MPC: Multi-node Payment Channel for Off-chain Transactions

Hao Lei[*], Longxia Huang[*], Liangmin Wang[*†], and Jinfu Chen[*]

[*]School of Computer Science and Communication Engineering, Jiangsu University, Jiangsu, P. R. China
Email: {leihao202110, hlongxia2017}@163.com, {†wanglm, jinfuchen}@ujs.edu.cn

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838626

Abstract—**Payment channel (PC) greatly improves blockchain scalability by allowing an unlimited number of off-chain transactions instead of committing every transaction to the blockchain. However, one PC just confines to two nodes. Thus, for a node, like Cafe, who frequently receives money from multiple nodes, massive PCs need to be created, which leads to massively and repeatedly information addition of channels to the blockchain and costs much fees. In this paper, we introduce a multi-node payment channel (MPC) method to solve the problem above. MPC suits the scenario that one node frequently receives money from multiple nodes, such as Cafe, shop, etc. Compared to PC, MPC can contain nearly unlimited number of nodes, thus avoiding repetitive channel information addition to the blockchain and meanwhile reducing the fee. MPC supports the same transaction logic of PC and cooperates well with existing PC. The security, economy and efficiency of MPC are proved. We implement MPC's smart contract in Ethereum and experimental results show that MPC outperforms the existing PC.**

Index Terms—**blockchain, smart contract, fee, payment channel, off-chain transaction**

I. Introduction

Despite the growing popularity, blockchain suffers from poor scalability. For example, the most prominent cryptocurrency Bitcoin [1] processes only 7 on-chain transactions per second and the Ethereum [2] processes up to 15 on-chain transactions per second. The root of poor scalability is to reach consensus of the whole network. Although researchers have proposed some novel consensus mechanisms [3-5] supporting a lot on-chain transactions, they are still far from satisfying the performance requirement of large-scale applications. For example, Visa deals with about 1700 transactions per second [6], and the throughput cannot be satisfied by any existing blockchains.

Instead of struggling to improve the throughput of on-chain transactions, payment channel (PC) [7,8] has been proposed to improve blockchain scalability by shifting on-chain transactions to off-chain transactions. PC can be created using smart contract. Specifically, if Alice buys coffee frequently from Cafe, then Alice can lock some funds as initial balance (IB) and assigns Cafe as payee node to create a PC contract through 1 on-chain transaction. After creation, Alice and Cafe can instantly change the distribution of the IB to make off-chain transactions, without committing every transaction to the blockchain. However, the mechanisms of PC let the high efficiency of off-chain transactions confined to only two nodes. Thus, for a node, like Cafe, who frequently receives money from multiple nodes, a large number of PCs need to be created. The process of creating one PC can be simply seen as adding information of one channel and two nodes to the blockchain, so these PCs related to Cafe repeatedly added the information of channel into the blockchain, meanwhile costed expensive fees.

Multiple PCs constitute a payment channel network (PCN), so that two nodes without direct PC can still make off-chain transaction in a cross-channel manner. PCN has been supported by many blockchain systems, like the Lightning Network [7] of Bitcoin and the Raiden Network [8] of Ethereum.

Many works have been made on PC and PCN. Sprite [9] aims at improving the performance in the worst-case of cross-channel payments. CheaPay [10] optimizes the payment routine to minimize the transaction fee. Perun [11] proposes a "virtual payment channel" to avoid the involvement of the intermediary for each individual payment. Spider [12] increases throughput by splitting payments into micro-unit payments and adding a delay queue to forward micro-unit payments. Suat et al. [13] balance the payment routing by adjusting the inbound and outbound capacity to increase the overall success rate of payments. However, they all ignore the confinement that one PC just contains two nodes. For the node, like Cafe, who frequently receives money from multiple nodes, massive on-chain transactions are needed to create these PCs and the fee is expensive, which also burdens the blockchain.

In this paper we present a multi-node payment channel (MPC). MPC suits the scenario that one node frequently receives money from multiple nodes, like Cafe, shop, etc. Like the Cafe scenario, the Cafe node can choose to create one MPC and be the only Center of this MPC eternally. The nodes who frequently transact with the Cafe can lock some funds to join the MPC to make off-chain transactions. Compared to using a large number of PCs, MPC avoids adding repetitive information to the blockchain and reduces the fee, meanwhile the blockchain scalability gets improved. MPC reaps these benefits by addressing the challenge below.

This work was supported by the National Key R&D Program of China (No. 2020YFB1005500), the Leading-edge Technology Program of Jiangsu Natural Science Foundation (No. BK20202001), the National Natural Science Foundation of China under(No. 62102168), the Natural Science Foundation of Jiangsu Province under (No. BK20200888), the Natural Science Foundation of the Jiangsu Higher Education Institutions under (No. 20KJB520031), and Postdoctoral Science Foundation of Jiangsu Province under (No.2021K596C).

We noticed the confinement of one PC that just contains two nodes. In the scenario that one node frequently receives money from multiple nodes, a lot of PCs need to be created, leading to the information of channel added to the blockchain massively and repeatedly, which not only costs much fees but also burdens the blockchain. In MPC, we addressed the confinement of the number of nodes, MPC can contain an unlimited number of nodes [14], which greatly reduces the amount of channel information added into the blockchain, meanwhile reduces the fees.

We implement MPC smart contract in Ethereum using the Turing-complete language Solidity [2]. MPC supports the same transaction logic of PC, and well cooperates with existing PC. Our main goal to design MPC is to avoid repetitive information addition to the blockchain, and then scale the blockchain. The main contributions of this paper are summarized as follows.

- We propose a multi-node payment channel (MPC) that suits the scenario that one node frequently receives money from multiple nodes. Compared to using a large number of PCs, MPC avoids adding repetitive information to the blockchain and reduces the fees, which also scales the blockchain.
- We implement MPC in Ethereum using the Turing-complete language Solidity. MPC is a smart contract where multiple nodes make off-chain transactions. MPC supports the same transaction logic of PC, and well cooperates with existing PC.
- We analyze MPC theoretically and multidimensionally to prove MPC's security, economy and high efficiency.
- We test the fee that MPC and PC consumes, experimental results show that the fee of joining MPC reduces 95% compared to creating a PC. When the number of customer nodes $\geq$ 3, MPC outperforms PC.

The remainder of this paper is organized as follows. Section II introduces the preliminaries that MPC relates to, followed by the system model in Section III. In Section IV, we present the MPC design in detail. We analyze MPC theoretically in Section V and present the experimental results in Section VI. Section VII finally concludes this paper.

II. PRELIMINARIES

A. Smart contract

Smart contract can be viewed as a program and ledger running on blockchain, with a unique account address and balance. A node submits the contract code to the blockchain through 1 on-chain transaction to create a smart contract. The Contract will be processed upon predefined code conditions. Nodes can interact with contract and every interaction is 1 on-chain transaction. In each transaction, nodes send coins to contract or return coins from contract to account, but the premise is only meeting the predefined conditions of the contract.

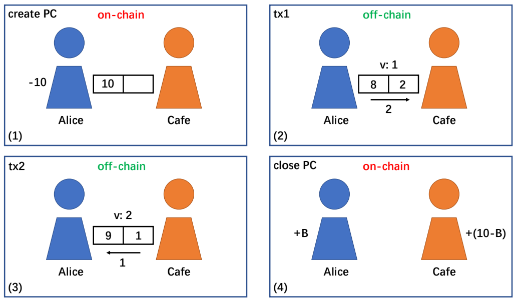


Fig. 1. A PC. (1) and (4) are on-chain transactions and require extra fee, the transactions in PC are off-chain and require no extra fee.

B. PC

PC is implemented by smart contract. When two nodes transact frequently, they lock some funds as IB to create one PC contract, then they can make instant off-chain transactions, without committing every transaction to the blockchain. In Raiden network [8], the validity of balance in PC is determined by the round i, once the balance in round $i+1$ is confirmed, the validity of balance in round i becomes outdated, we adopt this mechanism and name it $version$ in MPC.

In practical life, usually these frequent payments are given by customer node, so we assume only the customer locks funds as IB to create PC. For example, in Fig. 1, Alice locks 10 coins and assigns Cafe as payee node to create a PC (denote this PC with β), which is 1 on-chain transaction. After creation, Alice and Cafe can make off-chain transactions to maintain the 10 coins without committing these transactions to the blockchain. If Alice first needs to give Cafe 2 coins, Alice just gives Cafe a message $\{\beta$,1,8$\}$ (in β, Alice's balance is 8 coins when version is 1) and corresponding signature, then Cafe verifies the correctness of message and signature. If verified correct, this off-chain transaction is accomplished. Similarly, if Cafe needs to give Alice 1 coin, Cafe gives Alice a message $\{\beta$,2,9$\}$ (in β, Alice's balance is 9 coins when version is 2) and corresponding signature. These off-chain transactions can be accomplished instantly and require no extra fee.

Either Alice or Cafe can close the channel to return IB back to account, which usually happens in the fund is insufficient for more transactions or they want to return the funds back. The IB will be transferred according to the biggest version.

C. PCN

Multiple PCs constitute a PCN, which enables two nodes without direct PC can still make off-chain payment in a cross-channel manner. For example, Alice would like to pay Carol 3 coins and find the path $Alice \rightarrow Bob \rightarrow Carol$ directs to Carol. If Bob relays this payment, then the cross-channel payment is accomplished, meanwhile Bob will get an incentive from Alice. A critical challenge here is how to securely transfer funds and prevent Bob from stealing fund. The Hashed TimeLock Contract (HTLC) [7] has been proposed to address the challenge. The basic idea of HTLC is binding the transactions of $Alice \rightarrow Bob$ and $Bob \rightarrow Carol$

into one atomic operation, either they all succeed or they all failed.

D. Motivation

Due to the confinement that one PC just contains two nodes, thus, for the node, like Cafe, who frequently receives money from multiple nodes, a large number of PCs need to be created. As shown in Fig. 2, the information of channel is added to the blockchain massively and repeatedly, meanwhile the cost of creating these PCs is expensive. Although PC had improved the scalability of blockchain, it remains to be improved in situation above.

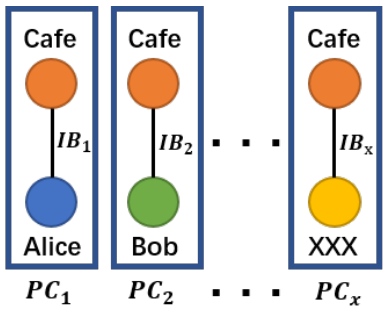


Fig. 2. Every PC contains two nodes (Cafe and one customer), a large number of PCs are created.

III. System model

A. MPC model

Either PC or MPC is a smart contract with a unique account address and balance, where nodes make off-chain transactions. We denote the PC as β and the MPC as $\mathcal{C}$.

There are two types of roles in MPC, Center and node. Center role is the creator of MPC and its number is only 1 eternally, and usually the Center role is the node who frequently receives money from multiple nodes, e.g., Cafe node. Node role is the customer node who frequently transacts with Center and the number of it is unlimited. Every initial balance that node locked in MPC is represented as IB, the balance of one MPC is comprised of all IB in it ($Balance\_MPC = \sum_{i=0}^{n} IB_i$). The information of all nodes is stored in $Nodes$ (a dynamic array) and every $index \in$ Nodes represents a unique node and stores several necessary information of this node, including its IB, address and other information relevant to returning this IB back to account.

The off-chain transaction in MPC is accomplished through message and corresponding Sig (signature). Either Center or node holds their own private key, they exchange messages and Sigs with each other and these Sigs are signed by private key, which guarantees that the Sigs of these messages cannot be forged or denied.

The network structure of MPC is similar to autonomous system (AS) [15] in Internet and the Center is gateway, then the nodes in different MPCs can make off-chain transactions by Centers sending messages and Sigs to each other.

B. Threat model

Since MPC is a smart contract where nodes lock funds to make off-chain transactions, the main malicious behaviors in MPC are cheating or stealing funds. e.g., double-spending or replay attack. Besides, other nodes who are not in MPC may also attempt to steal funds locked in MPC.

C. Goals

We noticed that the node, like Cafe, who frequently receives money from multiple nodes, a large number of PCs need to be created, making massive channel information repeatedly added to the blockchain. By MPC, Cafe and multiple nodes can make off-chain transactions in one space. We focus on achieving the following goals.

Security. The fundamental requirement is guaranteeing every node can safely make off-chain transactions and securely return funds back to account. MPC needs to protect the funds well from malicious behaviors.

Economy. The amount of extra fees in every on-chain transaction heavily depends on the amount of information added to the blockchain. Therefore, it is imperative to minimize the amount of information, i.e., minimize the extra fees.

Efficiency. The payments in PC can be accomplished instantly, MPC needs to similarly satisfy the high efficiency of off-chain transactions.

Compatibility. MPC is a structure different from PC, but MPC needs to support the same transaction logic of PC, then MPC can be compatible well with existing PC.

IV. System design

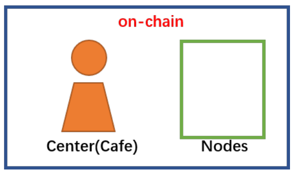
This section presents the formal description of MPC. We elaborate on every behavior according to the order which actually takes place in MPC.

A. Creation of MPC

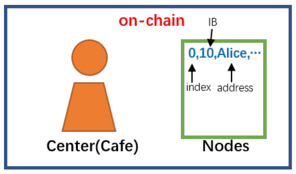
If a node, like Cafe, who frequently receives money from multiple nodes, then the Cafe node can create an MPC smart contract through 1 on-chain transaction. Once the transaction is written into the blockchain, Cafe's MPC (denote this MPC as $\mathcal{C}$) is created successfully. $\mathcal{C}$ will record the creator to be the only Center eternally, thus, Cafe represents Center of $\mathcal{C}$ and vice versa. The newly created MPC $\mathcal{C}$ looks as depicted in Fig. 3(a). The left side is Center (Cafe) and the right side is Nodes (all nodes joining $\mathcal{C}$), the number of nodes in Nodes is 0 initially.

B. Joining MPC

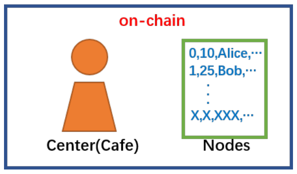
If a node needs frequently transacts with a Center, it can lock some funds as IB through 1 on-chain transaction to create one PC or join MPC. The former manner can be simply seen as adding information of one channel and two nodes to the blockchain while the latter manner can be simply seen as adding information of one node to the blockchain. Obviously, the latter manner sends much less information, i.e., costs much less fees. Therefore, the node chooses to lock some funds to join MPC. Then MPC adds a new index in Nodes and this



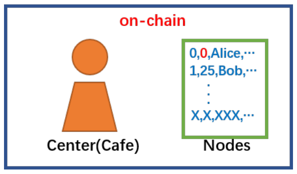
(a) Inception.



(b) Alice joined.



(c) Multiple nodes joined.



(d) Alice (index 0) leaved.

Fig. 3. The status of $\mathcal{C}$, it changes after every interaction.

index stores its IB, address and other information relevant to return this IB back to account. Finally, MPC will return this index to the node. Once the transaction is written into the blockchain, the node joins the MPC successfully.

For example, as shown in **Procedure 1**, Alice locks 10 coins to join $\mathcal{C}$, $\mathcal{C}$ will return Alice an index 0, where stored the 10 coins (IB), Alice's address and other information relevant to return the 10 coins back to account. This process can be obtained from the blockchain so that the Cafe knows index 0 had joined $\mathcal{C}$ with 10 coins, meanwhile the Cafe needs to record the information of index 0 for future off-chain transactions. Since then, index 0 represents Alice and vice versa, Alice (index 0) can make off-chain transactions with Cafe. It should be noted that only Alice (index 0) or Cafe can return the IB of index 0 (10 coins) back to account, $\mathcal{C}$ distinguishes who the node is by storing nodes' addresses.

```
Procedure 1: JoinMPC($Alice, 10coins$)
    $Nodes \leftarrow (Alice, 10, \dots)$;
    return $index$;
```

After Alice's joining, the status of $\mathcal{C}$ is shown in Fig. 3(b). The next node joining $\mathcal{C}$ will be returned index 1, and so on the n-th node joining MPC will be returned index n-1. With more nodes' joining, the status of $\mathcal{C}$ is depicted as Fig. 3(c).

C. *Off-chain transactions*

The main application of MPC is letting multiple nodes make off-chain transactions. Every IB in MPC just belongs to Center and corresponding node, so the relationship between Center and every node is equivalent to a PC. Thus, one MPC can be viewed like Center has PC to every joining node, but actually the information of channel is reused. The network topology structure of one MPC is star network topology, which is shown in Fig. 4. Every node can make off-chain transactions with the Center directly and cannot make off-chain transactions with each other directly, the off-chain transaction logic can be seen in Sect. II.

Generally, the two nodes needing off-chain transaction may not be in the same MPC, meanwhile the nodes in MPC won't transact with only one Center, but joining more MPCs means more on-chain transactions and costs more fees. We assume that every Center will create several PCs with other Centers, then it is easy to satisfy the paths of nodes' cross-channel payments according to the theory of small-world [16]. In practice, Center gets incentive fee from cross-channel payment, so

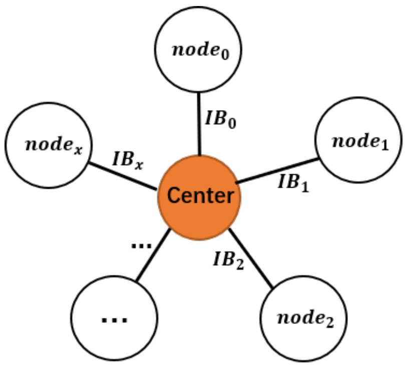


Fig. 4. The network topology of one MPC. Every IB is stored in MPC contract and maintained by Center and corresponding node.

Center is glad to do that. As cross-channel transactions go by, it is inevitable that the funds of one side in PC gradually run out, but the two Centers can lock coins once again to create a new PC before such situation, so Centers always have sufficient funds to relay payments to other Centers.

As shown in Fig. 5. After these PCs created among Centers, nodes can pay to other Centers or the nodes in other MPCs in a manner combined with cross-channel and cross-MPC. For example, Alice needs to pay Dave. We assume Alice is in MPC_1 and Dave is in MPC_2, then path $Alice \rightarrow Center_1 \rightarrow Center_2 \rightarrow Dave$ can accomplish this payment. If Dave did not join any MPC but created several PCs, if there is a path directed to Dave, then path $Alice \rightarrow Center_1 \rightarrow \dots \rightarrow Dave$ can accomplish the payment, MPC cooperates with existing PC well to accomplish payments. The extra fee of this cross-channel payment is much cheaper than the extra fee of 1 on-chain transaction. Whether node to join another MPC depends on the tradeoff between the fee of 1 on-chain transaction and the fees of multiple cross-channel payments.

D. *Returning IB back to account*

Eventually, these IBs locked in MPC will be returned back to account. The critical challenge here is to guarantee the right distribution between Center and node of every IB and the security of other IBs that are still locked in MPC, i.e., malicious nodes cannot steal funds.

In PC, there is only one IB belonging to the two nodes, so it is easy to close PC to return this IB back to account and don't worry about the problem of replay attack. While in MPC, there are a large number of IBs and every IB just belongs to

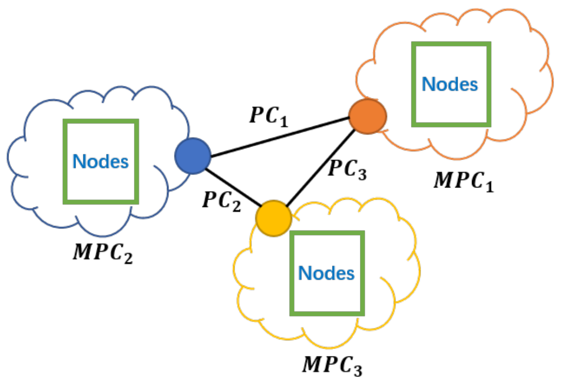


Fig. 5. The topology of MPC&PC network, which is similar to the AS in Internet. The three colorful nodes represent the Center of every MPC respectively and can be viewed as gateways. Centers created PCs with each other, so that the nodes in different MPCs can make off-chain transactions.

the Center and corresponding node. The way to return one IB back to account in MPC is quite similar to closing one PC, but MPC needs two more operations: 1) guaranteeing the transferring fund of one IB is correct, 2) setting IB to 0 after transferring this IB fund. For example, if Center and one node both agree to return the node's IB back to account, as shown in **procedure 2**, MPC will first do a series of checks to guarantee the safety of funds. If checked right, MPC will transfer the IB fund according to their consensus. After transferring, MPC will set the IB of this index to 0 in case of replay attack. An example of returning IB of index 0 is shown in Fig. 3(d), MPC sets the IB of index 0 to 0. If Center or the node wants to return this IB again, MPC will find the transferring fund ($balance$) is abnormal and stop transferring.

Procedure 2: ReturnIB($index_i, balance, Sig$)

```
if caller is not Center or $index_i$ then return;
if $Sig$ is wrong then return;
if $balance > index_i.IB$ or $balance < 0$ then return;
MPC transfers $balance$ to $index_i$;
MPC transfers $index_i.IB - balance$ to Center;
$index_i.IB \leftarrow 0$;
```

V. Analyses of MPC

In this section we present theoretical analyses of MPC through four parts.

A. Security of funds analysis

After node locking fund into MPC, Center can obtain that from the blockchain and record its information. The transaction logic in MPC is similar to PC, so Center can make off-chain transactions with nodes trustingly.

MPC transfers funds only rely on returning IB back to account, meanwhile only Center and the node who locked this IB can do that because MPC can distinguish who the caller is by storing nodes' addresses. Besides, before returning one IB, MPC will do a series of checks. Only checked right, MPC transfers the IB fund. Once one IB is returned back to account, MPC will set this IB to 0 in case of replay attack. If Center or the node wants to return this IB again, MPC will find the transferring fund is abnormal and stop transferring. The security of funds in MPC can be well guaranteed.

B. Fee analysis

PC makes micropayments come true by shifting on-chain transactions to off-chain transactions. But one PC confines to only two nodes, the process of creating one PC can be simply seen as adding information of one channel and two nodes to the blockchain. Thus, a node, like Cafe, who frequently receives money from multiple nodes, a large number of PCs need to be created, the information of channel is added into the blockchain massively and repeatedly. In MPC, the functionality of joining MPC is equivalent to creating a PC, but the information added to the blockchain is just one node, the amount of information is greatly reduced, i.e., the fee of joining MPC is much less than the fee of creating PC.

C. Efficiency analysis

If Center uses a large number of PCs, it needs to store all the addresses of these PCs and nodes, every address in Ethereum is a 42 bytes string (including the "0x"). While in MPC, Center just stores the only MPC's address and every node's index and address, which brings Center manageability.

By distinguishing every node with an index, Center can easily query the information of one node in $O(1)$, the computational overhead to accomplish one off-chain transaction is $O(1)$, the off-chain transactions in MPC can be accomplished as instant as in PC.

D. Decentralization or Centralization

Maybe the MPC will be seen as a degree of centralization due to the existence of Center, but it is inevitable that there will be a center role who frequently receives money from multiple nodes in routine payments, like Cafe. In practical, MPC is not in conflict with decentralization. The process of locking funds into MPC or returning funds locked in MPC back to account is determined by the whole blockchain network without relying on any central trusted third-party. Besides, every IB in MPC is independent and just belongs to Center and the node who locked it. MPC does not require trust between Center and nodes, and the off-chain transactions in MPC also need not to be validated through the central trusted third-party.

The so-called centralization of MPC attributes to the property that the Center node has to receive funds frequently from multiple nodes as a center role. This situation cannot be stoppable even in the main-net of blockchain. Therefore, MPC is not in conflict with decentralization.

VI. Implementation and Evaluation

We implement MPC smart contract in Ethereum using the Turing-complete language Solidity. MPC contains two types of roles: Center and node. The creator of MPC will be its only Center eternally. Any node can join the MPC and the number

TABLE I. The gas fee for executing behavior under PC and MPC.

behavior	on-chain tx	gas	gas payer
Create PC	1	1579137	Customer
Create MPC	1	2970848	Cafe
Join MPC	1	78615	Customer
Create 3 PCs	3	4737411	3 Customers
Create MPC	1	2970848	Cafe
3 nodes join MPC	3	235845	3 Customers
Create n PCs	n	1579137*n	n Customers
Create MPC	1	2970848	Cafe
n nodes join MPC	n	78615*n	n Customers
Tx in PC	0	0	—
Tx in MPC	0	0	—
Return IB in PC	1	48175	Customer/Cafe
Return IB in MPC	1	56359	Customer/Cafe

of it is unlimited. MPC stores the information of the joined nodes in Nodes (a dynamic array) and updates information upon predefined code conditions. MPC supports the same transaction logic of PC and well cooperates with existing PC. Our main goal to design MPC is to reuse the information of channel and reduce the fee of on-chain transaction compared to PC, and then scale the blockchain.

The evaluation criteria of fees for on-chain transactions over Ethereum is an internal currency called *gas*. The amount that gas consumed of 1 on-chain transaction heavily depends on the amount of information added to the blockchain, so it is a challenge to reduce the amount of information as less as possible.

We assume a scenario where multiple customers transact frequently with Cafe. Traditionally, every customer node needs to lock IB and pays gas fee to create PC. While in MPC, Cafe node (Center) pays gas fee to create MPC and every customer node locks IB and pays gas fee to join Cafe's MPC. The process of creating one PC can be simply seen like adding information of one channel and two nodes to the blockchain. The functionality of joining MPC is equivalent to creating one PC, but the information added to blockchain just one node, then the amount of information can be greatly reduced, i.e., the fee of joining MPC is much less than the fee of creating one PC.

The biggest advantage of MPC over PC is that it reuses the channel information and contains unlimited nodes, so we mainly test the gas consumption for the creation of PC and MPC, and test the gas consumption for joining MPC. As shown in Table. 1, the fee of creating an MPC is 2970848 gas, which is paid by Cafe (Center). For customer node, the fee of joining MPC and creating a PC is 78615 gas and 1588774 gas respectively, the fee of joining MPC reduces 95% compared to the fee of creating PC. If there are 3 customer nodes, the total gas fee in MPC is beginning to be less than PC. Because the process of creating PC or joining MPC is independent, we show the gas cost when the number of customer nodes is n. Besides, neither the off-chain transaction in PC or MPC requires fee, i.e., the gas fee is 0. Finally, when returning IB back to account, MPC needs more checks to guarantee the security of other IBs in MPC, so we also test the gas consumption when returning IB under PC and MPC and they are 48175 gas and 56359 gas respectively. Although the cost to return IB in MPC is a little more than PC, the increase amount is negligible because the fee of joining MPC is much less than creating a PC. Obviously, in the scenario that one node frequently receives transactions from multiple nodes, the total fees in MPC is much less than PC.

VII. Conclusion

This paper presents MPC, a smart contract where multiple nodes can securely and efficiently make off-chain transactions. Every off-chain transaction in MPC needs confirmation by the only Center of this MPC. MPC supports the same transaction logic of PC and well cooperates with existing PC. MPC suits the scenario that one node frequently receives money from multiple nodes, like Cafe, shop, etc. In the scenario above, compared to using a large number of PCs, MPC avoids repetitive channel information addition to the blockchain and reduces the fee. We implement MPC in Ethereum using the Turing-complete language Solidity and experimental results show that MPC outperforms the existing PC. As each off-chain transaction in MPC needs to be confirmed by the only Center, in the future, it is necessary to study how to accommodate more Centers to enhance the variety of nodes.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[2] Ethereum. [Online]. Available: 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, pp. 17-30, 2016.
[4] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in 13th USENIX symposium on networked systems design and implementation (NSDI 16), pp. 45-59, 2016.
[5] 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, pp. 51-68, 2017.
[6] Visa. [Online]. Available: https://usa.visa.com/run-your-business/small-business-tools/retail.html/
[7] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016. [Online]. Available: lightning.network/lightning-network-paper.pdf
[8] R. Network. [Online]. Available: 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] Y. Zhang, D. Yang, G. Xue, "Cheapay: An optimal algorithm for fee minimization in blockchain-based payment channel networks," ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019.
[11] S. Dziembowski, S. L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in 2019 IEEE Symposium on Security and Privacy (SP), pp. 106-123, IEEE, 2019.
[12] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), pp. 777-796, 2020.
[13] S. Mercan, E. Erdin, and K. Akkaya, "Improving transaction success rate in cryptocurrency payment channel networks," in Computer Communications, 2020.
[14] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, pp. 1-32, 2014.
[15] Autonomous system. [Online]. Available: https://en.wikipedia.org/wiki/Autonomous_system_(Internet)/
[16] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," nature, 393(6684), pp. 440-442, 1998.