RobustPay: Robust Payment Routing Protocol in Blockchain-based Payment Channel Networks

Yuhui Zhang Dejun Yang

***Abstract*—The past decade has witnessed an explosive growth in cryptocurrencies, but the blockchain-based cryptocurrencies have also raised many concerns, among which a crucial one is the scalability issue. Suffering from the large overhead of global consensus and security assurance, even the leading cryptocurrencies can only handle up to tens of transactions per second, which largely limits their applications in real-world scenarios. Among many proposals to improve the cryptocurrency scalability, one of the most promising and mature solutions is the payment channel network (PCN), which offers the off-chain settlement of transactions with minimal involvement of expensive blockchain operations. However, transaction failures may occur due to external attacks or unexpected conditions, e.g., an uncooperative user becoming unresponsive. In this paper, we present a distributed robust payment routing protocol RobustPay to resist transaction failures, which achieves robustness, efficiency and distributedness. Moreover, we modify the original HTLC protocol and adapt it to the robust payment routing protocol.**

***Index Terms*—Cryptocurrency, payment channel network, routing, blockchain**

I. Introduction

Over the past decade, the blockchain-based cryptocurrencies have risen to more than $80 billion in peak capital, including Bitcoin [7], Ethereum [12], and Ripple [14]. Nevertheless, there are concerns that blockchain-based crytocurrencies can be widely applicable when scaling up. A crucial concern is that, when adding a block to the blockchain to verify the transactions, the network capability to process transactions is limited by a certain maximum rate, due to the necessary proof-of-work calculations that needs to be carried out by generating a number whose hash value that starts with a pre-decided number of zeros. Take the Bitcoin blockchain as an example, the maximum number of transactions per second (tps) is only 7 [2], which is not comparable to over $47,000$ peak tps processed by Visa [18].

The payment channel network (PCN) was proposed to tackle the scalability issues [10]. PCNs have been employed to develop Bitcoin's Lightning Network [10] and Ethereum's Raiden Network [8]. Section II provides an in-depth discussion of the PCN working mechanism. A simple illustration of PCN is shown in Fig. 1. PCNs can process instant and less valuable payments without involving blockchain transactions which are slow and expensive. Only the initial and final balances of each channel are required to be registered. It allows a payment sent to the recipient through multiple hops between payment channels. In the payment transferring process, hosts of the channels on the route can charge fees accordingly. Therefore, the key motivation is to optimize the routing in PCNs, and guarantee the success of a payment.

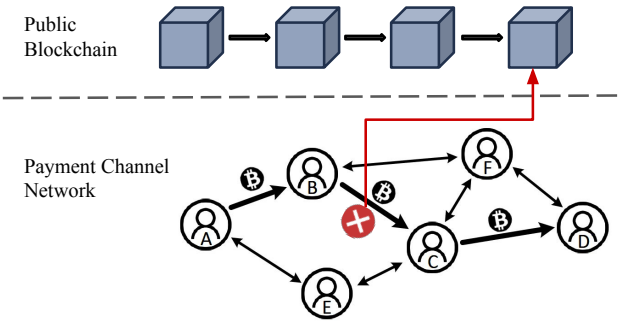

Fig. 1. Payment channel network (PCN) with a payment transaction from A to D. The transactions are off-chain. When two parties disagree with the transaction history, the transaction history will be published to the blockchain for verification. The dishonest party will be punished with a penalty.

Several reported research works have investigated payment routing in PCNs [6, 11, 15, 16, 20]. However, these efforts either emphasize on privacy [6, 16], or underestimate the importance of key realistic constraints such as the transaction fees [6, 11, 15, 16, 20]. Recently, Zhang *et al.* [21] designed an optimal algorithm CheaPay, which generates a payment path that minimizes the transaction fee in PCNs and satisfies both the feasibility and timeliness constraints. However, CheaPay only constructs a single path for a payment request, and therefore cannot resist transaction failures due to unexpected conditions or external attacks along the path. Therefore, PCNs are expected to provide better **robustness** against transaction failures, i.e., a payment routing protocol satisfies robustness, if it constructs two or more node-disjoint payment paths, where each payment path can fulfill the payment request. A payment is transferred on these payment paths simultaneously. If one path fulfills the payment first, the other path(s) will be invalidated.

In this paper, we investigate the robust payment routing by constructing two node-disjoint payment paths for a payment request in PCNs. A robust payment routing has a number of distinct characteristics, thus it is expected to satisfy a set of desired properties. First, a robust payment routing protocol should satisfy **efficiency**, i.e., to minimize the routing

Zhang and Yang are affiliated with Colorado School of Mines, Golden, CO 80401. Email:{yuhzhang, djyang}@mines.edu This research was supported in part by NSF grants 1525920, 1704092, 1717197, and 1717315. The information reported here does not reflect the position or the policy of the federal government.

978-1-7281-2700-2/19/$31.00 2019

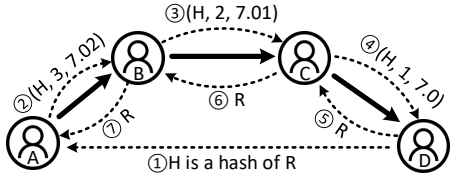

Fig. 2. Hashed time-lock contract (HTLC). The sender A sends a payment of 7 to the recipient D via B and C with an HTLC tolerance of 3. Assume that the transaction fee charged by each user is 0.01. Circled numbers represent the sequence of the operations.

and payment latency incurred by transmitting a payment through multiple payment paths simultaneously. Secondly, a robust payment routing should satisfy **distributedness**, as no central administrative operator exists in PCNs. we propose RobustPay, a robust payment routing protocol. **The main contributions of this paper are**:

- To the best of our knowledge, we are the first to consider the robust payment routing protocol, which provides resistance to transaction failures in PCNs.
- We investigate important design goals of payment routing in PCN, which are referred to as robustness, efficiency and distributedness.
- We propose RobustPay, a distributed **Robust Pay**ment routing protocol against transaction failures and enhance the robustness for payment routing in PCNs by constructing two node-disjoint paths for a payment request. We also modify the original HTLC protocol and adapt it to the robust payment routing to guarantee efficiency.

The remainder of the paper is organized as follows. In Section II, we present the background and system overview of PCNs and outline the design goals. In Section III, we illustrate the robust payment routing protocol RobustPay, demonstrate the design of the routing algorithm and analyze the properties. We summarize this paper in Section IV.

II. Background and System Overview

In this section, we provide the necessary background on permissionless blockchains such as Bitcoin and Ethereum, and present an overview of our payment channel network system.

A. Decentralized Ledger

Cryptocurrencies like Bitcoin [7], Ethereum [12], and Ripple [14] are based on the blockchain technology, which is an append-only decentralized ledger of transactions shared among mutually distrusted entities. However, the consensus algorithm, e.g. proof-of-work in Bitcoin, that guarantees the unique global state, requires large local storage due to the high levels of data replication and high computational power for adding a block containing transactions to the blockchain.

B. Payment Channel

The use of payment channels is one way to realize the off-chain approach. Two users establish a payment channel by each depositing a certain amount into a joint account and adding this transaction to the blockchain.

Now a transaction between them is essentially a channel balance update agreed upon by them. A channel is protected by multi-signature smart contracts, which ensure validity, nonequivocality and non-repudiation of the on-going transactions. When one party publishes an obsolete balance history to reverse settled transactions or to double-spend, the contract guarantees that the dishonest party is punished by granting all its remaining channel balance to the other party. This economically prevents an adversary from utility gain via dishonest behaviors. When the channel closes because either it is not needed anymore or the deposit is depleted, a closing transaction will be broadcast to the blockchain and will send deposited amount to each user according to the most recent balance.

C. Payment Channel Network

Unfortunately, payment channel alone cannot solve the scalability issue. Requiring everyone to create a payment channel with everyone else results in a large amount of on-chain transactions broadcast to the blockchain. In order to enable payments between any two users, payments can be routed through multiple hops of channels in the payment channel network (PCN) formed by users connected by payment channels. This, however, can lead to issues that a user denies performing payment transfer after receiving a preceding one, or the recipient denies receiving the payment.

D. Hashed Time-Lock Contract (HTLC).

To address these issues, the Hashed Time-Lock Contract (HTLC) mechanism has been introduced [10], as shown in Fig. 2. The recipient first generates a random value R and sends its hash H to the sender. The sender, as well as any intermediate user, includes H in the transaction contract, such that the transferred payment can be claimed by the transferee only when the secret R is provided to the transferor. In addition, each transaction is restricted by an HTLC tolerance, such that if the transferor does not receive R within the HTLC tolerance, the transferred fund will be refunded to the transferor after the HTLC expires. The unit of the HTLC tolerance, denoted by δ, is the worse-case bound on time for one on-chain transaction. Every user in the payment path sets a tolerance, which is a smaller HTLC tolerance in the outgoing payment channel than that in the incoming payment channel. For example, in Lightning Network, the tolerance is set as the number of hops until the recipient [10]. As an example, the HTLC $(H, 2, 7.01)$ from B to C in Fig. 2 means that C can receive a payment of 7.01 from B if C can provide the preimage of H within 2δ. This mechanism ensures that a user can pull the payment from its predecessor after its payment has been pulled by its successor. Note that the HTLC tolerance time is not the time of payment routing, which is fast when users are cooperative and responsive. In addition to the payment to the recipient, an HTLC also includes the transaction fees charged by the intermediate nodes for transferring the sender's payment. The fees are significantly

lower than blockchain transaction fees largely due to the time-value of locking up funds in the channel, as well as paying for the chance of channel close on the blockchain.

E. Design Goals

In the blockchain and PCN systems specified above, we derive a set of desirable design goals that a payment routing protocol should satisfy, which are elaborated below.

- **Robustness**: A payment routing protocol satisfies robustness, if it generates two or more node-disjoint payment paths, where each of them can fulfill the payment request individually. In PCNs, a node may become unresponsive due to external attacks, unexpected conditions or uncooperative behaviors, which leads to transaction failures. If the routing protocol generates only a single path for a payment request, it fails when a node on this path becomes unresponsive.
- **Efficiency**: A payment routing protocol satisfies efficiency, if it minimizes the routing and payment latency incurred by transmitting a payment through more than one path simultaneously. In the robust payment routing, only one payment path will be used to fulfill the payment request, and the other payment path(s) will be invalidated. Thus, it is necessary to guarantee that this payment path introduces the minimum latency.
- **Distributedness**: A payment routing protocol satisfies distributedness, if it dose not rely on a centralized trusted party. Centralized routing is subject to a single point of failures upon external attacks, and hence cannot be trusted by users. Instead, users need to communicate with each other and conduct local computations to find routes for payments.

III. A Distributed Robust Payment Routing Protocol in PCNs

In this section, we present the design of RobustPay. We first provide the high-level overview and intuition behind RobustPay, and then follow the design goals that are outlined in Section II-E to provide a detailed protocol description.

A. Design Rationale

A payment *transaction failure* occurs, if there are external attacks or unexpected conditions along the payment path, e.g., an uncooperative IU decides to become unresponsive by not providing the preimage of H to its transferor within the HTLC tolerance. Therefore, it is necessary to design a payment routing protocol that satisfies robustness, efficiency and distributedness. we propose a distributed robust payment routing protocol RobustPay against transaction failures for payment transmissions in PCNs.

B. Design Challenge

The challenge comes from the establishment of HTLCs, since the HTLC mechanism was originally designed to guarantee the off-chain security almost the same as the original blockchain, when routing on a single payment path. To apply the HTLC mechanism in a robust payment routing protocol, we need to modify the current HTLC protocol carefully to satisfy efficiency as well as off-chain security.

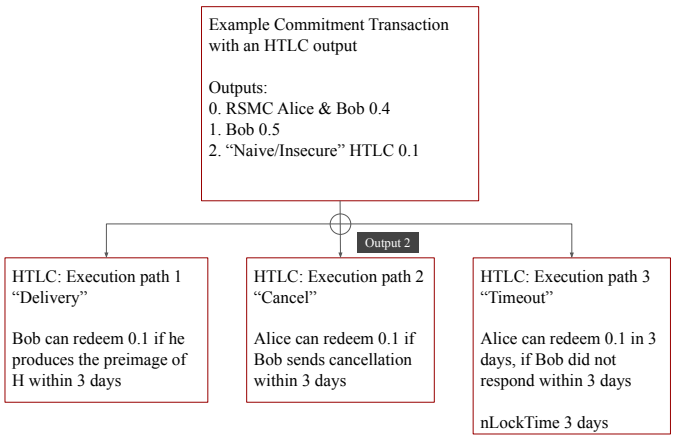

Fig. 3. Modified hashed time-lock contract (HTLC). Alice sends a payment of 0.1 to the Bob via B and C with an HTLC tolerance of 3. Note that there are two possible spends from an HTLC output. If Bob can produce the preimage of H within 3 days, Bob can redeem path 1. If Alice sends cancellation before Bob can produce the preimage of H within 3 days, Alice can redeem path 2. After three days, Alice is able to redeem path 3, if there is no response from Bob.

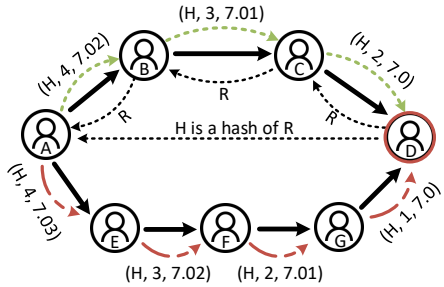

Fig. 4. Payment Forwarding in RobustPay. The sender A sends a payment of 7 to the recipient D. Two node-disjoint payment paths have been constructed. One payment path is $A \rightarrow B \rightarrow C \rightarrow D$; another payment path is $A \rightarrow E \rightarrow F \rightarrow G \rightarrow D$. HTLCs are established on both payment paths simultaneously, from the sender A to the recipient D, sequentially. The recipient D provides the preimage of H to C on the upper (green) payment path and refunds G on the lower (red) payment path.

C. Payment Path Construction

The first stage is to construct two payment paths for a payment request, such that either payment path can fulfill the payment request, and there is no *intermediate user* (IU) shared on both payment paths. Such two payment paths are referred to as a pair of node-disjoint payment paths. If a transaction failure occurs on a payment path due to unexpected conditions, the other payment path can still work to fulfill the transaction. In this stage, RobustPay can be integrated with any distributed algorithm that determines a pair of node-disjoint payment paths.

D. HTLC Establishment

A Hashed Time-Locked Contract (HTLC) is a script that permits a designated party (the transferee) to spend funds by

disclosing the preimage of a hash. It also permits a second party (the transferor) to spend the funds after a timeout is reached, in a refund situation. The original HTLC introduced in [10] was designed for payment routing in a single payment path. In the HTLC, the on hold payment is refunded to the transferor, only if the transferee does not provide the preimage of H within the HTLC tolerance. However, the HTLC does not provide the transferee with flexible choices to cancel a transaction before the expiration. Even if the transferee decides to cancel a transaction, it can only wait until the expiration of the HTLC.

To adapt the HTLC protocol to RobustPay, we modify the original HTLC to provide more flexible choices as follows: If the transferee does not provide the preimage of H within the HTLC tolerance, or if the transferee cancels the transaction before the preimage of H is provided, the on hold payment in the HTLC is refunded to the transferor. The script of the modified HTLC takes the following form, and the modification of the HTLC is highlighted:

```
OP_IF
    [HASHOP] ⟨digest⟩ OP_EQUALVERIFY OP_DUP
    OP_HASH160 ⟨seller pubkey hash⟩
OP_NOTIF
    [CANCELOP] ⟨digest⟩ OP_DROP OP_DUP
    OP_HASH160 ⟨seller pubkey hash⟩
OP_ELSE
    ⟨num⟩ [TIMEOUTOP] OP_DROP OP_DUP
    OP_HASH160 ⟨buyer pubkey hash⟩
OP_ENDIF
OP_EQUALVERIFY
OP_CHECKSIG
```

Such a modification on HTLC can provide flexible choices for PCN users. A simple illustration of the modified HTLC is shown in Fig. 3.

E. Payment Forwarding

After the payment path construction and the HTLC establishment processes, the sender can forward the payment to the recipient via the constructed payment paths. Once one of the two payment paths successfully transfers the payment to the recipient, the other payment path should be invalidated. A simple illustration of the payment forwarding is shown in Fig. 4. Two node-disjoint payment paths have been constructed in the previous stage, where A is the sender and D is the recipient. An HTLC has been created on each IC on both payment paths. Since D receives the HTLC from G on the upper (green) payment path earlier, D provides the preimage of H to C and receives the payment from C. Therefore, the lower (red) payment path is invalidated. D can choose to cancel the transaction from G to D. The on hold payment in the HTLC between G and D is refunded to G. So are the rest on hold payments in the HTLCs on the ICs along the lower (red) payment path.

IV. Conclusion

In this paper, we investigated the robust payment routing protocol to resist payment transaction failures in PCNs. We first suggested a set of crucial design goals for payment routing, which are referred to as robustness, efficiency and distributedness. Following these design goals, we presented a distributed robust payment routing protocol RobustPay consisting of three stages: Payment Path Construction, HTLC Establishment and Payment Forwarding. For Payment Path Construction, RobustPay achieved robustness by constructing two payment paths, where either payment path can fulfill the payment request. Moreover, we modified the original HTLC protocol to provide efficiency and adapted it to the robust payment routing protocol.

References

[1] R. Bellman, "On a routing problem," in *Quarterly of applied mathematics*, vol. 16, no. 1, 1958, pp. 87–90.

[2] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling decentralized blockchains," in *FC*. Springer, 2016, pp. 106–125.

[3] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *SSS*. Springer, 2015, pp. 3–18.

[4] L. R. Ford Jr, "Network flow theory," RAND CORP SANTA MONICA CA, Tech. Rep., 1956.

[5] C.-L. Li, S. T. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Applied Mathematics*, vol. 26, no. 1, pp. 105–115, 1990.

[6] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "SilentWhispers: Enforcing security and privacy in decentralized credit networks." in *IACR Cryptology ePrint Archive*, 2016, pp. 1054–1071.

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

[8] R. Network. [Online]. Available: https://raiden.network/

[9] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," in *Theoretical Computer Science*, vol. 344, no. 1. Elsevier, 2005, pp. 3–14.

[10] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016. [Online]. Available: lightning.network/lightning-network-paper.pdf

[11] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An approach to routing in lightning network," in *Whitepaper*, 2016.

[12] E. Project. [Online]. Available: https://www.ethereum.org/

[13] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and privacy in social networks*. Springer, 2013, pp. 197–223.

[14] Ripple. [Online]. Available: https://www.ripple.com/

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

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

[17] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[18] M. Trillo, "Stress test prepares visanet for the most wonderful time of the year (2013)," 2013.

[19] P. F. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very ltaiarge networks," in *SIGCOMM*, vol. 18, no. 4. ACM, 1988, pp. 35–42.

[20] 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 *ICCCN*. IEEE, 2018, pp. 1–9.

[21] Y. Zhang, D. Yang, and G. Xue, "Cheapay: An optimal algorithm for fee minimization in blockchain-based payment channel networks," in *ICC*. IEEE, 2019.