x_i = 0$. The balance $b(T,e)$ on channel e at time T by considering all previously arrived payments before time T is:

$$b(T,e) = b(0,e) + \sum_{i \in \mathcal{K}} \tau(T,i) x_i \sum_{p \in \mathcal{P}_i} (f_{i,e'}(p) - f_{i,e}(p)). \tag{4}$$

Rather than purely optimizing long-term throughput (sum of successful payment amounts), we instead maximize a *weighted* sum, taking into account the potentially different valuation of each sender. This *generalizes* the throughput maximization problem, as throughput maximization can be formulated as weighted throughput maximization with all valuations $C_i = 1$.

Definition 2. *Given a PCN G and payment set $\mathcal{R}$, the* ***balance-aware weighted throughput maximization*** *problem can be formulated as:*

$$\max_{X,\Phi} \quad \sum\nolimits_{i \in \mathcal{K}} C_i \delta_i x_i \tag{5}$$

$$\text{s.t.} \quad \sum\nolimits_{p \in \mathcal{P}_i} \boldsymbol{\delta}_i(p) \geq \delta_i x_i, \quad \forall i \in \mathcal{K}; \tag{5a}$$

$$\sum\nolimits_{i \in \mathcal{K}} \kappa(T,i) \sum\nolimits_{p \in \mathcal{P}_i} (f_{i,e}(p) + f_{i,e'}(p)) \leq c_e, \quad \forall T \in \mathcal{T}, \forall e \in E; \tag{5b}$$

$$b(T,e) \geq \alpha_e c_e, \ \forall T \in \mathcal{T}, \forall e \in E. \tag{5c}$$

Explanation: Our objective is to maximize the weighted throughput that is the sum of valuations of all accepted payments based on Eq. (5). To purely maximize total throughput, one can simply assume C_i is the same for $\forall R_i \in \mathcal{R}$. Constraint (5a) means the total received amount of a payment should be no less than its payment amount if it has been served successfully. In other words, payments are either fully accepted or fully rejected in an atomic manner. Constraint (5b) limits the total in-flight transaction amount on each channel by its capacity. The more saturated this constraint is, the more "congested" the channel is at a time. Constraint (5c) enforces the liquidity requirement of each owner by ensuring that its balance is lower bounded by $\alpha_e c_e$.

B. Fee-based Online Routing

To solve the weighted throughput maximization problem in PCN, we propose a fee-based online routing algorithm. Below, we first outline and motivate the high-level idea of fee-based online routing, and then propose our detailed design.

1) Online Routing based on Fee Setting: The transaction fee was invented to incentivize PCN participants to forward the payment and compensate for their opportunity cost, which improves the network stability. The transaction fee of the channel can be set freely by the channel owners in today's

LN [5]. It would be beneficial if the transaction fee could reflect the network states, such as the channel's congestion and available liquidity, in order to improve routing performance. Specifically, the dynamically changing fee setting of each node will affect the routing strategy of each payment, thereby impacting the congestion or imbalance level of the channels in the entire network.

Motivation: Routing design via fee setting has several advantages. First, senders are assumed to employ the same routing strategy as in the current PCN, *i.e.*, they are incentivized to take the minimum-cost paths. In comparison, most existing solutions require the senders to follow specific (non-cost-efficient) routing strategies, such as using K minimum-cost or non-minimum-cost paths, contradicting the users' default behaviors. Second, routing via fee setting requires minimal changes to routing algorithms of senders, and modification to router behaviors is only limited to fee setting with minimal overhead. Moreover, it requires no central coordination, as every router is able to set its transaction fee function independently based on local observations.

2) Solution Overview: Since the PCN does not know future payments before they arrive, finding the optimal strategy for maximizing long-term weighted throughput is difficult. Our idea is to manipulate transaction fees to motivate payment routing that leads to long-term network balancedness and reduces congestion. Picking a low-fee path should correspond to using channels with less congestion and imbalance. If a path has a fee higher than what a sender is willing to spend, it should indicate that the path likely has insufficient balance or is severely congested.

To achieve this goal, we need an *adaptive fee setting policy* that reflects the channel status for congestion and balance. Below, we define two desired properties of such a policy.

Consider a payment $R = (s, d, \delta, st, ed)$ arrives at a channel e. We define the level of congestion that R faces on a channel during the entire period $[st, ed]$ as the *congestion state*, denoted by $\boldsymbol{F} = (F_{st}, F_{st+1}, \dots, F_{ed})$ where each F_T denotes the *total in-flight amount* on both directions of channel $\overline{e}$ at time $T \in [st, ed]$. When comparing two congestion states $\boldsymbol{F}$ and $\widehat{\boldsymbol{F}}$, we say that $\boldsymbol{F} \succ \widehat{\boldsymbol{F}}$ iff $F_T \geq \widehat{F}_T$ for $\forall T \in [st, ed]$, and there exists $T \in [st, ed]$ such that $F_T > \widehat{F}_T$.

Definition 3. *Consider two congestion states $\boldsymbol{F}, \widehat{\boldsymbol{F}}$ and the same balance distribution on channel e, in which a payment is charged with two transaction fees $\phi_e, \widehat{\phi}_e$ respectively. Then, the fee policy is* ***decongestion-incentive*** *if*

$$\boldsymbol{F} \succ \widehat{\boldsymbol{F}} \quad \Rightarrow \quad \phi_e > \widehat{\phi}_e. \tag{6}$$

Informally, an owner should charge a higher fee when the in-flight amount on both directions of the channel is higher.

Definition 4. *Consider a payment that can go through either direction of a bidirectional channel (e or e') when it arrives. With the same congestion state on $\overline{e}$, consider two balance distributions $\{b_e, b_{e'}\}$ and $\{\widehat{b}_e, \widehat{b}_{e'}\}$, under which the fees charged by e and e' are $\{\phi_e, \phi_{e'}\}$ and $\{\widehat{\phi}_e, \widehat{\phi}_{e'}\}$ respectively. Then, the fee policy is* ***balance-incentive*** *if*

$$b_e \leq \widehat{b}_e \quad \Rightarrow \quad \phi_e \geq \widehat{\phi}_e, \tag{7}$$

$$b_e/\beta_e \leq b_{e'}/\beta_{e'} \quad \Rightarrow \quad \phi_e \geq \phi_{e'}, \tag{8}$$

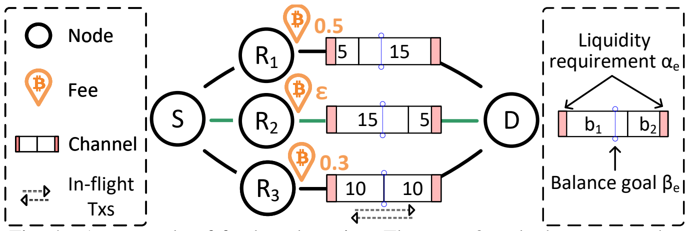


Fig. 4. An example of fee-based routing. There are 3 paths between sender S and recipient D: path $p_1 = (S \to R_1 \to D)$, path $p_2 = (S \to R_2 \to D)$, and path $p_3 = (S \to R_3 \to D)$. The sender will choose path p_2 because the channel from R_2 to D charges a lower fee ε than others.

and additionally, only a negligible fraction of balance distributions would result in strict equalities *in both* (7) *and* (8).

Informally, a balance-incentive fee policy satisfies that: 1) a higher fee is charged when an owner has a lower balance on its side, and 2) the owner with a balance lower than the balance goal $\beta_e c_e$ charges a higher fee than the other owner. Definitions 3 and 4 will be utilized in Lemma 1 to demonstrate how our designed adaptive fee function achieves these two properties.

Fig. 4 shows the intuition of the fee-based routing. Assume all channels have a capacity of ₿20, and the balance goal is $\beta_e = 0.5$. There are three paths p_1, p_2 and p_3 from sender S to recipient D. For p_1, since the channel from R_1 to D is unbalanced with R_1 having balance lower than its goal, a new payment through it will exacerbate its imbalance. So R_1 would set a high fee (₿0.5). For p_3, there are ongoing transactions on this channel even though this channel is currently in a balanced state. So R_3 would set a high fee (₿0.3) to alleviate possible congestion. For p_2, because R_2 to D is unbalanced with R_2's side having more balance, R_2 can set a low fee (*e.g.*, a minimal fee ₿ε for R_2) to attract payments from R_2 to D. S will choose the minimum-fee path p_2, which will avoid aggravating the imbalance and congestion on p_1 and p_3 respectively, and help the channel $R_2 \to D$ be more balanced.

3) Fee Design: Inspired by the online algorithm framework in [12], we design an *exponential fee function* to instruct routing in a PCN. Different from in traditional networks, online routing in a PCN has two unique challenges. First, the fee and the payment amount itself share the channel balance and capacity, and both need to be considered in the online framework. Second, besides occupying the capacity on each channel which will resume after the payment completes, a successful payment will impact the channel balance even after the payment is completed, continuously affecting all future payments. The fees need to reflect this long-lasting influence.

To measure the temporary capacity congestion and long-term (im)balance of a channel, we define two utilization ratios at the time when the i-th payment in $\mathcal{R}$ arrives:

$$\lambda_{1,\overline{e}}(T, i) \triangleq \sum\nolimits_{j<i} \kappa(T, j) x_j \frac{\boldsymbol{\delta}_j(\overline{e})}{c_{\overline{e}}}, \quad \forall T \in \mathcal{T}, \forall \overline{e} \in \overline{E}; \tag{9}$$

$$\lambda_{2,e}(i) \triangleq \frac{\beta_e c_e - b^*(st_i, e)}{(\beta_e - \alpha_e) c_e}, \quad \forall e \in E, \tag{10}$$

where $b^*(T, e) \triangleq b(0, e) + \sum_{i \in \mathcal{K}} \tau(T, i) x_i (\boldsymbol{\delta}_i(e') - \boldsymbol{\delta}_i(e))$.

$\lambda_{1,\overline{e}}(T, i)$ is the *capacity utilization* w.r.t. constraint (5b), which measures the level of congestion based on payments that are active at time T. $\lambda_{2,e}(i)$ is the *balance utilization* w.r.t. constraint (5c), which measures the level of imbalance w.r.t. the balance goal $\beta_e c_e$, based on the current balance that depends on all payments before st_i. In the definition of $\lambda_{2,e}(i)$, the numerator defines *how much the current balance* $b^*(st_i, e)$ *is short from the balance goal* $\beta_e c_e$, and the denominator defines the total balance budget as the balance goal $\beta_e c_e$ minus the minimum balance requirement $\alpha_e c_e$. Clearly $\lambda_{1,\overline{e}}(T,i) \in [0,1]$, while $\lambda_{2,e}(i) \le 1$ and can be a negative value, meaning the channel's balance is exceeding the balance goal.

Our fee function is based on the utilization ratios. Recall that n is the maximum hop count of a payment. We define a *congestion cost* $\sigma_{1,\overline{e}}(T,i)$ at each time T for any bidirectional $\overline{e} \in \overline{E}$ (Eq. (11)) and an *imbalance cost* $\sigma_{2,e}(i)$ for any directed $e \in E$ (Eq. (12)) upon payment R_i's arrival:

$$\sigma_{1,\overline{e}}(T,i) \triangleq \mathbb{Z} c_{\overline{e}} (\mu_1^{\lambda_{1,\overline{e}}(T,i)} - 1), \tag{11}$$

$$\sigma_{2,e}(i) \triangleq \begin{cases} \mathbb{Z}(\beta_e - \alpha_e) c_e (\mu_2^{\lambda_{2,e}(i)} - 1), & \text{if } \lambda_{2,e}(i) \ge 0, \\ 0, & \text{if } \lambda_{2,e}(i) < 0, \end{cases} \tag{12}$$

where $\mu_1 = 2(n\mathbb{T}F_1 + 1)$ and $\mu_2 = 2(nF_2 + 1)$, F_1 and F_2 are two conservativeness parameters of the router in charge of fee setting, and $\mathbb{Z}$ is a global factor to match the valuation and the usage of capacity, in other words, a *basis rate* for congestion- or imbalance-incurred costs. We discuss how to set these parameters in Sec. VI. Note that $\lambda_{1,e}(T,i)$ and $\sigma_{1,e}(T,i)$ are shared by both directions of $\overline{e} \in \overline{E}$. Both cost functions are exponential with respect to their respective utilizations. As the congestion or imbalance level of channel e increases, congestion cost or imbalance cost rapidly escalates. This constrains the passage of transactions exacerbating this situation, pushing transactions to utilize other channels with lower congestion and imbalance. The coefficients of the cost functions indicate the extent to which the usage of capacity affects their respective costs. When the capacity utilization is 0, the congestion cost is 0; when the balance utilization is 0, the imbalance cost is 0. Based on these costs, we define the channel unit cost upon a payment R_i's arrival as:

$$\sigma_e(i) \triangleq \sum\nolimits_{T=st_i}^{ed_i} \frac{\sigma_{1,\overline{e}}(T,i)}{c_e} + \frac{\sigma_{2,e}(i)}{(\beta_e - \alpha_e)c_e}, \ \forall e \in E. \tag{13}$$

Based on Eq. (13), we propose the following ***adaptive fee function*** of channel e with payment amount δ, upon R_i's arrival, taking the instantaneous cost of the channel as input:

$$\phi_e(\delta, \sigma_e(i)) \triangleq \sigma_e(i) \cdot \delta. \tag{14}$$

Considering the *linear summation fee* defined in Eq. (1), the fee for sending payment amount δ along path p is:

$$\varphi_i^{\mathsf{lin}}(p, \delta) \triangleq \sum\nolimits_{e \in p} \phi_e(\delta, \sigma_e(i)), \tag{15}$$

and the total fee of a routing scheme $(P_i, \boldsymbol{\delta}_i)$ is $\varphi_i^{\mathsf{lin}}(P_i, \boldsymbol{\delta}_i) \triangleq \sum_{p \in P_i} \varphi_i^{\mathsf{lin}}(p, \boldsymbol{\delta}_i(p))$. Eq. (15) comes from the definition in Eq. (1). Similarly, define the fees under the *convolutional fee* for sending payment δ along p and for a routing scheme $(P_i, \boldsymbol{\delta}_i)$, as $\varphi_i^{\mathsf{conv}}(p, \delta)$ and $\varphi_i^{\mathsf{conv}}(P_i, \boldsymbol{\delta}_i)$ respectively, by using the definition in Eq. (2) with Eq. (14).

Lemma 1 rigorously proves that our proposed fee function is designed to assign higher fees in the presence of increased channel congestion and greater imbalance, and vice versa. Consider a synthetic channel $\widetilde{e}$ which has the same capacity and initial balance as e, but the channel updates only consider the payment amounts without transaction fees, *i.e.*, the in-flight amount is $F_{\widetilde{e}} = \sum_{j<i} \kappa(T,j) \boldsymbol{\delta}_j(\overline{e}) x_j$ at time T upon R_i's arrival, and the balance is $b_{\widetilde{e}} = b^*(st_i, e)$.

Lemma 1. *The fee function defined by Eq.* (14) *is decongestion-incentive and balance-incentive on channel* $\widetilde{e}$.

Proof. As the capacity utilization in Eq. (9) increases with the total in-flight payment amount, the congestion cost $\sigma_{1,\widetilde{e}}(T,i)$ is increasing. Hence when channel balances are fixed, the fee function $\phi_{\widetilde{e}}(\delta, \sigma_{\widetilde{e}}(i))$ is *decongestion-incentive*.

When $\lambda_{2,\widetilde{e}}(i) < 0$, according to Eq. (12), we have imbalance cost $\sigma_{2,\widetilde{e}}(i) = 0$. When $\lambda_{2,\widetilde{e}}(i) \ge 0$, the imbalance cost $\sigma_{2,\widetilde{e}}(i)$ strictly increases with decreasing $b^*(st_i, \widetilde{e})$. Hence Eq. (7) holds given the same congestion state.

For Eq. (8), if $b^*(st_i, \widetilde{e})/\beta_{\widetilde{e}} \le b^*(st_i, \widetilde{e}')/\beta_{\widetilde{e}'}$, we have $b^*(st_i, \widetilde{e}) \le \beta_{\widetilde{e}} c_{\widetilde{e}}$ and $b^*(st_i, \widetilde{e}') \ge \beta_{\widetilde{e}'} c_{\widetilde{e}}$. Then according to Eq. (10), we have $\lambda_{2,\widetilde{e}}(i) \ge 0$ and $\lambda_{2,\widetilde{e}'}(i) \le 0$. So we have $\lambda_{2,\widetilde{e}}(i) \ge \lambda_{2,\widetilde{e}'}(i)$ and hence Eq. (8) holds given the same congestion state. Both equalities hold only when the channel $\widetilde{e}$ is perfect balanced. The lemma follows. ☐

We note that channels e and $\widetilde{e}$ differ only by the transaction fee of each payment. This may make the fee function $\phi_e(\cdot)$ slightly violate the incentive properties on channel e. However, the violation is bounded by a factor of $(1 + \mathbb{C})$ because of the maximum transaction fee bound $\mathbb{C}$, which can be very small in practice (*e.g.*, $\le 0.015\%$), as we further discuss in Sec. VI.

4) Online Routing Algorithm Design: We propose a scalable distributed online fee setting and routing algorithm in Algorithm 1.

Sender's algorithm is in Lines 1–5. Upon a payment's arrival, the sender finds a minimum-fee path based on either the linear or convolutional fee, *e.g.*, using Dijkstra's algorithm or its variant. The sender then checks if the transaction fee of the path for its *full payment* is lower than its valuation. If the transaction fee is higher, the sender will choose not to use the PCN for this payment, but instead use alternative means for payment such as the blockchain itself. If it happens that the transaction fee of PCN is higher than the blockchain, it means the network is undergoing significant congestion and imbalance where almost no payment may succeed, in which case using the blockchain can actually be a better option for the user. Otherwise, the sender will attempt to send the payment along the minimum-fee path.

Each router's algorithm is in Lines 6–13. Lines 6–7 initialize the utilizations and fee functions. When a router forwards a payment, it updates channels' *capacity* and *balance utilizations* used by this payment, and sets the fee function accordingly.

Remark: Algorithm 1 routes each payment along a *single path*, despite that the routing problem (Program 5) and our fee function in Eq. (15) both consider possible multi-path payments with demand allocation. The consideration of multi-path payments are for compatibility with existing PCN protocols, and/or users and routers who do not follow our algorithm for routing. As we show later, single-path payments are sufficient for achieving a non-trivial competitive ratio under a special circumstance, and show good performance in simulations.

V. Online Routing Algorithm Analysis

Our online routing algorithm is theoretically analyzed in this section utilizing the *competitive analysis* framework.

Algorithm 1: Online Routing with Fee Functions

```
   /* Sender algorithm */
   Input: Network $G$, payment $R_i$, function $\varphi \in \{\varphi_i^{\mathsf{lin}}, \varphi_i^{\mathsf{conv}}\}$
   Output: Decision $x_i$, routing path $p$ if $x_i = 1$
1  Find min-fee path $p^* \leftarrow \text{argmin}_{p \in \mathcal{P}_i}\{\varphi(p, \delta_i)\}$;
2  if $p^* \neq \emptyset$ and $\varphi(p^*, \delta_i) \leq C_i\delta_i$ then
3      $\delta_i(p^*) \leftarrow \delta_i$;
4      return Send $R_i$ ($x_i = 1$) along path $p = p^*$.
5  else return Reject ($x_i = 0$).
   /* Router algorithm */
   Input: Bidirectional channel $\overline{uv}$
   Output: Channel fee announcements
6  Initialize $\lambda_{T,\overline{uv}}(0, 1)$, $\lambda_{2,e}(1)$ for $\forall e \in \overline{uv}$ according to
     Eqs. (9) and (10)
7  Nodes $u, v$ set and broadcast $\phi_e(\cdot)$ for $\forall e \in \overline{uv}$ based
     on Eqs. (11)–(14)
8  while payment $R_i$ arrives along $e \in \overline{uv}$ do
9      for $\forall T \in [st_i, ed_i]$ do
10         $\lambda_{1,\overline{uv}}(T, i+1) \leftarrow \lambda_{1,\overline{uv}}(T, i) + \frac{\delta_i}{c_{\overline{uv}}}$
11     $\lambda_{2,e}(i+1) \leftarrow \lambda_{2,e}(i) + \frac{\delta_i}{(\beta_e - \alpha_e)c_e}$
12     $\lambda_{2,e'}(i+1) \leftarrow \lambda_{2,e'}(i) - \frac{\delta_i}{(\beta_{e'} - \alpha_{e'})c_e}$
13     Nodes $u, v$ set and broadcast $\phi_e(\cdot)$ for $\forall e \in \overline{uv}$.
```

Competitive analysis has been widely used in analyzing the performance of online algorithms, for example, in resource allocation [53], online routing [30], scheduling [48], edge computing [37], etc. Competitive analysis assesses an online algorithm's performance with no complete future knowledge, by comparing it to an optimal offline algorithm that can anticipate all the requests in advance. Competitive ratio indicates the worst-case performance of the online algorithm [14].

Definition 5. *An online algorithm is (a, b)-competitive ($a, b \geq 1$) if given any sequence of online arriving payments $\mathcal{R}$, it achieves at least $1/a$ of the optimal weighted throughput, while ensuring that Eqs.* (5b) *and* (5c) *are violated by at most a factor of b.*

We first show that our algorithm achieves an ***asymptotically tight competitive ratio*** with a constant violation in a special case, *i.e.*, when the network consists of only unidirectional channels. Note that none of the existing works mentioned in Sec. II-B provided any theoretical guarantee even in this special case. Then we prove a negative result for the competitive ratio of any online algorithm in the general bidirectional PCN setting to highlight the difficulty of the problem.

A. Competitive Analysis in a Unidirectional PCN

In this subsection, we analyze the performance of our algorithm in a special case, where the PCN consists of only unidirectional channels. In this case, rebalancing through opposite-direction transactions is impossible. We analyze the competitive ratio of unidirectional PCN to guide our algorithm design and provide the theoretical guarantee. This is meaningful given the absence of such guarantees even in a unidirectional PCN from existing works. This analysis provides invaluable insights into the design of effective and efficient algorithms for optimizing network balance, given that the real-world depletion issues are most commonly caused by temporary unidirectional transaction flows on bidirectional channels.

In the case of a unidirectional channel, the channel's initial balance is always equal to its full capacity, and is monotonically decreasing with ongoing transactions due to the inability to rebalance. The competitive ratio analysis under PCN takes into account both instant capacity and long-term balance limitations. Specifically, the analysis considers the specific challenge in PCN where the total transaction amount settled before a time t affects the balances of the bidirectional channel for all times after t. This distinguishes the analysis from traditional network routing, where bandwidth consumption is only temporary and has no long-term impact beyond a traffic flow's duration. Let p_i be the payment path chosen by our routing algorithm for payment R_i with $\boldsymbol{\delta}_i(p_i) = \delta_i$. To facilitate our analysis, we make two assumptions on $\forall R_i \in \mathcal{R}$:

Assumption 1. $\mathbb{Z}n\mathbb{T} \leq C_i \leq \mathbb{Z}n\mathbb{T}F_1 + \mathbb{Z}nF_2$.

Assumption 2. $\delta_i \leq \min\left\{\frac{\min_{e\in E}\{c_e\}}{\log_2 \mu_1}, \frac{\min_{e\in E}\{(\beta_e - \alpha_e)c_e\}}{\log_2 \mu_2}\right\}$.

Recall that $\mathbb{Z}$ is a basis rate factor to match the valuation and the usage of capacity, C_i is the constant coefficient for valuation of payment R_i, F_1 and F_2 are conservativeness parameters of the router w.r.t fee setting, and $\mu_1 = 2(n\mathbb{T}F_1 + 1)$ and $\mu_2 = 2(nF_2 + 1)$. In short, Assumption 1 bounds the range of each sender's valuation, such that any single sender cannot have a valuation that is too high or too low compared to others. Assumption 2 upper bounds each payment's amount, *i.e.*, the payment amount of each payment cannot be too large to easily saturate the channel. Note that we allow the total payment amount of all users to significantly exceed the total capacity of the network, and our algorithm still outperforms state-of-the-art algorithms, as shown in the evaluation.

Remark on assumptions: We make these two assumptions to facilitate our theoretical analysis. For the algorithm to work in reality, these assumptions do not need to hold strictly. Notably, these assumptions provide guidelines for setting core parameters in our algorithm. These parameters can further be fine-tuned during the actual operation by each router to better reflect the network condition. In evaluation, we show results where these assumptions do not hold while our algorithm still achieves superior performance compared to state-of-the-art algorithms. In Sec. VI, we will thoroughly discuss how to set parameters based on guidelines from these assumptions.

In the following, we divide our analysis into three parts: *capacity and balance constraint violation*, *total weighted throughput bounds*, and *competitive ratio*. We first present capacity and balance constraint violation in Lemmas 2 and 3, and then present total weighted throughput bounds in Lemmas 4 and 5. Finally, we wrap up all analysis into Theorems 1 and 2 showing the asymptotically tight competitive ratio of our algorithm. Detailed proofs of these results are in Appendix IX.

1) Capacity and balance constraint violation: In the following, let $\mathcal{A}$ denote the set of payments that senders decide to send based on Algorithm 1. We have Lemma 2 which shows that the total payment amount (without fees) accepted by our algorithm will not exceed either the capacity or the balance constraint, despite the fact that the sender makes routing decision *without any knowledge about the balance or* *capacity of each channel, but only the fee.*

Lemma 2. *For $\forall e \in E$ and $\forall T \in \mathcal{T}$, two inequalities hold: $\sum_{i\in\mathcal{A}} \kappa(T,i)\delta_i \leq c_e$, and $\sum_{i\in\mathcal{A}} \tau(T,i)\delta_i \leq (\beta_e - \alpha_e)c_e$.*

Lemma 2 shows that if the balance or capacity constraint of a channel is violated by accepting a payment, the payment sender's valuation must be strictly less than the transaction fee on this channel. This contradicts with our online routing algorithm, as the sender would then not choose to send the payment because of the high fee. Based on Lemma 2, we next show that each balance or capacity constraint can be violated by up to a constant factor in our algorithm.

Lemma 3. *The transaction amount handled on a channel can violate constraint* (5b) *or* (5c) *by at most a factor of* $(1+\mathbb{C})$.

Lemma 3 proves the maximum violation of capacity or balance constraint for our algorithm by showing that the transaction amount is also bounded since the transaction fee is bounded by the valuation.

2) Total weighted throughput bounds: In the following, we further prove the total weighted throughput achieved by our algorithm is within a poly-logarithmic factor of the total weighted throughput of an optimal offline algorithm that knows all the future payments in advance. The following proof consists of four pieces: Lemma 4 gives a *lower bound* on our algorithm's total weighted throughput, and Lemma 5 gives an *upper bound* of the offline optimal algorithm's total weighted throughput. Combining all the results of Lemmas 3−5, Theorem 1 gives the competitive ratio of our algorithm and Theorem 2 shows the competitive ratio is asymptotically tight.

Let k be the index of the last payment in $\mathcal{A}$, and $\Gamma_e^i = \sum_{T\in\mathcal{T}} \sigma_{1,e}(T,i) + \sigma_{2,e}(i)$ be the total cost of using up all the resources on e. Lemma 4 proves a *lower bound* on our algorithm's total weighted throughput, by the final costs in the network after accepting all payments in $\mathcal{A}$.

Lemma 4. $2\log_2(\mu_1\mu_2)\sum_{i\in\mathcal{A}} \rho_j \geq \sum_{e\in E} \Gamma_e^{k+1}$. (16)

Through induction and showing that the changes in both congestion cost and imbalance cost of two adjacent payments are bounded by the total weighted throughput, we get the lower bound of the total weighted throughput. Below, Lemma 5 proves a total weighted throughput *upper bound* that an *offline optimal algorithm* can achieve, by the same costs in Lemma 4.

Lemma 5. *Let $\mathcal{A}^*$ be the set of payments accepted by an optimal offline algorithm, and let $\mathcal{Q} = \mathcal{A}^* \setminus \mathcal{A}$ be the set of payments served by the offline algorithm but not by our online algorithm. Then it satisfies that $\sum_{i\in\mathcal{Q}} \rho_i \leq \sum_{e\in E} \Gamma_e^{k+1}$.*

For the payments that are accepted by the offline algorithm but not accepted by the online algorithm, their valuations are bounded by the path costs. Because the costs in unidirectional PCN increase monotonically and the offline algorithm balance utilization cannot exceed 1, the total weighted throughput of the optimal offline algorithm is also bounded.

3) Competitive ratio: Based on Lemmas 3-5, we get the competitive ratio of our algorithm.

Theorem 1. *Algorithm 1 is a $(O(\log n\mathbb{T}), 1+\mathbb{C})$-competitive algorithm for the weighted throughput maximization problem in Program* (5).

Further in Theorem 2, we show that this competitive ratio is in fact asymptotically tight.

Theorem 2. *In a unidirectional PCN, any online algorithm has competitive ratio of $\Omega(\log n)$.*

Theorem 2 shows that for any online algorithm, we can always design a special payment sequence that dividing the payments that passed through multiple intermediate nodes at the previous time into multiple payments with the same weighted throughput between multiple intermediate nodes. So that the weighted throughput of the offline algorithm is $\Omega(\log n)$ times that of the online algorithm in a unidirectional PCN. This is because that the offline algorithm can consider all possible future payment sequences to select the optimal strategy, whereas the online algorithm cannot.

B. Infinite Competitive Ratio in a General PCN

For the general bidirectional PCN, the problem becomes increasingly hard. Theorem 3 states that without making any assumption, there is no online algorithm that can achieve a finite competitive ratio in a general bidirectional PCN.

Theorem 3. *Given any online algorithm* $\mathbf{A}$, *and an arbitrarily large $\boldsymbol{a} > 0$, there exists a sequence of payments $\mathcal{R}$, such that the competitive ratio of the algorithm $\mathbf{A}$ on $\mathcal{R}$ is at least $\boldsymbol{a}$.*

VI. Discussions on Protocol Design

This section discusses how to set algorithm parameters and explains the motivation behind a router following our algorithm for fee setting and how to address the capacity violation. This section also gives potential solutions for balance privacy.

Parameter setting. While our theoretical analysis assumes global knowledge such as all senders' valuations and future payment amounts, in practice, the parameters α_e, β_e, F_1, F_2, $\mathbb{C}$, $\mathbb{Z}$ and $\mathbb{T}$ can be independently decided by each router based on its own preferences and historical observations.

Specifically, balance goal parameter β_e can be negotiated by channel owners upon opening. The liquidity requirement α_e is set based on a channel owner's preference. $\mathbb{C}$ and $\mathbb{Z}$ relate to the willingness of senders to pay via the PCN. Each router stores all payments it receives. While valuations $\{C_i\}$ are senders' private information, an empirical upper bound of each C_i is to divide the current *on-chain transaction fee* by the payment amount δ_i. Indeed, if the off-chain fee exceeds the on-chain fee, senders may be encouraged to make on-chain payments instead. $\mathbb{C}$ and $\mathbb{Z}$ can then be empirically estimated, the latter based on the left-hand side of Assumption 1. $\mathbb{T}$ can be estimated based on recent successful payments by a router.

Parameters F_1 and F_2 are two conservativeness parameters set by each router. The right-hand side of Assumption 1 gives a baseline for setting the values of F_1 and F_2 based on estimated valuations. Based on it, their values can be scaled up or down. The higher value F_1 or F_2 has, the more the channel owner believes that congestion or imbalance will aggravate in the future, and that a higher fee should be applied in accordance. These values can be adaptive to how busy the PCN is based on demand estimations, which is a future work for us.

Fee announcement. In practice, each channel owner can estimate the average fee for the next estimated period of $\mathbb{T}$, based on changes in the levels of congestion and imbalance in the recent past. A window-based method can be used to update and announce the fee based on a minimum window.

The fee updates can be broadcast to all senders via distributed protocols such as link-state announcements (LSAs) [41] or distance vector protocols [44]. Alternatively, centralized services can be used to provide up-to-date fee information, similar to the *directory servers* in Tor [8].

Router motivation and capacity violation handling. We observe that current payment services such as Visa charges around 2.5% of the payment amount as fees [7]. Instead, the current LN proportional transaction fee has a median of 0.015%, two orders of magnitude lower than the traditional service. Hence having a transaction fee upper bound of $\mathbb{C} = 0.015\%$ or slightly higher in our algorithm does not remove any of the cost-efficiency benefit of PCN, but is able to result in greatly improved payment success ratio and throughput, as shown in our evaluations. In practice, the fee can be set either smaller or larger, as long as it is defined as an exponential function of the channel utilization in terms of both imbalance and congestion. For simplicity, a router can simply set its parameters to make sure the average fee it charges is approximately the same as its current fee without our mechanism, based on estimated network congestion/imbalance. The low transaction fee upper bound also implies that the balance/capacity violation factor $(1 + \mathbb{C})$ is almost negligible, and can be easily handled by leaving an additional $\mathbb{C} \cdot c_e$ margin on each channel. Our fee function may result in a zero fee when a channel has enough balance and is idle. To cover up routers' costs, a small base fee can be set (as in the current LN) on top of the exponential fee function. In real PCN, the fee policies of all channels are known to all of the nodes based on standard fee announcement scheme [5].

As a new technology, the growth of the user base for LN depends on the long-term liquidity and success of payments [57]. As a rational router, rather than solely focusing on maximizing its own income, it is motivated to maintain good liquidity in its channels in order to make the network more sustainable and attract more users. It also wants to maintain a high success ratio for payments, so that others will trust its service and use it as a intermediate node [36]. Therefore, in addition to just earning a maximum award in a short amount of time before channel depletion, it is important for a channel to consider its reputation and maintain liquidity. Additionally, in Sec. VII, we conducted a study on the behavior of routers when they followed the fee setting function versus when they intentionally set lower fees to attract more users and increase their own income. The results showed that when a small fraction of routers acted dishonestly, the selfish behavior harmed selfish routers' income instead of benefiting them, while having a negligible impact on the overall network throughput.

We note that despite some proposals for zero fees in LN, the current LN still utilizes transaction fee as an important incentive for users to participate as intermediate routers [6].

Balance privacy. Since the fee is related to the balance distribution of a channel, an attacker may violate the balance privacy by observing real-time fee changes. We design a privacy-preserving version of our algorithm to protect the balance changes on channels. Specifically, we allow channel owners to announce updated fees every k payments instead of after every single payment. By aggregating the effects of k payments on the channel balance, an attacker cannot infer the exact channel balance difference corresponding to each on-going transaction. We refer to this scheme as the k-private fee setting scheme, which is an extension of the well-known k-anonymity privacy guarantee [21], [22], [26], [50] to PCN balance privacy. We evaluate the performance of this privacy-preserving version of our algorithm in Sec. VII in addition to our original algorithm. We note that this is likely not the optimal nor the only method for providing privacy preservation, but just serves as an example of the compatibility of our algorithm with more complex privacy preservation techniques. More complex technique can be based on differential privacy [51] by applying a small random perturbation to the fee functions, for instance. Other types of privacy such as sender-recipient privacy can be realized by existing techniques [46] and hence are omitted in this paper.

VII. Performance Evaluation

A. Experiment Settings

We extended the OMNET++ simulator in [49] for evaluation. The implementation used the C++ programming language. The CPU of the experimental machine was Intel(R) Xeon(R) Gold 5317 CPU @ 3.00GHz with 64-bit operating system and 256GB running memory. Our implementation is available as open source on GitHub [4].

1) Topology: We extracted a core network with 128 nodes and 897 edges from a real LN topology snapshot on Oct. 5, 2020 [49]. We kept the largest connected component consisting of the top 0.4% channels in terms of capacity with both nodes having degrees larger than 2. Each channel took 30ms (one time slot) to process and forward a payment. We converted the channel capacities from Satoshis to € to match the payment dataset described below. After prepossessing, the minimum and mean capacities were €4776 and €6285, respectively.

2) Payment workloads: The payment amounts were randomly chosen from a preprocessed real-world credit card dataset [3] with mean €7.43, median €5.97, and maximum €22. Payments arrived following a Poisson distribution. Due to privacy concerns, there is no realistic dataset on transactions happening on the LN or other PCNs. To realistically reflect real-world transaction scenarios, we generated a workload as follows. The workload consisted of two types of sender-recipient pairs in the network: those who frequently transacting with each other (the frequently trading source-destination pairs, such as cryptocurrency exchanges or large companies) and those who infrequently made transactions (such as normal users) and were truly randomly selected as source-destination pairs. In particular, the frequently transacting pairs were randomly selected during initialization and remained fixed throughout the workload. Transactions between frequently transacting nodes were periodically generated to simulate exchanges or companies transacting with each other in regular cycles. Each frequently transacting pair swapped their roles (sender or receiver) every $25,000$ transactions. We designed this workload to reflect real-world transactions, such as regular transactions between cryptocurrency exchanges or financial institutions, which could have a significantly higher volume than between individual users. To model such a workload, we applied the Pareto principle [47] by designating 20% of nodes as frequently transacting pairs, responsible for 80% of total transactions in the network. The rest 20% transactions were between randomly selected normal user pairs.

3) Privacy preservation and selfishness: As discussed in Sec. VI, we also evaluated the performance of Fence with k-private fee setting. Each router only announced its latest fee per k transactions, and we let the routers each have asynchronous announcement cycles (by having each router start from a random initial transaction counter towards k). We denote this privacy-preserving version of our algorithm as **Fence**$^+$. In addition, considering that some nodes may not follow our fee setting scheme and may selfishly set low fees to attract users in an attempt to gain higher income, we introduced two parameters: selfish node ratio and selfish fee ratio. The selfish node ratio indicates how many routers in the network are selfish. Following the assumption of the blockchain, we assume that the majority of nodes in PCN are honest. Therefore, we set the selfish node ratio to be less than or equal to 0.5. The selfish fee ratio is the factor by which these selfish routers set their fees, compared to the original fee that they should have set by using our fee setting scheme.

4) Comparison algorithms: We evaluated **Fence** and **Fence**$^+$ by comparing to seven different routing algorithms:

- **One Shortest Path**: This is the default routing algorithm in LN [5], which finds the shortest path by hop count using the Dijkstra's algorithm to send a payment.
- **K Shortest Path**: This baseline underlies many state-of-the-art PCN routing algorithms, such as Eclair [1] and Flash [54], which use Yen's algorithm [58] to find K paths with fewest hops between sender and recipient, and randomly chooses one of them for a payment.
- **Landmark Routing**: As used in several state-of-the-art PCN routing schemes [39], [46], landmark routing chooses K maximum-degree nodes as landmarks. It then routes each payment via the minimum hop count path through one of the landmarks.
- **Spider Routing** [49]: Spider is one of the state-of-the-art routing schemes that employs congestion control to regulate the payment rate along K *edge-disjoint widest paths*. To avoid the long waiting time and high overhead for payment slicing and queueing, we chose a non-slicing, queue-less version of Spider for fair comparison.
- **Merchant** [52]: Merchant is a recently proposed balance-aware fee setting scheme, whose high-level idea is similar to ours, but with a heuristic linear transaction fee function and without theoretical analysis. For a given balance point of a channel, if an incoming payment brings the channel closer to the balance point, then no fee is charged. If an incoming payment pushes the channel further away from the balance point, the fee on this channel is proportional to the distance that the payment pushes from the balance point, where the distance is measured as the absolute difference between the current channel balance and the balance point, divided by the channel capacity and then multiplied by a constant coefficient. The payments are sent along the path with minimum fee.
- **OptimizedFees** [19]: OptimizedFees is a fee policy that applies a fixed fee plus a variable part which depends on the size of the payment and the imbalance status of the channel. The variable part has two slopes: the low slope s_{low} is applied to payments that decrease the imbalance of the channel, and the high slope s_{high} is applied to payments that increase the imbalance of the channel. The imbalance is measured as the absolute difference between the balances on both sides of the channel.
- **FixedExpFee** [45]: FixedExpFee is a fee policy that depends on the unidirectional balance of a payment channel. It introduces a fixed tunable parameter a as the exponent of the balance in the fee function to reflect the balance status of the channel, where the reciprocal of the balance raised to the power of a is used as the coefficient of the proportional fee.

5) Simulator parameters: According to Sec. VI, we set $\beta_e = 0.5$, $\alpha_e = 0.1$, $n = 10$, $\mathbb{T} = 20$ (time slots), $F_1 = 1$, $F_2 = 1$. To realistically set the fees and valuations, we obtained the fee policy of all channels in the preprocessed LN topology [49], and used the median 0.00015 as our $\mathbb{C}$. This ensures that the fee policy in our simulation does not deviate significantly from the current transaction fees in the LN, and our fees are significantly lower than on-chain or traditional payment methods. Then we set $\mathbb{Z} = \mathbb{C}/(n\mathbb{T}) = 7.5 \times 10^{-7}$ to match the valuation and the usage of capacity according to Assumption 1 in Sec. V-A. For each channel, we assumed that the initial balance was $b(0, e) = 0.5c_e$. For k-private **Fence**$^+$, we set k to 100. Both the selfish node ratio and the selfish fee ratio were set to 0.1 by default. For algorithms involving finding K paths, we set $K = 4$. For **Merchant**, the balance point was set to 0.5 and the constant coefficient was set to 1. For **OptimizedFees**, the fixed fee was 1 Satoshi, $s_{low} = 0.01$, and $s_{low} = 0.03$. For **FixedExpFee**, the tunable parameter a was set to 0.5. Each simulation ran for $2,000,000$ payments, at a default arrival rate of 100 payments per second. We ran each setting for 5 times with different seeds to average-out random noise.

We used the following performance metrics for evaluation. **Payment success ratio** denotes the number of successful payments over the total number of payments. **Throughput success ratio** denotes the successful amount over total amount arrived in the network. **Network imbalance** is defined as the sum of normalized balance difference on two sides of each channel to measure the level of imbalance in the network. **Income ratio** is defined as the ratio between the income received by a node when engaging in selfish behavior and the income when behaving honestly, which measures the impact of the selfish behavior on routers' income.

B. Evaluation Results

Fig. 5 shows the payment and throughput success ratios with different arrival rates under scenarios without and with selfish nodes. Specifically, because the misbehavior of selfish nodes only affects schemes based on fee settings, for the sake of clarity and readability, we only presented the results of Fence, Fence$^+$, Merchant, FixedExpFee, and OptimizedFees in the

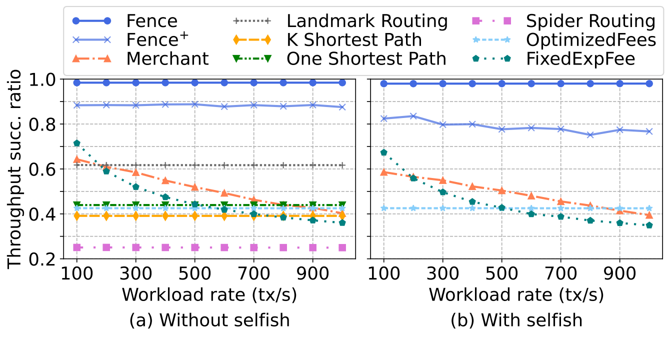


Fig. 5. Throughput success ratios vs. workload arrival rates.

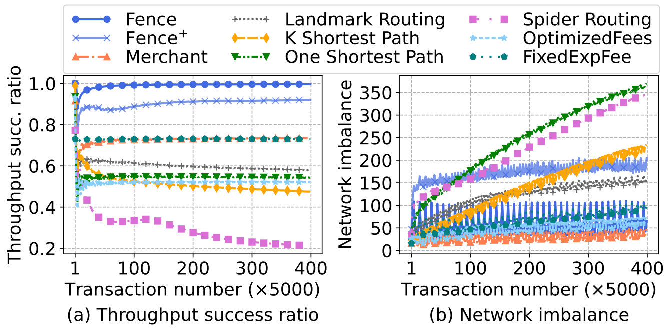


Fig. 6. Throughput success ratio and network imbalance over time.

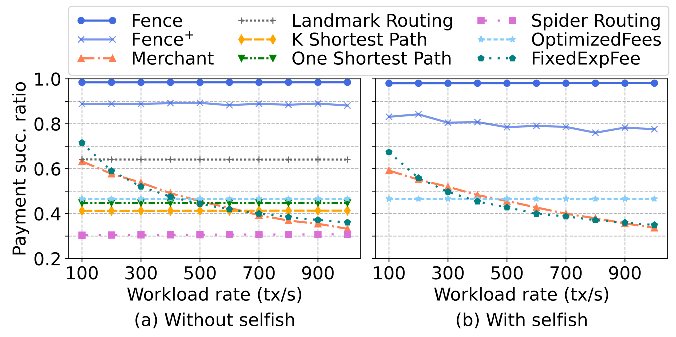


Fig. 7. Payment success ratios vs. workload arrival rates.

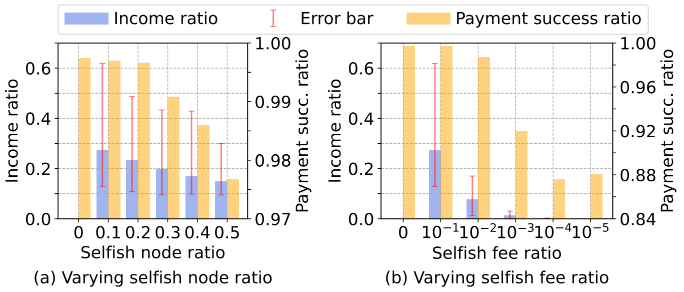


Fig. 8. Payment success ratio and income ratio under different selfish node ratio and selfish fee ratio.

scenario with selfish nodes, where 10% of the nodes intentionally set their fees to one-tenth of the original value to attract more users and attempt to increase their own income, as shown in Fig. 5(b). As shown in Fig. 5, Fence and Fence$^+$ achieved better throughput either with or without selfish behaviors. Due to the privacy preservation consideration, Fence$^+$ performed less effectively than Fence. This indicates a trade-off between the utility and security of the proposed fee setting scheme. The success ratios of FixedExpFee and Merchant decreased with higher arrival rates because these two mechanisms were more affected by changes in channel fees, leading to insufficient fees for forwarding. Comparing Fig. 5(a) with Fig. 5(b), except for OptimizedFees, all fee-based schemes experienced a decrease in the throughput success ratio when there were misbehaving nodes. The success ratio of OptimizedFees remained unchanged compared to the scenario without selfishness. This is because even without selfish behaviors, OptimizedFees only had limited ability to maintain low network imbalance and improve payment throughput, as shown in Fig. 6. As a result, the selfish behaviors had limited impact on its performance. Fence$^+$ was more sensitive to selfishness compared to Fence. This is because Fence$^+$ takes into consideration the privacy and performs periodic, non-real-time announcements of fees. The misbehavior of certain nodes further hampered the ability of fees to accurately reflect the current state of the channels, leading to a decrease in the throughput success ratio. Payment success ratio results were similar to those of throughput success ratio, as shown in Fig. 7.

Fig. 6 shows the changes in throughput success ratio and network imbalance over time in one simulation run. The fee-based schemes (Fence, Fence$^+$, Merchant, OptimizedFees, FixedExpFee) had increasing throughput success ratios over time, while other algorithms (except Spider Routing) generally had decreasing throughput success ratios. This is because in the balance-aware fee-based schemes, payments that might cause channel imbalance could be rejected in the early stage. Other algorithms blindly accepted payments whenever there was available balance, leading to high success ratios at the beginning. Over time, the fee based schemes ensured most channels were relatively balanced, leading to increased throughput success ratios. The other algorithms suffered from high imbalance on critical channels, which led to degrading success ratios over time. Overall, *Fence and Fence$^+$ kept a low and stable level of imbalance during the experiment.* Fence$^+$ was less effective than Fence in maintaining network balance due to the asynchronous fee update frequency, leading to imbalances in some channels. However, the overall network imbalance remained relatively low and stable, and the throughput success ratio was higher than others. OptimizedFees, FixedExpFee and Merchant kept a comparable or slightly lower level of network imbalance as Fence, but with a lower throughput success ratio. This suggested that these algorithms can lead to some channels being very unbalanced while other channels idle or underutilized. One Shortest Path and Spider Routing had both low throughput success ratio and high network imbalance. For Spider Routing, congestion control without slicing led to too many payments being rejected, and hence a low success ratio was observed.

Fig. 8 illustrates how the selfish node ratio and selfish fee ratio affect the payment success ratio and the average income ratio among selfish nodes. In Fig. 8(a), we can observe that as the selfish node ratio increased, the payment success ratio showed a slightly decreasing. This is because the presence of selfish nodes led to channels with low fees being quickly depleted, preventing more payments from being successful. Nevertheless, Fig. 8(a) also shows that as the selfish node ratio increased, the selfish nodes' income ratio actually decreased. This means that selfish nodes received lower average income compared to the income when they honestly. In Fig. 8(b), we

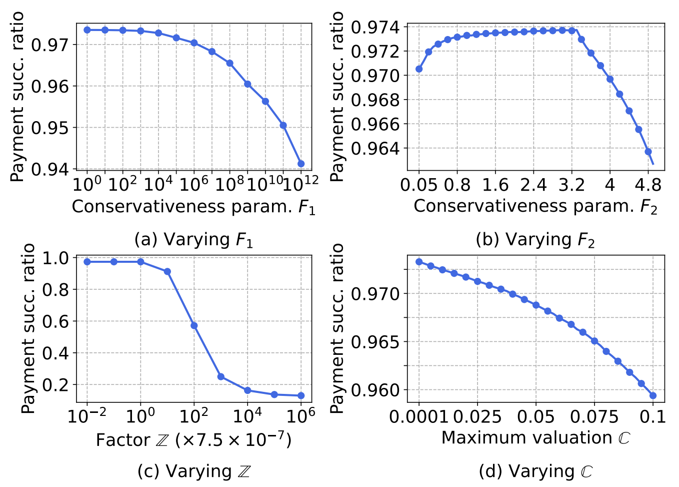


Fig. 9. Payment success ratio under different F_1, F_2, $\mathbb{C}$ and $\mathbb{Z}$.

can observe that some nodes intentionally setting very low fees led to decrease in payment success ratio, but the selfish behavior of these nodes did not bring them more income as the average income ratio of the selfish nodes decreased. Hence we conclude that in real-world scenarios where routers have incomplete knowledge of the behaviors of other nodes, they are incentivized to follow our fee-setting algorithm to achieve a high income on expectation.

Fig. 9 shows how changes in the conservativeness parameters F_1 and F_2, the valuation upper bound $\mathbb{C}$ and the global factor $\mathbb{Z}$ (for matching valuation and capacity usage) affect the payment success ratio of Fence. In Fig. 9(a) and (b), we can observe a similar trade-off for F_1 and F_2. In Fig. 9(a), a workload rate of 1000 was used to illustrate changes in the congestion conservative factor F_1 in a more congested network. A higher F_1 (F_2) meant the routers were more conservative on keeping the channel less congested (more balance) by rejecting payments, which lowered the long-term throughput of the PCN. In Figs. 9(c) and (d), higher valuation led to more violations of the balance constraints, lowering the overall success ratio of Fence. As $\mathbb{Z}$ exceeded 7.5×10^{-7} and increased at a rate of 10 times, the success ratio decreased significantly. This is because when $\mathbb{Z}$ exceeded the point of $\mathbb{C}/(n\mathbb{T})$, the network experienced "inflation" where the same valuation can utilize fewer capacity resources than before. This caused the network to become overly conservative and resulted in a large number of payments being rejected.

VIII. Conclusion

This paper focused on using dynamic balance-aware transaction fees to influence the path selection of users and in turn achieve network balance and improve throughput in a PCN. We proposed an exponential fee setting function for incentivizing users to utilize payment paths with less congestion and more balance. Our algorithm design was backed by rigorous theoretical analysis. We proved that no online algorithm can achieve a finite competitive ratio in a general PCN, and that our algorithm achieves an asymptotically tight competitive ratio in a unidirectional PCN. We then discussed how our algorithm can be turned into a fully distributed protocol with proper parameter setting, fee update and privacy preservation. Extensive simulations showed that our algorithm can keep a PCN balanced and achieve a high throughput, compared to state-of-the-art PCN routing algorithms.

References

[1] "ACINQ/eclair: A Scala Implementation of the Lightning Network," accessed 2023-01-19. [Online]. Available: https://github.com/ACINQ/eclair

[2] "Atomic Multi-path Payments (AMP)," accessed 2023-01-19. [Online]. Available: https://docs.lightning.engineering/lightning-network-tools/lnd/amp

[3] "Credit Card Fraud Detection," accessed 2023-01-19. [Online]. Available: https://www.kaggle.com/mlg-ulb/creditcardfraud

[4] "Implementation of Fence." accessed 2023-05-30. [Online]. Available: https://github.com/xiaojian-wang/Fence

[5] "Lightning Network In-Progress Specifications," accessed 2023-01-19. [Online]. Available: https://github.com/lightning/bolts

[6] "Real-Time Lightning Network Statistics," accessed 2023-01-19. [Online]. Available: https://1ml.com/statistics

[7] "The Visa System: Rates, Fees and Rules," accessed 2023-01-19. [Online]. Available: https://usa.visa.com/support/small-business/regulations-fees.html

[8] "Tor Project," accessed 2023-01-19. [Online]. Available: https://www.torproject.org/

[9] "Raiden Network." accessed 2023-05-10. [Online]. Available: https://raiden-network.readthedocs.io/en/stable/onboarding.html#the-raiden-protocol

[10] K. Asgari, A. A. Mohammadian, and M. Tefagh, "DyFEn: Agent-Based Fee Setting in Payment Channel Networks," *arXiv preprint arXiv:2210.08197*, 2022.

[11] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized Channels From Limited Blockchain Scripts and Adaptor Signatures," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2021, pp. 635–664.

[12] B. Awerbuch, Y. Azar, and S. Plotkin, "Throughput-Competitive On-Line Routing," in *IEEE FOCS*, 1993, pp. 32–40.

[13] 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.

[14] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge University Press, 2005.

[15] W. Chen, X. Qiu, Z. Hong, Z. Zheng, H.-N. Dai, and J. Zhang, "Proactive Look-ahead Control of Transaction Flows for High-throughput Payment Channel Network," in *ACM SoCC*, 2022, pp. 429–444.

[16] Y. Chen, Y. Ran, J. Zhou, J. Zhang, and X. Gong, "MPCN-RP: A Routing Protocol for Blockchain-based Multi-Charge Payment Channel Networks," *IEEE Transactions on Network and Service Management*, 2021.

[17] G. v. Dam, R. A. Kadir, P. N. Nohuddin, and H. B. Zaman, "Improvements of the Balance Discovery Attack on Lightning Network Payment Channels," in *IFIP SEC*, 2020, pp. 313–323.

[18] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards Scaling Blockchain Systems via Sharding," in *ACM SIGMOD*, 2019, pp. 123–140.

[19] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre, "Routing Payments on the Lightning Network," in *IEEE IThings and GreenCom and CPSCom and SmartData*, 2018, pp. 1161–1170.

[20] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, and K. Hostáková, "Multi-party Virtual State Channels," in *Springer EUROCRYPT*, 2019, pp. 625–656.

[21] K. El Emam and F. K. Dankar, "Protecting privacy using k-anonymity," *Journal of the American Medical Informatics Association*, vol. 15, no. 5, pp. 627–637, 2008.

[22] K. El Emam, F. K. Dankar, R. Issa, E. Jonker, D. Amyot, E. Cogo, J.-P. Corriveau, M. Walker, S. Chowdhury, R. Vaillancourt *et al.*, "A globally optimal k-anonymity method for the de-identification of health data," *Journal of the American Medical Informatics Association*, vol. 16, no. 5, pp. 670–682, 2009.

[23] 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.

[24] O. Ersoy, P. Moreno-Sanchez, and S. Roos, "Get Me Out of This Payment! Bailout: An HTLC Re-routing Protocol," *Cryptology ePrint Archive*, 2022.

[25] Z. Ge, Y. Zhang, Y. Long, and D. Gu, "Shaduf: Non-Cycle Payment Channel Rebalancing," in *ISOC NDSS*, 2022.

[26] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, 2007.

[27] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An Untrusted Bitcoin-compatible Anonymous Payment Hub," in *ISOC NDSS*, 2017.

[28] Z. Hong, S. Guo, P. Li, and W. Chen, "Pyramid: A Layered Sharding Blockchain System," in *IEEE INFOCOM*, 2021, pp. 1–10.

[29] Z. Hong, S. Guo, R. Zhang, P. Li, Y. Zhan, and W. Chen, "Cycle: Sustainable Off-chain Payment Channel Network with Asynchronous Rebalancing," in *IEEE/IFIP DSN*, 2022, pp. 41–53.

[30] P. Jaillet and M. R. Wagner, "Generalized Online Routing: New Competitive Ratios, Resource Augmentation, and Asymptotic Analyses," *Operations research*, vol. 56, no. 3, pp. 745–757, 2008.

[31] P. Joseph and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," 2016, accessed 2022-08-01. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[32] R. Khalil and A. Gervais, "Revive: Rebalancing Off-blockchain Payment Networks," in *ACM CCS*, 2017, pp. 439–453.

[33] S. Lin, J. Zhang, and W. Wu, "FSTR: Funds Skewness Aware Transaction Routing for Payment Channel Networks," in *IEEE/IFIP DSN*, 2020, pp. 464–475.

[34] Y. Liu, Y. Wu, F. Zhao, and Y. Ren, "Balanced off-chain payment channel network routing strategy based on weight calculation," *The Computer Journal*, p. bxad029, 2023.

[35] X. Luo and P. Li, "Learning-Based Off-Chain Transaction Scheduling in Prioritized Payment Channel Networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3589–3599, 2022.

[36] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards Bitcoin Payment Networks," in *Springer ACISP*, 2016.

[37] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online Task Dispatching and Scheduling with Bandwidth Constraint in Edge Computing," in *IEEE INFOCOM*, 2019, pp. 2287–2295.

[38] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and State Channels: Payment Networks That Go Faster Than Lightning," in *Springer FC*, 2019, pp. 508–526.

[39] P. Moreno-Sanchez, A. Kate, and M. Maffei, "Silentwhispers: Enforcing Security and Privacy in Decentralized Credit Networks," in *ISOC NDSS*, 2017.

[40] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008, accessed 2022-08-01. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[41] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, "Fast Local Rerouting for Handling Transient Link Failures," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 359–372, 2007.

[42] N. Papadis and L. Tassiulas, "Payment Channel Networks: Single-Hop Scheduling for Throughput Maximization," in *IEEE INFOCOM*, 2022, pp. 900–909.

[43] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An Approach to Routing in Lightning Network," *White Paper*, p. 144, 2016.

[44] C. Qian and S. S. Lam, "Greedy Distance Vector Routing," in *IEEE ICDCS*, 2011, pp. 857–868.

[45] A. H. J. Ren, L. Feng, S. A. Cheong, and R. S. M. Goh, "Optimal fee structure for efficient lightning networks," in *IEEE 24th ICPADS*, 2018, pp. 980–985.

[46] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling Payments Fast and Private: Efficient Decentralized Routing for Path-based Transactions," in *ISOC NDSS*, 2018.

[47] R. Sanders, "The pareto principle: Its use and abuse," *Journal of Services Marketing*, vol. 1, no. 2, pp. 37–40, 1987.

[48] H. Shi and L. Chen, "From Spectrum Bonding to Contiguous-resource Batching Task Scheduling," *IEEE/ACM Transactions on Networking*, 2022.

[49] 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 *USENIX NSDI*, 2020, pp. 777–796.

[50] L. Sweeney, "k-anonymity: A model for protecting privacy," *International journal of uncertainty, fuzziness and knowledge-based systems*, vol. 10, no. 05, pp. 557–570, 2002.

[51] G. van Dam and R. A. Kadir, "Hiding Payments in Lightning Network with Approximate Differentially Private Payment Channels," *Computers & Security*, vol. 115, p. 102623, 2022.

[52] Y. Van Engelshoven and S. Roos, "The Merchant: Avoiding Payment Channel Depletion Through Incentives," in *IEEE DAPPS*, 2021, pp. 59–68.

[53] G. I. Vineet Goyal and R. Udwani, "Asymptotically Optimal Competitive Ratio for Online Allocation of Reusable Resources," in *Springer WINE*, 2021, p. 543.

[54] P. Wang, H. Xu, X. Jin, and T. Wang, "Flash: Efficient Dynamic Routing for Offchain Networks," in *ACM CoNEXT*, 2019, pp. 370–381.

[55] Q. Wang, Y. Zhang, Z. Bao, W. Shi, H. Lei, H. Liu, and B. Chen, "SorTEE: Service-Oriented Routing for Payment Channel Networks with Scalability and Privacy Protection," *IEEE Transactions on Network and Service Management*, 2022.

[56] G. Wood *et al.*, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[57] H. Xue, Q. Huang, and Y. Bao, "EPA-Route: Routing Payment Channel Network with High Success Rate and Low Payment Fees," in *IEEE ICDCS*, 2021, pp. 227–237.

[58] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[59] 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 *IEEE ICCCN*, 2018, pp. 1–9.

[60] J. Zhang, Y. Ye, W. Wu, and X. Luo, "Boros: Secure and Efficient Off-Blockchain Transactions via Payment Channel Hub," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[61] X. Zhang and C. Qian, "Towards Aggregated Payment Channel Networks," in *IEEE ICNP*, 2022, pp. 1–11.

[62] X. Zhang, S. Shi, and C. Qian, "WebFlow: Scalable and Decentralized Routing for Payment Channel Networks with High Resource Utilization," *arXiv preprint arXiv:2109.11665*, 2021.

[63] Y. Zhang and D. Yang, "RobustPay+: Robust Payment Routing with Approximation Guarantee in Blockchain-based Payment Channel Networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1676–1686, 2021.

IX. Appendix

This section shows the proof of Lemmas 2−5 and Theorems 1−3. Notations used in this section are listed in Table III.

Proof of Lemma 2

Proof. We prove by contradiction. Assume a channel e has balance constraint violated for the first time after payment R_i is sent by the sender. This means that the balance utilization $\lambda_{2,e}(i+1) > 1$. According to the balance utilization update policy in Algorithm 1, we have $\lambda_{2,e}(i) > 1 - \delta_i/((\beta_e - \alpha_e)c_e)$.

According to Assumption 2, we have

$$\lambda_{2,e}(i) > 1 - 1/\log_2 \mu_2. \tag{17}$$

Inserting Eq. (17) into Eq. (12), we can get an inequality about imbalance cost

$$\sigma_{2,e}(T,i) > \mathbb{Z}(\beta_e - \alpha_e)c_e(\mu_2^{1-\frac{1}{\log_2 \mu_2}} - 1) = \mathbb{Z}(\beta_e - \alpha_e)c_e n F_2.$$

Similarly, if the capacity constraint at T is violated, we can get an inequality about congestion cost

$$\sigma_{1,e}(T,i) > \mathbb{Z}c_e n \mathbb{T} F_1.$$

Let $\sigma_{1,e}^{\mathsf{unit}}(T,i) = \frac{\sigma_{1,e}(T,i)}{\mathbb{Z}c_e}$ and $\sigma_{2,e}^{\mathsf{unit}}(i) = \frac{\sigma_{2,e}(i)}{\mathbb{Z}(\beta_e - \alpha_e)c_e}$. Consider only one time slot and only one channel's cost:

$$\mathbb{Z}\sigma_{1,e}^{\mathsf{unit}}(T,i) + \mathbb{Z}\sigma_{2,e}^{\mathsf{unit}}(i) > \mathbb{Z}n\mathbb{T}F_1 + \mathbb{Z}nF_2.$$

According to Assumption 1, we have:

$$\rho_i = C_i\delta_i \leq \mathbb{Z}n\mathbb{T}F_1\delta_i + \mathbb{Z}nF_2\delta_i < \varphi_i^{\mathsf{lin}}(p_i,\delta_i) \leq \varphi_i^{\mathsf{conv}}(p_i,\delta_i).$$

The last inequality above is because, by definition, for the same payment amount on the same path, the *convolutional fee* is always ***no less than*** the *linear summation fee* given the same fee function $\phi_e(\cdot)$ on any channel e. With either linear or convolutional fees, this contradicts with the condition for the sender to send a payment in Line 2 of Algorithm 1. □

TABLE III
NOTATION TABLE OF APPENDIX

Symbol	Definition
$\mathcal{A}, \mathcal{A}^*$	Accepted payments set of Algorithm 1 and the optimal offline algorithm
k	The index of the last payment in $\mathcal{A}$
$\sigma^{\mathsf{unit}}_{1,e}(T,i), \sigma^{\mathsf{unit}}_{2,e}(i)$	Unit congestion and imbalance cost
Γ^i_e	The total cost of using up all the resources on e
Π_i	The increase in network-wide cost caused by an accepted payment R_i
$\Delta_{1,e}(T), \Delta_{2,e}$	The difference in congestion and imbalance cost between two adjacent payments
$\mathcal{Q}$	The set of payments served by the offline algorithm but not by Algorithm 1
P^*_i	The path set that offline algorithm serves $R_i \in \mathcal{Q}$
E^*_i	The channel set used by the offline algorithm
$\boldsymbol{\delta}^*_i(e)$	The contribution of payment R_i to the channel e
p^{onl}_i	The minimum-fee path for R_i when Algorithm 1 tries to serve R_i
$\mathcal{R}_\tau$	All payments arriving at time $T=\tau$
ξ_τ, η	Weighted throughput obtained by an online algorithm from payments in $\mathcal{R}_\tau$, its cumulative weighted throughput until time τ
S_τ	The amortized weighted throughput of the algorithm until $T=\tau$

Proof of Lemma 3

Proof. Following definition of the valuation in payment model III-B, the total transaction fee of a forwarded payment R_i is bounded by $\mathbb{C}\cdot\delta_i$. This means that the transaction amount on a channel e, which is the payment amount plus the transaction fee incurred on e and any channel after e along the path, is bounded by $(1+\mathbb{C})\delta_i$, *i.e.*, $f_{i,e}(p_i) \le (1+\mathbb{C})\delta_i, \forall i \in \mathcal{A}$.

For constraint (5b), we have $\sum_{i\in\mathcal{A}} \kappa(T,i)\delta_i \le c_e$ according to Lemma 2. Therefore, we have

$$\sum\nolimits_{i\in\mathcal{A}} \kappa(T,i) f_{i,e}(p_i) \le \sum\nolimits_{i\in\mathcal{A}} \kappa(T,i)(1+\mathbb{C})\delta_i \le c_e(1+\mathbb{C}).$$

Constraint (5c) follows a similar proof. ☐

Proof of Lemma 4

Proof. We prove by induction on k. Before the first transaction comes ($k=0$), since the unidirectional channel's initial balance is equal to its capacity, we have $b(0,e)=c_e$, and hence Eq. (16) is true.

Let $\Pi_i = \sum_{T\in\mathcal{T}}\sum_{e\in p}(\sigma_{1,e}(T,i+1)-\sigma_{1,e}(T,i)) + \sum_{e\in p}(\sigma_{2,e}(i+1)-\sigma_{2,e}(i))$ be the increase in network-wide cost caused by an accepted payment R_i. To prove Eq. (16), we first show that $\Pi_i \le 2\rho_i(\log_2\mu_1 + \log_2\mu_2), \forall R_i \in \mathcal{A}$.

For ease of expression, we define $\Delta_{1,e}(T) \triangleq \mathbb{Z} c_e \mu_1^{\lambda_{1,e}(T,i)}\left(2^{\frac{\delta_i}{c_e}\log_2\mu_1}-1\right)$. Based on Eq. (11), we have $\sigma_{1,e}(T,i+1)-\sigma_{1,e}(T,i) = \Delta_{1,e}(T)$ which is the difference in congestion cost between two adjacent payments.

Similarly, $\Delta_{2,e} \triangleq \mathbb{Z}(\beta_e-\alpha_e)c_e\mu_2^{\lambda_{2,e}(i)}\left(2^{\frac{\delta_i\log_2\mu_2}{(\beta_e-\alpha_e)c_e}}-1\right)$. Given that the imbalance cost is defined with two conditions, we must consider the various cases that may arise when adjacent payments are in different segments. Based on Eq. (12), when $\lambda_{2,e}(i), \lambda_{2,e}(i+1) \ge 0$, we have $\sigma_{2,e}(i+1)-\sigma_{2,e}(i) = \Delta_{2,e}$; when $\lambda_{2,e}(i)<0$, we have $\sigma_{2,e}(i+1)-\sigma_{2,e}(i) < \Delta_{2,e}$. Therefore, we always have

$$\sigma_{1,e}(T,i+1)-\sigma_{1,e}(T,i) = \Delta_{1,e}(T), \text{ and} \\ \sigma_{2,e}(i+1)-\sigma_{2,e}(i) \le \Delta_{2,e} \tag{18}$$

According to Assumption 2, the fact that $2^{\mathbf{x}}-1 \le \mathbf{x}$ for $0 \le \mathbf{x} \le 1$, and the definition of $\sigma_{1,e}(T,i)$, we have

$$\sum_{T\in\mathcal{T}}\sum_{e\in p_i}\Delta_{1,e}(T) \le \log_2\mu_1 \sum_{e\in p_i}\sum_{T\in\mathcal{T}} \mathbb{Z}\delta_i\left(\sigma^{\mathsf{unit}}_{1,e}(T,i)+1\right).$$

Based on Line 2 in Algorithm 1 and Assumption 1, under *linear summation fee*, we have $\mathbb{Z}\sum_{e\in p_i}\sum_{T\in\mathcal{T}}\sigma^{\mathsf{unit}}_{1,e}(T,i)\delta_i \le \varphi^{\mathsf{lin}}_i(p_i,\delta_i) \le \rho_i$ and $\mathbb{Z}\sum_{T\in\mathcal{T}}\sum_{e\in p}\delta_i \le \mathbb{Z}n\mathbb{T}\delta_i \le C_i\delta_i = \rho_i$.

$$\sum\nolimits_{T\in\mathcal{T}}\sum\nolimits_{e\in p_i}\Delta_{1,e}(T) \le 2\rho_i\log_2\mu_1. \tag{19}$$

Under *convolutional fee*, Eq. (19) still holds, as the convolutional fee (and ρ_i) is no less than the linear summation fee.

Following the same derivation as above, we have

$$\sum\nolimits_{e\in p_i}\Delta_{2,e} \le 2\rho_i\log_2\mu_2. \tag{20}$$

Combining Eqs. (18)–(20), Eq. (16) holds. ☐

Proof of Lemma 5

Proof. Suppose the offline algorithm uses path set P^*_i to serve payment $R_i \in \mathcal{Q}$. The contribution of this payment to the channel $e \in E^*_i$ is $\boldsymbol{\delta}^*_i(e) = \sum_{p\in P^*_i : e\in p}\boldsymbol{\delta}^*_i(p)$, where E^*_i is the channel set used by the offline algorithm. We note that for the payments that are served by offline algorithm, their contribution to the channel is also constrained by (5b) and (5c).

Suppose p^{onl}_i is the minimum-fee path for R_i when Algorithm 1 tries to serve R_i. Since this payment is not served by the online algorithm, we know $\varphi^{\mathsf{lin}}_i(p^{\mathsf{onl}}_i,\delta_i) > \rho_i$ according to Line 2 of Algorithm 1, and thus

$$\sum_{i\in\mathcal{Q}}\rho_i < \sum_{i\in\mathcal{Q}}\sum_{e\in p^{\mathsf{onl}}_i}\frac{\delta_i\Gamma^i_e}{(\beta_e-\alpha_e)c_e} \le \sum_{i\in\mathcal{Q}}\sum_{p\in P^*_i}\sum_{e\in p}\frac{\boldsymbol{\delta}^*_i(p)\Gamma^i_e}{(\beta_e-\alpha_e)c_e}$$
$$\le \sum_{e\in E^*_i}\Gamma^{k+1}_e\sum_{i\in\mathcal{Q}}\frac{\boldsymbol{\delta}^*_i(e)}{(\beta_e-\alpha_e)c_e} \le \sum_{e\in E}\Gamma^{k+1}_e.$$

The first inequality is due to definition of the fee functions. The second inequality is because p^{onl}_i is the minimum-fee path for R_i. The third inequality is because the costs monotonically increase in a unidirectional PCN. The fourth inequality is due to the offline balance utilization being bounded by 1. ☐

Proof of Theorem 1

Proof. The optimal offline profit is less than or equal to $\sum_{i\in\mathcal{Q}}\rho_i + \sum_{i\in\mathcal{A}}\rho_i$. Based on Lemmas 4 and 5, we have

$$\sum\nolimits_{i\in\mathcal{Q}}\rho_i + \sum\nolimits_{i\in\mathcal{A}}\rho_i \le (2\log_2\mu_1\mu_2+1)\sum\nolimits_{i\in\mathcal{A}}\rho_i.$$

So the competitive ratio of Algorithm 1 is $a = (2\log_2\mu_1\mu_2 + 1)$. Based on Lemma 3, Algorithm 1 is $(O(\log n\mathbb{T}), 1+\mathbb{C})$-competitive in this unidirectional graph special case. ☐

Proof of Theorem 2

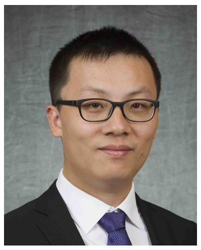
Proof. We construct an example graph on which any online algorithm has competitive ratio of $\Omega(\log n)$. Consider a line graph with nodes $V=\{v_1,\ldots,v_{n+1}\}$ and channels $E=\{(v_1,v_2),(v_2,v_3),\ldots,(v_n,v_{n+1})\}$, where $b(0,e)=\beta_e c_e = 1$ and $\alpha_e = 0$ for all channels. Assume n is a power of 2. Assume each payment takes unit time to complete, and has equal payment amount of $1/\log_2 n$ (asymptotically satisfying Assumption 2) and unit valuation ($C_i=1$). At time $T=0$, there are $\log_2 n$ payments arriving, each from v_1 to v_n. At $T=1$, there are two groups each consisting of $\log_2 n$ payments arriving; the first group is from v_1 to $v_{n/2}$, and the second group is from $v_{n/2+1}$ to v_n. At $T=\tau$, there are 2^τ groups each with $\log_2 n$ payments arriving, and all group-j payments are from node $v_{jn/2^\tau}$ to $v_{(j+1)n/2^\tau}$. All groups of payments arriving at each time T saturate all channels, while the number of payments arriving at time T is twice of that at time $T-1$. Let $\mathcal{R}_\tau$ be all payments arriving at time $T=\tau$.

Let ξ_τ be weighted throughput obtained by an online algorithm from payments in $\mathcal{R}_\tau$, and let $\eta = \sum_{\kappa\leq\tau}\xi_\kappa$ be its cumulative weighted throughput until time $T=\tau$. At time $T=\tau$, it takes $2^{-\tau}n$ balance to achieve unit weighted throughput. The total balance budget of the network is n. Hence the cumulative balance up to time $T=\log_2 n$ satisfies that $\sum_{\tau=0}^{\log_2 n}2^{-\tau}n\xi_\tau\leq n$. Define $S_\tau=\eta/2^\tau$ as the amortized weighted throughput of the algorithm until $T=\tau$. We have $\sum_{\tau=0}^{\log_2 n}S_\tau=\sum_{\tau=0}^{\log_2 n}2^{-\tau}\sum_{\kappa=0}^{\tau}\xi_\kappa\leq\sum_{\tau=0}^{\log_2 n}2\cdot 2^{-\tau}\xi_\tau\leq 2$. Taking the average, we have at least one $\tau\leq\log_2 n$ satisfying $S_\tau\leq 2/\log_2 n$. The online algorithm accumulates at most $\sum_{\kappa=0}^{\tau}\xi_\kappa=S_\tau\cdot 2^\tau\leq 2^{\tau+1}/\log_2 n$ weighted throughput until time $T=\tau$. Meanwhile, an offline algorithm can simply accept all payments in $\mathcal{R}_\tau$, reject all others, and obtain 2^τ weighted throughput. □

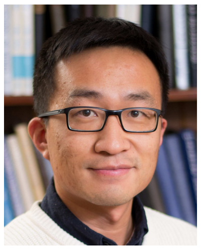
Proof of Theorem 3

Proof. We construct an example graph on which any online algorithm has a infinite competitive ratio. Consider a network with a single bidirectional channel between nodes v_0 and v_1 with a capacity of c. Initially, $b(0,v_0v_1)=0.5c+\epsilon$, and $b(0,v_1v_0)=0.5c-\epsilon$, where $\epsilon>0$ can be arbitrarily small. Without loss of generality, consider $\alpha_e=0$. At $T=0$, a payment from v_0 to v_1 with amount ϵ arrives. If $\mathbf{A}$ rejects it, since this could be the only payment in $\mathcal{R}$ and an offline algorithm can accept it, the competitive ratio is infinity. If $\mathbf{A}$ accepts it, then we assume that at each time $T=1,2,\ldots$, a payment with amount $0.5c+\epsilon$ arrives, in alternating directions between v_0 and v_1, starting from v_0 to v_1 initially. Clearly $\mathbf{A}$ cannot accept any of these payments due to insufficient balance on either side. An offline algorithm, by rejecting the first payment at $T=0$, can accept all subsequent payments, since two consecutive payments simply cancel out each other. If $\mathcal{R}$ contains $\lceil\frac{\mathbf{a}\cdot\epsilon}{0.5c+\epsilon}\rceil+1$ payments as above, the competitive ratio of $\mathbf{A}$ is at least $\boldsymbol{a}$. The theorem follows. □

Xiaojian Wang (Student Member 2021) received her B.E. degree from Taiyuan University of Technology, China, in 2017 and received her M.S. degree in Computer Science from University of West Florida, FL, USA and Taiyuan University of Technology, China, in 2020. She is now a Ph.D. student in the department of Computer Science, College of Engineering at North Carolina State University. Her research interests include payment channel network, security, blockchain, edge computing and so on.

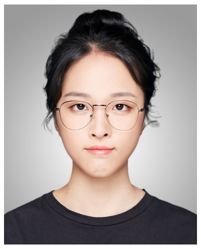


Ruozhou Yu (Student Member 2013, Member 2019, Senior Member 2021) is an Assistant Professor of Computer Science at North Carolina State University, USA. He received his PhD degree (2019) in Computer Science from Arizona State University, USA. His research interests include internet-of-things, cloud/edge computing, smart networking, algorithms and optimization, distributed machine learning, security and privacy, blockchain, and quantum networking. He has served or is serving on the organizing committees of IEEE INFOCOM 2022-2023 and IEEE IPCCC 2020-2023, as a TPC Track Chair for IEEE ICCCN 2023, and as members of the technical committee of IEEE INFOCOM 2020-2024 and ACM Mobihoc 2023. He is a recipient of the NSF CAREER Award in 2021.



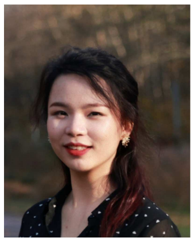
Dejun Yang (Senior Member, IEEE) received the B.S. degree in computer science from Peking University, Beijing, China, and the Ph.D. degree in computer science from Arizona State University, Tempe, AZ, USA.
He is currently an Associate Professor of Computer Science with the Colorado School of Mines, Golden, CO, USA. His research interests include the Internet of Things, networking, and mobile sensing and computing with a focus on the application of game theory, optimization, algorithm design, and machine learning to resource allocation, security, and privacy problems.
Prof. Yang has received the IEEE Communications Society William R. Bennett Prize in 2019. He has served as the TPC Vice-Chair for Information Systems for IEEE International Conference on Computer Communications (INFOCOM). He currently serves an Associate Editor for the IEEE Transactions on Mobile Computing, IEEE Transactions on Network Science and Engineering, and IEEE Internet of Things Journal.

Guoliang Xue (Member 1996, Senior Member 1999, Fellow 2011) is a Professor of Computer Science in the School of Computing and Augmented Intelligence at Arizona State University. His research interests span the areas of Internet-of-things, cloud/edge/quantum computing and networking, crowdsourcing and truth discovery, QoS provisioning and network optimization, security and privacy, optimization and machine learning. He received the IEEE Communications Society William R. Bennett Prize in 2019. He is an Associate Editor of IEEE Transactions on Mobile Computing, as well as a member of the Steering Committee of this journal. He served on the editorial boards of IEEE/ACM Transactions on Networking and IEEE Network Magazine, as well as the Area Editor of IEEE Transactions on Wireless Communications, overseeing 13 editors in the Wireless Networking area. He has served as VP-Conferences of the IEEE Communications Society. He is the Steering Committee Chair of IEEE INFOCOM.



Huayue Gu (Student Member 2021) received her M.S. degree from the University of California, Riverside, CA, USA, in 2021. Currently, she is a Ph.D. student in the Computer Science department at North Carolina State University. Her research interests are quantum networking, quantum communication, data analytics, etc.

Zhouyu Li (Student Member 2021) received his B.S. degree from Central South University, Changsha, China, in 2019 and his M.S. degree from Georgia Institute of Technology, Atlanta, U.S., in 2020. Currently, he is a Ph.D. student of Computer Science at North Carolina State University. His research interests include privacy, cloud/edge computing, network routing, etc.



Fangtong Zhou (Student Member 2021) received her B.E. degree (2018) in Electrical Engineering and Automation from Harbin Institute of Technology, Harbin, China and M.S. degree (2020) in Electrical Engineering from Texas A&M University, College Station, Texas, USA. Currently she is a Ph.D candidate in the School of Computer Science at North Carolina State University. Her research interests include machine learning in computer networking, like federated learning, reinforcement learning for resource provisioning, etc.