Secure Balance Planning of Off-blockchain Payment Channel Networks

Peng Li*, Toshiaki Miyazaki*, and Wanlei Zhou†
*The University of Aizu, Japan. Email: {pengli, miyazaki}@u-aizu.ac.jp
†University of Technology Sydney, Australia. Email: wanlei.zhou@uts.edu.au

***Abstract*—Off-blockchain payment channels can significantly improve blockchain scalability by enabling a large number of micro-payments between two blockchain nodes, without committing every single payment to the blockchain. Multiple payment channels form a payment network, so that two nodes without direct channel connection can still make payments. A critical challenge in payment network construction is to decide how many funds should be deposited into payment channels as initial balances, which seriously influences the performance of payment networks, but has been seldom studied by existing work. In this paper, we address this challenge by designing PnP, a balance planning service for payment networks. Given estimated payment demands among nodes, PnP can decide channel balances to satisfy these demands with a high probability. It does not rely on any trusted third-parties, and can provide strong protection from malicious attacks with low overhead. It obtains these benefits with two novel designs, the cryptographic sortition and the chance-constrained balance planning algorithm. Experimental results on a testbed of 30 nodes show that PnP can enable 30% more payments than other designs.**

***Index Terms*—blockchain, payment network, balance planning, chance constraint**

I. Introduction

With the inception of Bitcoin [1], blockchain has shown great promise in implementing decentralized cryptocurrencies without trusted third parties. Later, Ethereum [2] has been designed to significantly extend the blockchain functions by implementing smart contracts, which are essentially autonomous computer programs executed by all blockchain nodes.

Blockchain uses a serious of sophisticated designs to achieve decentralization and consistency, but sacrifices the scalability because of the consensus protocol involving all nodes. Currently, Bitcoin can handle only 7 transactions per second, and Ethereum increases the performance to about 15 transactions per second. Some recent work [3]–[5] has proposed novel consensus mechanisms that can support thousands of transactions per second, but it is still far from satisfying the performance requirement of large-scale applications. For example, Trillo [6] has reported that Visa handled about 47,000 transactions per second during 2013 holidays, which cannot be supported by any existing blockchains.

Instead of struggling with consensus mechanism design, payment channels [7], [8] have been proposed to address the scalability challenge by creating a private channel between two blockchain nodes. Payment channels can be implemented using smart contracts. Specifically, two nodes create a special channel contract that locks a certain amount of funds as initial balances. After that, they can make transactions over the channel and maintain their channel balances by themselves. Since transactions over the channel are not committed to the blockchain, the scalability of blockchain can be significantly improved.

Multiple payment channels among nodes form a payment network, so that two nodes without direct payment channel can still make off-blockchain transactions in a multi-hop manner. Payment networks have been supported by many blockchain systems. For example, Lightning Network [7] has been designed for Bitcoin, and Ethereum also has its own payment network implementation called Raiden Network [8].

To establish a payment channel, a critical problem is to decide how many funds should be deposited into the channel as initial balances. With fewer funds, the channel needs to be frequently reconstructed, leading to more blockchain accesses. On the other hand, locking funds more than payment demands on the channel is unnecessary. The problem becomes more challenging when we consider a payment network with multi-hop transactions. Although many research efforts [9]–[11] have been made on payment channels and payment networks, they ignore this issue by holding an unrealistic assumption that there are always sufficient funds in payment channels. Khalil et al. [12] have started to pay attention to this challenge by designing REVIVE, which enables nodes to rebalance funds among their payment channels. However, merely rebalancing is not always a good solution because the amount of payments among nodes are usually asymmetric in practice. Furthermore, REVIVE adopts a centralized design that needs a trusted third-party to run the rebalancing algorithm, which would be vulnerable to malicious attacks.

In this paper, we study the balance planning problem of payment networks, i.e., deciding initial balances of payment channels, given the estimated payment demands among nodes. We design PnP, a balance planning service that can be easily integrated into existing payment networks. PnP can exploit the knowledge of payment demands, even with estimation errors, to minimize the total channel deposits. It does not rely on trusted third-parties and can resist attacks from malicious nodes. Moreover, it has low communication and computation overhead. PnP reaps these benefits by addressing two main challenges.

First, it is difficult to accurately estimate the payment demands among nodes. An intuitive idea is to predict future payment demands according to historical transaction records

and accordingly design algorithms to decide initial channel balances. However, the amount of payments that actually happen in practice would be more or less than the prediction. Once the balance of a channel is exhausted, all transactions associated with this channel fail. Of course, we can improve the prediction method to get more accurate estimation of payment demands, unfortunately, which cannot fundamentally solve this problem. In this paper, we deal with the uncertainty of payment demands by expressing the balance planning problem as a chance-constrained optimization problem. We then design an algorithm to solve this problem, which can theoretically guarantee that payment demands can be satisfied with a high probability.

The second challenge stems from the untrustiness of open distributed systems like blockchain, i.e., malicious nodes may exist in the payment network and they attempt to compromise the balance planning service. Many existing work on payment networks assumes the existence of trusted third-parties. However, this assumption is too strong, and sometimes impossible in practice. PnP allows a portion of Byzantine adversaries who can behave arbitrarily and manipulate balance planning results. To protect the system, PnP randomly selects a group of nodes as a decision committee to generate correct balance planning results. Without any trusted third-parties, PnP guarantees safety by addressing two critical issues: who should be selected into the decision committee and how the decision committee achieves agreement on correct planning results. We propose a cryptographic sortition mechanism for committee member elections. Each node in the committee independently runs our proposed balance planning algorithm and then achieves agreement via the Byzantine protocol.

We integrate PnP with the Lightning Network Daemon (lnd) [13], a complete implementation of the Lightning Network [7] nodes, and use it to create a virtual payment network as the testbed for performance evaluation. Since PnP is the first work for balance planning, we compare it with several variants with different design choices. The main contributions of this paper are summarized as follows.

- We design PnP for balance planning of payment networks. To the best of our knowledge, PnP is the first work targeting on solving the balance planning problem.
- To protect PnP service from attacks, we use cryptographic sortition to select a decision committee responsible for running the planning algorithm. It has low communication overhead because of no interaction among nodes.
- We formulate the problem of minimizing the total funds deposited into payment channels, while satisfying the payment demands among nodes. We use chance constraints to describe the uncertainty of payment demands and solve the corresponding problem using approximation techniques.
- We evaluate PnP on a testbed of 30 nodes. Experimental results show that PnP can satisfy payment demands with a high probability while completing at most 30% more payments than other designs.

The rest of this paper is organized as follow. Some preliminaries about payment networks and the motivation of PnP are presented in Section II, followed by the system model in Section III. In Section IV, we present the PnP design for payment networks with unidirectional and bidirectional channels. We present experimental results in Section V. Related work is given in Section VI. Section VII finally concludes this paper.

II. Preliminaries and Motivation

In this section, we present some necessary preliminaries about payment channels and payment networks, followed by some observations that motivate this paper.

A. Payment channels

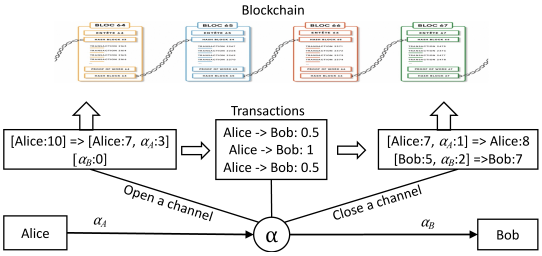
Payment channels can be implemented by smart contracts. When two nodes agree to open a payment channel, they create a smart contract that locks some cryptocurrency coins as channel balances. Once the channel is established, they can make transactions over the channel and maintain channel balances by themselves, without committing these transactions to the blockchain. The payment channel can be unidirectional or bidirectional. An example of unidirectional channels is shown in Fig. 1(a). Alice creates a unidirectional channel, represented by α, to Bob, and their balances on the channel are denoted by α_A and α_B, respectively. Alice deposits 3 coins from her blockchain account (with 10 coins) into this channel as her initial balance, i.e., $\alpha_A = 3$, which is committed to the blockchain. After that, Alice can make payments, without touching the blockchain, to Bob if her channel balance α_A is sufficient. Since α is a unidirectional channel, it does not allow Bob to pay Alice over this channel.

The bidirectional channel allows two-way payments. As shown in Fig. 1(b), both Alice and Bob can deposit some coins into the channel, which are recorded by the blockchain when they open the channel. Then, they can make transactions in any direction and maintain their channel balances accordingly. An interesting observation of the bidirectional channel is that it is possible to let a node pay more than its initial deposit. In the example in Fig. 1(b), Alice pays Bob 3.5 coins in total, but her initial deposit is only 3 coins. That is because the payment from Bob to Alice can charge Alice's balance. Therefore, the bidirectional channel has stronger payment capability.

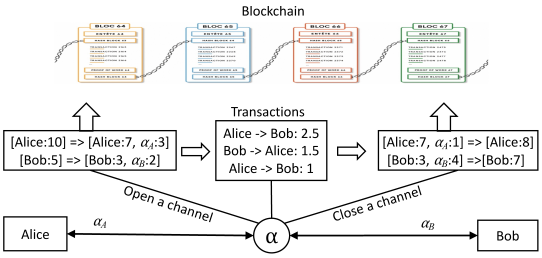
Either Alice or Bob can close the channel, usually because their balances are insufficient for more transactions or they do not agree on channel balances. Their final channel balances are committed to the blockchain.

B. Payment networks

It is usually impossible for a node to establish a payment channel with every other node due to the high overhead of maintaining a large number of smart contracts. To enable transactions between two nodes without direct channel connection, the concept of payment network consisting of multiple payment channels has been proposed. An example of payment networks with unidirectional channels is shown in Fig. 2. If node A would like to pay node C, it can ask node B to forward



(a) A unidirectional payment channel



(b) A bidirectional payment channel

Fig. 1. Payment channels.

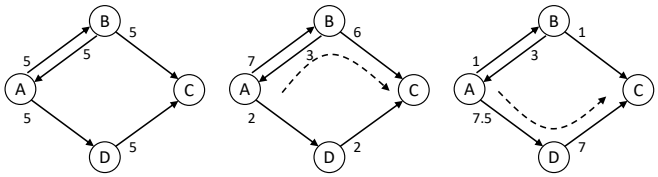
this payment, which forms a payment path (A, B, C). A critical challenge here is to guarantee that the node B forwards correct amount of coins, without stealing the money. The Hash Time-Lock Contract (HTLC) [7] has been proposed to address this challenge. The basic idea of HTLC is to let both A and B lock the same amount of coins on the channels along the path. The HTLC uses cryptographic techniques to guarantee that node B can get the coins locked on the channel (A,B) only when it successfully makes the payment to C.

In practice, intermediate nodes along a payment path do not forward payments for free, but charge some forwarding fees that could be a fixed amount of coins or be proportional to the amount of forwarded payments, which depends on their charging policies. Due to the existence of forwarding fees, each node needs to prepare coins more than its actual payment demand. In the example shown in Fig. 2, node A intends to pay 5 coins to C. Since node B charges 1 coins as forwarding fees, node A needs to pay 6 coins to B. There are usually multiple paths between two nodes and the ones with less forwarding fees are always preferred.

C. Motivation

In larger payment networks with many payment demands, it is challenging to decide how many coins should be deposited into these payment channels, and which paths should be selected for multi-hop transactions. We use an example in Fig. 2 to elaborate on these challenges that motivate our work. This example contains 4 nodes, each of which has a budget of 10 coins that can be used for creating unidirectional payment channels.

1) Balance planning: In an intuitive scheme, we suppose that each node knows his payment receivers, but is unaware of payment amount. This assumption has been widely adopted by existing work [9]–[12]. As shown in Fig. 2(a), each node uses



(a) (b) (c)

Fig. 2. A motivation example of 4-node payment network. The payment demands are as follows: $A \to B : 1$, $A \to C : 5$, $A \to D : 2$, $B \to A : 3$, $B \to C : 1$, $D \to C : 2$. Node B charges 1 as forwarding fee, and D charges 0.5.

all its budget to create payment channels. Note that node A can ask either B or D to forward its payments to node C. We apply the rebalancing scheme advocated by [12], which deposits an identical amount of coins among channels belonging to the same node. In this example, node A deposits 5 coins into channels (A, B) and (A, D), respectively. However, such a kind of budget allocation cannot satisfy A's payment demands no matter which path it chooses. For example, if A chooses the path (A, B, C), the total payment over the channel (A, B) is 6, exceeding its payment capability. On the other hand, if A asks D to forward payments, the total payment over channel the (A, D) is 7. Meanwhile, node B wastes its budget because its payments over (B, A) and (B, C) are 3 coins and 1 coin, respectively. An alternative scheme is shown in Fig. 2(b). Suppose that node B forwards the payment from A to C, and charges 1 coin as forwarding fees. All payment demands can be satisfied and the total channel deposits are 20 coins.

2) Payment path selection: In the example shown in Fig. 2(b), the payments from A to C can go through two paths (A, B, C) and (A, D, C), which incurs different forwarding fees. The node B charges 1 coins, but node C charges 0.5. Compared with Fig. 2(b), the total channel deposits can be reduced to 19.5 if the path (A, D, C) is selected, as shown in Fig. 2(c). This example indicates that payment path selection plays a critical role in reducing the cost of multi-hop payment. Intuitively, the nodes charging less forwarding fees are always preferred, but they usually have limited budgets for payment channel construction and thus cannot afford too many forwarded transactions. Therefore, it is still challenging to decide payment paths for each demand, given the constraints of forwarding fees and node budgets.

3) Uncertainty of payment demands: In the above example, we assume that all payment demands are accurately estimated. However, such an accurate estimation is unlikely to happen in practice. Even though some payment patterns can be found, the actual amount of payment may fluctuate. For example, if the actual payment from node D to C is 3 coins, more than the expected demands of 2 coins, the channel (D, C) would be exhausted before satisfying all payment demands in Fig. 2(c).

III. System Model

A. Network model

We consider a payment network that is modeled as a graph $G(N, E)$, where N denotes the set of nodes and E is the set of

payment channel candidates. The possible neighbors of node i are maintained in set $N(i) = \{j|(i,j) \in E\}$. Each node $i \in N$ holds a budget of B_i, which can be used to create payment channels to its neighbors. These channels can be unidirectional or bidirectional ones, whose differences in balance planning will be studied in Sections IV-C and IV-D, respectively. Each node $i \in N$ charges forwarding fees proportional to the amount of forwarded coins. We let ϕ_i denote the charging proportion.

The expected amount of payments from node u to node v is denoted by d_{uv}. Note that the actual payment may randomly fluctuate around d_{uv}. We consider only one-to-one payment demands. The one-to-many or many-to-one payment demands can also be accommodated in our model by dividing them into multiple one-to-one demands. All node pairs with payment demands are maintained in set D. For each pair of nodes $(u,v) \in D$, there exist multiple payment paths, which are included in set P_{uv}. We define payment outage as events that transactions fail due to channel exhaustion along all paths in P_{uv} before satisfying d_{uv}.

Each node $i \in N$ holds a pair of public key pk_i and private key sk_i. Nodes can exchange messages with each other via an underlying peer-to-peer network. All messages are signed by private keys and are propagated through a gossip protocol [14], which guarantees that messages can be delivered within a known fixed time, but their arriving order is unnecessarily preserved.

B. Threat model

In the payment network G, there are malicious nodes who can behave arbitrarily, e.g., sending wrong messages or manipulating planning results, but they hold a limited amount of budgets that account for less than one-third of total budgets of all nodes. They can launch the Sybil attack by forging multiple identities. Since messages are signed by private keys of their senders, malicious nodes cannot compromise message exchanges among nodes. The remaining nodes in the network are honest and faithfully follow the instructions of PnP.

C. Goals

Given a payment network G, we study the balance planning problem to decide how many coins should be deposited into channel balances, so that the payment demands in D can be satisfied with a high probability. We focus on achieving the following goals.

Economy goal. An easy way to satisfy payment demands is to deposit all budgets into channel balances, especially when payment demands cannot accurately be estimated, as we assume in this paper. However, it incurs unnecessary waste when there are fewer transactions. Since coins deposited into payment channels come from blockchain accounts, the nodes locking too many coins into payment channels may be short for funds to run other smart contracts or transactions on the blockchain. Therefore, it is imperative to minimize the total amount of channel balances, while theoretically guaranteeing outage probability.

Safety goal. The main attack launched by malicious nodes is to manipulate balance planning results to increase outage probability. If the attack successes, many transactions over the payment network would fail before satisfying payment demands. Moreover, nodes need to frequently access the blockchain to reconstruct payment channels, which further burdens the blockchain. PnP needs to protect the system from this attack by guaranteeing correct balance planning results. Another kind of attacks is to deviate from balance planning results by depositing more or less coins or making excessive transactions to quickly exhaust payment channels. Fortunately, it is easy to detect such attacks and therefore to eliminate associated malicious nodes, because these activities are recorded by the blockchain or the payment network.

Efficiency goal. PnP is designed for the payment network without centralized control or trusted third-parties. In such a distributed environment, nodes need to exchange messages to conduct balance planning in a safe way. Therefore, PnP should incur less message exchanges and impose low computational burden on nodes.

PnP aims to achieve the above three goals simultaneously with a series of new designs, which will be presented in the next section.

IV. System Design

In this section, we study the balance planning for payment networks with unidirectional or bidirectional channels. We first give an overview about system design, and then elaborate on key techniques of decision committee selection and balance planning algorithms for both kinds of channels.

A. Overview

The balance planning is conducted in epochs. At the beginning of each epoch, nodes estimate their expected payments to corresponding payees. The length of each epoch depends on the tradeoff between the estimation accuracy of payment demands and the overhead of blockchain accesses. Longer epochs can reduce the frequency of blockchain accesses, but need more coins locked into channels as more transactions need to be handled. Moreover, the error of payment estimation becomes bigger, which makes the design of the balance planning algorithm harder. On the other hand, with shorter epochs, it would be easy to estimate payment demands with small errors, but payment channels with less deposits could be quickly exhausted, leading to frequent blockchain operations.

The payment network is essentially a distributed system with malicious nodes. It would be unsafe to rely on a centralized control or a trusted third-party to run the balance planning algorithm, because they could be compromised by malicious nodes. At the beginning of each epoch, PnP uses the cryptographic sortition to randomly select a set of nodes, which is called the decision committee, responsible for running the balance planning algorithm. The cryptographic sortition can work in a distributed environment and require no interaction among nodes. As the first line shown in Procedure 1, the function Sortition$(\cdot)$ takes the private key as well as

Procedure 1 Procedure of each node $i \in N$ in epoch r

1: $(hash_i, \pi_i, s_i) = \text{Sortition}(sk_i, r, seed_r)$;
2: **if** $s_i = True$ **then**
3: Send the message $\langle hash_i, \pi_i, s_i \rangle$ to all nodes in $N/\{i\}$;
4: $N_c \leftarrow i$;
5: **end if**
6: Listen and receive sortition messages from other nodes;
7: **for** each received message $\langle hash_j, \pi_j, s_j \rangle$ **do**
8: **if** $\text{VerifySortition}(pk_j, hash_j, \pi_j, r, seed_r) = True$ **then**
9: $N_c \leftarrow j$;
10: **end if**
11: **end for**
12: Send message $\langle B_i, \{d_{i0}, d_{i1}, d_{i2}, ...\}, N(i) \rangle$ to all nodes in N_c;
13: **if** $s_i = True$ **then**
14: Listen and receive messages of payment demands;
15: Solve the balance planning problem of uniBP_linear or biBP_linear;
16: Run the PBFT [16] protocol to achieve agreement on solving results;
17: Send balance planning results to nodes in $N/\{i\}$;
18: **else**
19: Listen and receive balance planning results;
20: **end if**

other information, whose details will be given later, as input and returns a Boolean value s_i that indicates whether node i is selected as a decision committee member. Meanwhile, it also returns a hash value $hash_i$ and a proof π_i, which can be used to verify the correctness of sortition results. Although a similar sortition method has been proposed by [15], it targets on voting for block proposal and cannot be directly applied in our scenario.

The nodes who are selected as decision committee members disseminate their sortition results using the gossip protocol. Meanwhile, every node listens to communication channels and receives sortition results. For each received sortition message, node i verifies its correctness using the function VerifySortition$(\cdot)$ and maintains a decision committee by including any node j with $s_j = True$, as shown in lines 7-11. Once the decision committee is setup, all nodes send their budgets, payment demands and neighbor information to the committee. The members of the decision committee solve the balance planning problem independently. It should be noted that malicious nodes may be selected into the decision committee, and they can arbitrarily manipulate balance g results. Therefore, PnP runs the PBFT [16] protocol among decision committee members, so that they can achieve an agreement on the correct planning results. After that, the planning results are sent to the corresponding nodes in the network.

Procedure 2 $\text{Sortition}(sk_i, r, seed_r)$

1: $\langle hash_i, \pi_i \rangle = \text{VRF}_{sk_i}(r, seed_r)$;
2: $\delta_i = \frac{w_i \overline{w}}{\sum_{j \in N} w_j^2}$;
3: **if** $\frac{hash_i}{2^{hashlen}} \leq \delta_i$ **then**
4: return $\langle True, hash_i, \pi_i \rangle$;
5: **else**
6: return $\langle False, hash_i, \pi_i \rangle$;
7: **end if**

Procedure 3 $\text{VerifySortition}(pk_i, hash_i, \pi_i, r, , seed_r)$

1: **if** $\text{VerifyVRF}_{pk_i}(hash_i, \pi_i, r, seed_r) = True$ **then**
2: $\delta_i = \frac{w_i \overline{w}}{\sum_{j \in N} w_j^2}$;
3: **if** $\frac{hash_i}{2^{hashlen}} \leq \delta_i$ **then** return $True$;
4: **else** return $False$; **end if**
5: **else** return $False$; **end if**

B. Selection of Decision Committee

The selection of decision committee is critical for balance planning. A good design should be able to defend against Sybil attacks by minimizing the probability of selecting malicious nodes as committee members. In addition, it should require no centralized control and has low complexity with minimum message exchanges among nodes.

To achieve the above design goals, we propose the cryptographic sortition to select a subset of nodes as the decision committee according to their payment contributions in history. Specifically, we let W denote the total amount of channel deposits of all nodes during the last τ rounds, and the portion contributed by node i is w_i, which is also called the sortition weight. We aim to select a decision committee whose total weight is about $\overline{w}$ and $\overline{w} \leq W$. The cryptographic sortition can guarantee that the probability of choosing node i into the decision committee is proportional to w_i, so that malicious nodes with limited budgets have low probability of being selected into the decision committee. The information of sortition weights $\{w_i | i \in N\}$ can be obtained from the blockchain. When nodes open or close payment channels, their channel balances are included in channel contracts stored by the blockchain, which can be seen by all nodes.

The pseudo codes of sortition function are shown in Procedure 2. A verifiable random function $\text{VRF}_{sk_i}(r, seed_r)$ [17] is invoked first, where r is the epoch ID and $seed_r$ is a random seed associated with r. The $seed_r$ is generated by the decision committee in the epoch $r-1$ using the similar techniques in [4], [5]. The VRF function returns a hash value $hash_i$ and a proof π_i, which uniquely depend on the private key sk_i, r and $seed_r$. The hash value $hash_i$ contains $hashlen$ bits. Then, we check whether $\frac{hash}{2^{hashlen}}$ is no greater than the threshold δ_i, which indicates the probability of selecting node i into to the decision committee. The value of δ_i can be calculated according to $\{w_i | i \in N\}$ readily available on the blockchain as follows. As an admission control, we constrain the total weight of decision committee members as $\overline{w} \leq W$. Therefore, we

have $\sum_{i\in N}\delta_i w_i = \overline{w}$. Meanwhile, the value of δ_i should be proportional to w_i, i.e., $\frac{\delta_i}{\delta_j} = \frac{w_i}{w_j}$. Based on above constraints, each node can easily calculate the values of $\delta_i = \frac{w_i\overline{w}}{\sum_{j\in N} w_j^2}$ independently, without message exchanges.

As shown in Procedure 3, to verify sortition results, we use $\text{VerifyVRF}_{pk_i}(\cdot)$ [17] to check the correctness of $hash_i$ according to public key pk_i, π_i, r and $seed_r$. If $hash_i$ is correct, we continue to check wether $\frac{hash_i}{2^{hashlen}}$ is no greater than δ_i that can be calculated according to $\{w_i|i \in N\}$. The function returns $True$ if passing both condition checks. Otherwise, $False$ is returned.

C. Balance planning for unidirectional channels

To achieve the economy goal of network planning, we need to design an algorithm to minimize the amount of coins deposited into payment channels. We use b_{ij} to denote the initial balance of the unidirectional channel $(i,j) \in E$, and $f_{uv}^p(i,j)$ to denote the payment from node u to v over the channel (i,j) along the path $p \in P_{uv}$. The balance planning problem can be formulated as follows.

uniBP_chance:
$$\min \sum_{(i,j)\in E} b_{ij}$$
$$\sum_{j\in N(i)} b_{ij} \le B_i, \forall i \in N; \quad (1)$$
$$(1-\phi_j) f_{uv}^p(i,j) = f_{uv}^p(j,k), \forall (i,j)\in p, (j,k)\in p, p\in P_{uv}, (u,v)\in D; \quad (2)$$
$$\sum_{(u,v)\in D}\sum_{p\in P_{uv}} f_{uv}^l(i,j) \le b_{ij}, \forall (i,j)\in E. \quad (3)$$
$$\Pr\Big\{\sum_{p\in P_{uv}}\sum_{(u,j)\in p} f_{uv}^p(u,j) \ge d_{uv}\Big\} \ge 1-\epsilon, \forall (u,v)\in D. \quad (4)$$

Each node $i \in N$ has a limited budget B_i for channel construction, which is constrained by (1). Due to the forwarding fees charged by each intermediate node i along a path $p \in P_{uv}$, the payment of its previous-hop channel (i,j) should be more than the next-hop channel (j,k), which is represented by constraint (2). The constraint (3) indicates that the total payments over each channel $(i,j) \in E$ cannot exceed its initial balance b_{ij}. Finally, the total payment over all possible paths should be no less than the demand d_{uv}, which can be expressed by

$$\sum_{p\in P_{uv}}\sum_{(u,j)\in p} f_{uv}^p(u,j) \ge d_{uv}, \forall (u,v)\in D. \quad (5)$$

However, it is hard to accurately estimate the payment demand d_{uv} between each pair of nodes in practice. To circumvent this difficulty, we model the payment demand d_{uv} as a random variable, and express the constraint in a chance-constrained form (4), which allows a small outage probability ϵ.

The main difficulty in solving the problem uniBP_chance is to address the chance constraint (4). In the following, we apply the Bernstein approximation [18] to transfer (4) into a linear constraint, while guaranteeing the feasibility of corresponding solutions. Suppose that the distribution of d_{uv} is bounded within $[d_{uv}^{min}, d_{uv}^{max}]$, which can be obtained by analyzing the transaction records in history. By defining $\alpha_{uv} = \frac{1}{2}(d_{uv}^{max} - d_{uv}^{min})$ and $\beta_{uv} = \frac{1}{2}(d_{uv}^{max} + d_{uv}^{min})$, d_{uv} can be normalized within $[-1,1]$ as follows:

$$\xi_{uv} = \frac{d_{uv} - \beta_{uv}}{\alpha_{uv}} \in [-1,1]. \quad (6)$$

The chance constraint (4) can be equivalently written as

$$\Pr\Big\{\beta_{uv} - \sum_{p\in P_{uv}}\sum_{(u,j)\in p} f_{uv}^p(u,j) + \alpha_{uv}\xi_{uv} \le 0\Big\} \ge 1-\epsilon, \forall (u,v)\in D. \quad (7)$$

According to Bernstein approximation [18], the above constraint can be approximated by:

$$\inf_{\rho>0}\Big[\beta_{uv} - \sum_{p\in P_{uv}}\sum_{(u,j)\in p} f_{uv}^p(u,j) + \rho\log\frac{1}{\epsilon} + \rho\Omega(\rho^{-1}\alpha_{uv})\Big] \le 0, \forall (u,v)\in D, \quad (8)$$

where $\Omega(\rho^{-1}\alpha_{uv})$ can be expressed as

$$\Omega(\rho^{-1}\alpha_{uv}) = \log\mathbb{E}[\exp(\rho^{-1}\alpha_{uv}\xi_{uv})]. \quad (9)$$

Theorem 1: For a given ϵ, suppose that there exists a solution such that (8) holds. Then, this solution satisfies (4).

The proof can be completed by following [18]. We omit details due to space limit. Moreover, another useful conclusion offered by [18] is that the upper bound of $\Omega(w)$ is given by

$$\Omega(w) \le \max\{\mu^- w, \mu^+ w\} + \frac{\sigma^2 w^2}{2}, \quad (10)$$

where $-1 \le \mu^- \le \mu^+ \le 1$ and $\sigma \ge 0$ are constants that depend on the given probability distribution.

By applying (10), the constraint (8) can be approximated by

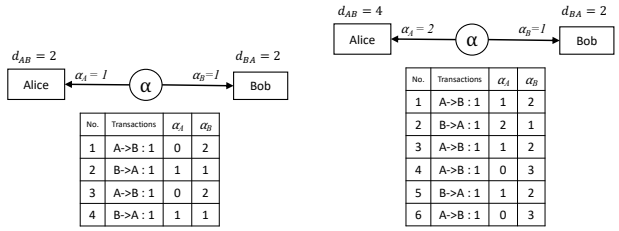
$$\inf_{\rho>0}\Big[\beta_{uv} - \sum_{p\in P_{uv}}\sum_{(u,j)\in p} f_{uv}^p(u,j) + \rho\log\frac{1}{\epsilon} + \mu^+\alpha_{uv} + \frac{(\sigma\alpha_{uv})^2}{2\rho}\Big] \le 0, \forall (u,v)\in D. \quad (11)$$

The $\inf(\cdot)$ in above constraint can be removed by substituting $\rho = \frac{\sigma\alpha_{uv}}{\sqrt{2\log\epsilon^{-1}}}$, so that (11) can be equivalently written as

$$\beta_{uv} - \sum_{p\in P_{uv}}\sum_{(u,j)\in p} f_{uv}^p(u,j) + \mu^+\alpha_{uv} + \sqrt{2\log\epsilon^{-1}}\sigma\alpha_{uv} \le 0, \forall (u,v)\in D. \quad (12)$$

Finally, we obtain a linear programming formulation as follows.

uniBP_linear:
$$\min \sum_{(i,j)\in E} b_{ij}$$
$$(1)-(3), \text{ and } (12).$$



No.	Transactions	α_A	α_B
1	A->B : 1	0	2
2	B->A : 1	1	1
3	A->B : 1	0	2
4	B->A : 1	1	1

(a)

No.	Transactions	α_A	α_B
1	A->B : 1	1	2
2	B->A : 1	2	1
3	A->B : 1	1	2
4	A->B : 1	0	3
5	B->A : 1	1	2
6	A->B : 1	0	3

(b)

Fig. 3. Examples of payments over bidirectional channels.

Since BP_linear contains real variables and linear constraints, it can be efficiently solved by well-developed algorithms, e.g., simplex algorithms or interior-point algorithms [19].

D. Balance Planning for Bidirectional Channels

Bidirectional payment channels have stronger payment capability by allowing two-way payments. Since a node can make payments more than its initial deposit, the constraint (4) cannot be applied for bidirectional channels. We use an example in Fig. 3 to motivate the formulation of bidirectional channels. We first assume that Alice and Bob have payment demands of 2 coins to each other, i.e., $d_{AB} = 2$ and $d_{BA} = 2$. It is unnecessary to let them deposit 2 coins as initial balances. As shown in Fig. 3(a), their payment demands can be satisfied with initial balances of $\alpha_A = 1$ and $\alpha_B = 1$. If the payment demand from Alice to Bob grows to 4 coins, Alice needs to deposit 2 coins, as shown in Fig. 3(b). By carefully examining the payment process in the above examples, we find that the Alice's initial balance α_A should be no less than the amount of incoming payments minus that of outgoing payments. Bob has no such a constraint because it makes less payment to Alice. The above facts can be expressed by:

$$\max\Big\{0, \sum_{(u,v)\in D}\sum_{p\in P_{uv}} f_{uv}^{p}(i,j) - \sum_{(u,v)\in D}\sum_{p\in P_{uv}} f_{uv}^{p}(j,i)\Big\} \le b_{ij}, \forall (i,j)\in E. \quad (13)$$

According to (13), Bob's initial balance can be 0 in the example shown in Fig. 3(b). However, this setting would stuck all Bob's payments until he receives some coins from Alice. In our model, we impose no requirement on the payment delay and it is fine to finish all payments before the end of the epoch. However, we still let each node deposits some coins b_{ij}^{min} as initial channel balance to reduce payment delay in practice. The value of b_{ij}^{min} can be estimated according to the payment history and the requirement of payment delay, but it cannot provide strong delay guarantee in theory. The balance planning problem for payment networks with bidirectional channels can be described by:

biBP_chance:

$$\min \sum_{(i,j)\in E} b_{ij}$$

$$\sum_{(u,v)\in D}\sum_{p\in P_{uv}} f_{uv}^{p}(i,j) - \sum_{(u,v)\in D}\sum_{p\in P_{uv}} f_{uv}^{p}(j,i) \le b_{ij}, \quad \forall (i,j)\in E; \quad (14)$$

$$b_{ij} \ge b_{ij}^{min}, \forall (i,j)\in E; \quad (15)$$

$$(1), (2), \text{and } (4).$$

The above problem can be transferred as a linear version biBP_linear using the similar techniques developed in Section IV-C.

E. Complexity Analysis

We first study the number of message exchanges incurred by PnP. The number of nodes in the network is denoted by $n = |N|$, and m nodes are expected to be chosen as the decision committee. Since sortition process needs no interaction among nodes, only the dissemination of sortition results incurs $O(mn)$ messages. In addition, $O(mn)$ messages are needed for reporting payment demands by nodes. The PBFT protocol incurs $O(m^2)$ message exchanges among decision committee members. Finally, counting $O(mn)$ messages for disseminating planning results, the communication complexity of PnP is $O(mn + m^2)$. Fast polynomial-time algorithms exist for verifiable random functions and linear programming problems uniBP_linear and biBP_linear. Considering blockchain nodes are usually equipped with strong computational capability, PnP imposes minor computational overhead on them.

V. Performance Evaluation

A. Experiment settings

We implement PnP as a pluggable service that can interact with the Lightning Network Daemon (lnd) [13], a complete implementation of a Lightning Network node, over RPC interfaces. PnP contains two main modules, i.e., the sortition and the planning solver. In the sortition module, we use Curve 25529 to implement signatures and VRFs. The hash values are generated according to SHA-256. We create a testbed of 30 nodes to evaluate the performance of PnP. The potential payment channels among nodes are randomly generated according to the Watts-Strogatz small-world model [20], which has been widely adopted to describe real-world connections. In our settings, each node holds a random budget among [1000, 1500] coins and can create channels to at most 6 neighbors. Forwarding fees are charged proportional to the amount of forwarded payments, and the ratios are randomly distributed within [0.01, 0.1]. We also set $\overline{w} = W/3$, i.e., the sortition weights held by decision committee is about one third of total weights of all nodes. The payment outage probability is set to 0.05. All experiments run 50 epochs. Since PnP is the first work for balance planning, we compare it with several variants using different design choices as follows.

PNP_greedy: For each demand, we first choose the shortest path with the minimum forwarding fee from the candidate path

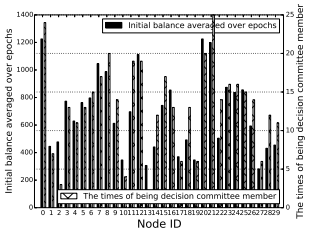


Fig. 4. The sortition results of different nodes.

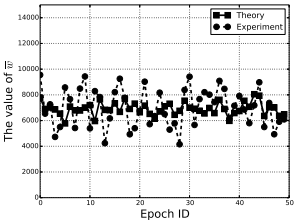


Fig. 5. The sortition weights of decision committee in different epochs.

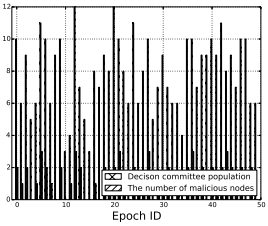


Fig. 6. The population of decision committee.

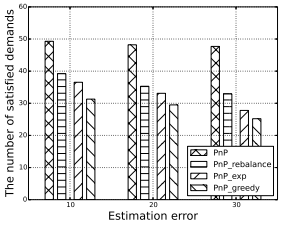


Fig. 7. The number of satisfied demands under different estimation errors.

set and open channels along this path. If the residual budget of this path is insufficient to satisfy this demand, we then choose the next shortest path, until all candidate paths are used.

PNP_exp: The decision committee solves balance planning problems based on the expectation of payment demands, by replacing the constraint (4) with (5).

PNP_rebalance: We combine the rebalancing idea of REVIVE [12] with PnP by letting nodes periodically rebalance their channels. Due to the overhead of rebalancing, we constrain rebalancing operations within 3 times in each epoch.

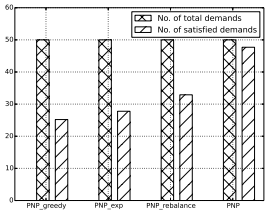


Fig. 8. The number of satisfied demands.

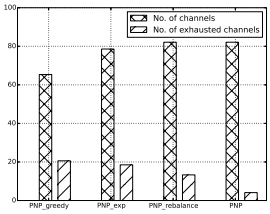


Fig. 9. The number of exhausted channels.

B. Experiment results

1) Sortition results: We first evaluate the sortition function by checking the probability of being decision committee members. As illustrated in Fig. 4, black bars indicate nodes' sortition weights averaged over 50 epochs and the others represent the times of being decision committee members. The results show that the chances of being chosen roughly coincide with the proportion of sortition weights. The nodes contributing more channel deposits have more opportunities to be decision committee members. Since malicious nodes hold limited budgets, they have lower probabilities of joining the committee and therefore manipulating planning results. In Fig. 5, we show the value of $\overline{w}$ and the total weights hold by the decision committee in each epoch. Although a slight deviation exists, the average error is only 17%. We randomly mark several nodes as malicious ones, and they always account for less than one-third of the decision committee population, as shown in Fig. 6, so that the PBFT protocol can well guarantee the correctness of planning results.

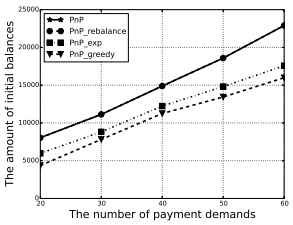
2) Balance planning results: We show the number of satisfied payment demands under different demand estimate errors in Fig. 7. The estimation error has little effect, less than 3%, on PnP, but PnP_exp strongly depends on the estimation accuracy. The number of satisfied demands under PnP_exp increases from 27.8 to 36.6 when the average error reduces from 30 to 10.

The number of satisfied demands under different schemes is shown in Fig. 8. The results are averaged over 50 epochs and each epoch has 50 demands whose estimation error is about 20. PnP completes about 95% of payment demands, which is coincident with the outage setting, while PnP_greedy performs worst, with less than 30 demands satisfied. We set the minimum amount of coins included in transactions to 0.2. If the channel balance is less than 0.2, we say this channel is exhausted. Fig. 9 show the number of exhausted channels. We observe that PnP creates about 82 channels on average, but exhausts only about 4 channels. Due to rebalancing, PnP_rebalance has about 13.3 channels exhausted, outperforming PnP_greedy and PnP_exp.

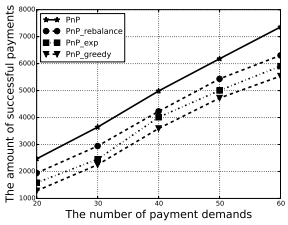
We then study the influence of payment demands on initial channel deposits and completed payments. As shown in Fig. 10(a), the amount of budgets deposited into channels increases as the number of demands grows from 20 to 60. Note that PnP and PnP_rebalance use the same amount of budgets because PnP_rebalance also uses the results of chance-constrained optimization, but periodically rebalancing channels. Although PnP and PnP_rebalance consume more budgets, they have stronger payment capability in practice. The evidences can be found in Fig. 10(b) that shows the amount of completed payments under different number of demands.

We then randomly remove some nodes from the network and show the influence of network size in Fig. 11. As the growth of network size, the total budgets deposited into the network increase. That is because the paths between two random nodes become longer in larger networks. All nodes along a path need to deposit their channels to forward payment, leading to the growth of channel deposits. On the other hand, there is slight decreasing of successful payments as the growth of network size because the probability of channel exhaustion increases in longer paths. Although PnP uses more budgets, it enables more successful payments thanks to the chance-constrained optimization.

Finally, we study the performance of PnP with unidirectional channels and bidirectional channels. As shown in Fig. 12, PnP with bidirectional channels needs less budget, but

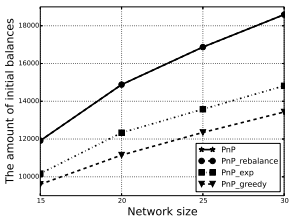


(a) The amount of channel deposits under different payment demands.

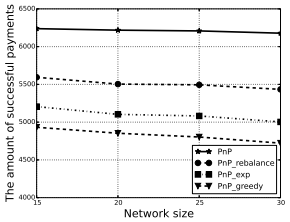


(b) The amount of successful payments under different payment demands.

Fig. 10. The influence of payment demands.



(a) The amount of channel deposits under different network sizes.



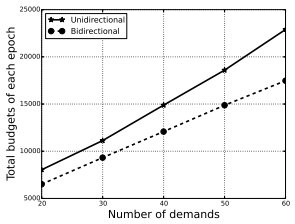
(b) The amount of successful payments under different network sizes.

Fig. 11. The influence of network size.

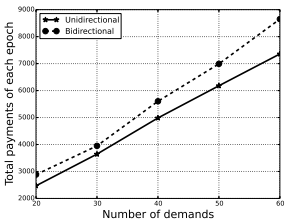
with strong payment capability. Moreover, the gap becomes larger when more payment demands are generated. That is reasonable because bidirectional channels show advantages in handling payments with reverse direction, which exist with higher probability under more payment demands.

VI. Related work

1) Blockchain: Bitcoin [1] is the first successful cryptocurrency based on blockchain, which is a distributed transaction ledger maintained by a number of participants connected over a peer-to-peer network. To keep a consistent view of the ledger, participants run a protocol called Nakasato consensus based on a proof-of-work (PoW) process. Inspired by the success of Bitcoin, various cryptocurrencies with different features have been invented [21]–[23]. For example, Ethereum [24] has extended blockchain to support smart contracts, and Zcash [25] protects user privacy by enabling confidential transactions using zero-knowledge proof. However, almost all popular blockchain-based cryptocurrencies face a common challenge of poor scalability, which stems from the consensus protocol that involves all blockchain nodes. A straightforward idea is to improve the performance of consensus protocols, which has been paid a great attention by both academia and industry. Algorand [15] uses an enhanced Byzantine Agreement protocol to reach consensus among users. OmniLedger [4] divides blockchain nodes into several shards and optimizes performance via parallel intra-shard transaction processing. RapidChain [5] is also a sharding-based consensus protocol that achieves complete sharding of communication, computation and storage.



(a) The amount of budgets under different payment demands.



(b) The amount of successful payments under different payment demands.

Fig. 12. Comparison of unidirectional and bidirectional channels

2) Payment channels and payment networks: The most famous payment network is the Lightning Network [7] that has been designed for Bitcoin. The bidirectional payment channels have been studied by [26]. Raiden Network [8] has been proposed as an off-blockchain scaling solution for Ethereum blockchain. Sprites [9] has enhanced the performance of payment channel by reducing the worst-case cost that each hop along the route may incur. Flare [27] has been designed as a hybrid routing algorithm for payment routing in Lightning Network. Yu et al. [28] have proposed a distributed routing mechanism called CoinExpress. The concept of general state channel has been proposed by [29] to allow the execution of arbitrary complex smart contracts on payment networks. Dziembowski et al. [10] have proposed the virtual payment channels that apply the channel technique recursively, so that two nodes without direct channel connection can still enjoy the benefits of payment channels, without building new channels. Molavolta et al. [11] have shown the privacy challenge of existing Lightning Network and have proposed a protocol called Fulgor to enhance the privacy guarantee. Moreover, they have addressed the concurrency issue of payment networks. REVIVE [12] allows a set of nodes in the payment network to securely rebalance their channels, but requiring a trusted third-party to run the rebalancing algorithm.

VII. Conclusion

This paper presents PnP, a balance planning service for payment networks. Given inaccurate estimated payment demands, PnP can minimize the total funds deposited into payment channels while guaranteeing small outage probability. Meanwhile, PnP requires no trusted third-party and can resist attacks by designing an efficient cryptographic sortition algorithm to select a decision committee responsible for running balance planning algorithms. We implement PnP on Lighting Network Daemon and experimental results on a testbed of 30 nodes show that PnP outperforms other designs.

Acknowledgment

This research is partially supported by the JSPS Grants-in-Aid for Scientific Research grant number JP19K20258. Prof. Wanlei Zhou's work is partially supported by the ARC Discovery Project DP180102828.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] Ethereum, *URL: https://www.ethereum.org*.

[3] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, 2016, pp. 17–30.

[4] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 583–598.

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

[6] M. Trillo, "Stress test prepares visanet for the most wonderful time of the year," *URL: http://www. visa. com/blogarchives/us/2013/10/10/stress-testprepares-visanet-for-the-most-wonderful-time-of-the-year/index. html*, 2013.

[7] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," *See https://lightning. network/lightning-network-paper. pdf*, 2016.

[8] R. network, *URL: https://raiden.network*.

[9] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, "Sprites: Payment channels that go faster than lightning," *CoRR abs/1702.05812*, 2017.

[10] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptographic currencies," IACR Cryptology ePrint Archive 2017, Tech. Rep., 2017.

[11] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, Texas, USA, 2017, pp. 455–471.

[12] R. Khalil and A. Gervais, "Revive: Rebalancing off-blockchain payment networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 439–453. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134033

[13] L. N. Daemon, *URL: https://github.com/lightningnetwork/lnd*.

[14] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, Nov 2000, pp. 565–574.

[15] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, Shanghai, China, 2017, pp. 51–68.

[16] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, Louisiana, USA, 1999, pp. 173–186.

[17] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, Oct 1999, pp. 120–130.

[18] A. Ben-Tal and A. Nemirovski, "Selected topics in robust convex optimization," *Math. Program.*, vol. 112, no. 1, pp. 125–158, Mar. 2008.

[19] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[20] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[21] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, Toronto, Canada, 2018, pp. 967–984.

[22] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi, "Solidus: Confidential distributed ledger transactions via pvorm," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, Dallas, Texas, USA, 2017, pp. 701–717.

[23] N. Narula, W. Vasquez, and M. Virza, "zkledger: Privacy-preserving auditing for distributed ledgers," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2018, pp. 65–80.

[24] V. Buterin, F. Vogelsteller *et al.*, "Ethereums white paper," 2014.

[25] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, May 2014, pp. 459–474.

[26] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Proceedings of the 17th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, 2015, pp. 3–18.

[27] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An approach to routing in lightning network," *White Paper (bitfury. com/content/5-white-papers-research/whitepaper_flare_an_approach_to_routing_in_lightning_n etwork_7_7_2016. pdf)*, 2016.

[28] R. Yu, G. Xue, V. T. Kilari, D. Yang, and J. Tang, "Coinexpress: A fast payment routing mechanism in blockchain-based payment channel networks," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, July 2018, pp. 1–9.

[29] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 949–966. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243856