Hindawi
Journal of Mathematics
Volume 2023, Article ID 9826741, 1 page
https://doi.org/10.1155/2023/9826741

Retraction

Retracted: Research on Multiparty Payment Technology Based on Blockchain and Smart Contract Mechanism

Journal of Mathematics

Received 19 December 2023; Accepted 19 December 2023; Published 20 December 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Y. Zhang, "Research on Multiparty Payment Technology Based on Blockchain and Smart Contract Mechanism," *Journal of Mathematics*, vol. 2022, Article ID 3434954, 14 pages, 2022.

Hindawi
Journal of Mathematics
Volume 2022, Article ID 3434954, 14 pages
https://doi.org/10.1155/2022/3434954

Research Article

Research on Multiparty Payment Technology Based on Blockchain and Smart Contract Mechanism

Yanjun Zhang

Party School of Changzhi Committee of the Communist Party of China, Changzhi, Shanxi 046000, China

Correspondence should be addressed to Yanjun Zhang; yanjunzhang@hainanu.edu.cn

Received 12 November 2021; Revised 3 December 2021; Accepted 9 December 2021; Published 21 February 2022

Academic Editor: Naeem Jan

As a peer-to-peer "P2P" distributed ledger, the blockchain has the advantages of decentralization, no trust, open autonomy, and nontampering. Therefore, many users are willing to conduct transactions in blockchain cryptocurrency systems such as Bitcoin and Ethereum. However, the throughput of traditional blockchain is extremely low, and the transaction is so delayed. The payment channel network is the most promising solution to expand the blockchain for widespread use. Achieving secure instant payment on the payment channel can significantly increase transaction throughput and reduce transaction delays. When the payment channel is closed, the balance in the channel will be returned to an account on the blockchain. In this paper, we discuss the design and the implementation of a multiparty payment channel network based on smart contracts. Where a two-party payment channel is designed based on blockchain and smart contracts, a new multiparty payment channel is established on the basis of the payment channel. A detailed definition and description are given, and the creation, update, and closing functions of the multiparty payment channel are designed. Moreover, we design a multiparty payment channel smart contract, deploy it to the local private blockchain, and conduct simulation and testing. The delay time of different transaction methods is counted, and the network topology type, transaction amount, and other factors are studied. The impact of transaction success rate and the gas consumption of different transaction methods are analyzed through multiple sets of experimental statistics.

1. Introduction

With the rapid development of the Internet, the financial industry and third-party payment software have risen rapidly. According to statistics from the Ministry of Commerce in 2015, the national online retail transaction volume was 3.88 trillion yuan, a year-on-year increase of 33.3%. According to the "Report on the Development of China's Internet in 20 Years" released by the China Cyberspace Research Institute in Wuzhen, China's online retail transaction volume has leaped to the world's first place, and the number of mobile phone online shopping users has reached 270 million [1]. People have higher and higher requirements for online banking and payment functions. How to process more transactions in a shorter time while ensuring higher security and how to enable two users to directly trade without the complicated process of intermediate third-party inspection and to obtain a substantial increase in efficiency have become hot topics of discussion in transaction reform [2]. Therefore, the storage of ledger data based on blockchain technology has important development prospects. Traditional online transactions need to rely on third-party payment intermediaries. For users, user transaction information is stored in a centralized database by an intermediary. Users need to trust the intermediary before they can use the payment platform provided by the intermediary. In order to ensure the credibility of transactions, payment companies need to collect a lot of user privacy information to determine the credibility of user accounts. For users, the more the private information is disclosed, the less secure it is. In recent years, private information leakage has occurred from time to time, and its scope includes industries involving assets such as healthcare, law, and real estate. According to the "2015 Data Breach Investigation Report" report, in 2015, there were 79,790 data breaches in 61 countries around the world, of which 2,122 have been confirmed [3]. The data of many

well-known hotels in my country was leaked, including Marriott Hotel and Starwood Hotel. The data leakage of these hotels has resulted in the fact that all the customer order information and personal information that has been consumed in these hotels can be easily obtained by hackers, and it can be seen by people who want to use this information. This kind of leakage of user privacy information is undoubtedly irresponsible for customers' sensitive information, and it also damages the reputation and market value of the hotel. For the transaction intermediary platform, in order to ensure the credibility of the accounts, clearing a large amount of transaction information requires high fees and lengthy time. Nevertheless, under the huge transaction volume, it is impossible to completely avoid mistakes, so it is necessary to pay more expenses and time to make up for these mistakes. This will cause unnecessary waste of funds and time loss for users. On the other hand, the traditional centralized management method [4] is to completely supervise the accounts by banks or third-party payment companies. The transaction process recorded by these intermediary companies is invisible to users, and the transparency and credibility of the ledger data are difficult to guarantee. Therefore, with the emergence of a trust crisis, many researchers want to establish a trustless, distributed privacy protection intermediary platform. Although there are many user privacy protection algorithms, such as differential privacy [5], k-anonymity [6], and t-closeness [7], there are also distributed storage systems, such as Google File System [8] and Big Table [9]. But none of them are distributed and trustless practical applications [10]. Smart contract technology helps to solve the above-mentioned problems faced by individuals and enterprises. The concept of smart contracts was first proposed by cryptographer Nick Szabo in 1994. It aims to establish a decentralized financial authentication system and free manpower from boring document processing. Although the vision of smart contracts is quite beautiful, the concepts and goals are also very clear. But in fact, smart contracts have not received widespread attention until blockchain currencies such as Bitcoin have become popular in recent years. Before the emergence of blockchain technology, the research on smart contracts progressed slowly. This is mainly caused by two reasons. First, it is difficult for computer programs to control actual assets, and there is a lack of an effective fund settlement channel between enterprises. In fact, to this day, cross-border transfers still need to go through multiple intermediaries, which makes it very difficult to automate smart contract procedures. Second, smart contracts are difficult to generate sufficient credibility. After one party breaches the contract, it is difficult for computer programs to work without personnel. In the case of intervention, the breaching party is guaranteed to be punished. The emergence of blockchain technology has changed this status quo. First, the blockchain currency led by Bitcoin is a natural settlement channel for digital assets; second, the openness and nonrepudiation characteristics of the blockchain ledger ensure that the contract execution process can be easily regulated. By establishing the contract in the form of code and recording it in the unalterable blockchain, the authenticity and reliability of the contract can be guaranteed. At the same time, the smart contract system helps enterprises realize digital asset management and lays the foundation for enterprises to realize fully digital office. Blockchain is a decentralized system. According to the blockchain standard, the effect of mutual communication and mutual trust between enterprises can be realized, and the communication cost of enterprises can be effectively reduced. In addition, the unforgeable nature of the blockchain helps to reduce the labor cost of the third-party audit department.

Throughout this paper, we give a background on the related works in Section 2, where we proceed to Section 3 to discuss and analyze the stages of creation of a two-party payment channel, besides the types and methods of the channel, the channel close, and the performance of the two-party payment channel. Then, we introduce the multiparty payment channel, which is designed based on the two-party payment channel. In Section 4, experiments and results were done in order to reduce transaction delay and increase the throughput of blockchain transactions. Finally, in Section 5, we end our study with a conclusion.

2. Related Work

Current blockchain-based smart contract projects include Ethereum [11], Hyperledger [12], Codius [13], and Hawk [14]. Among them, Ethereum is the earliest open-source platform for smart contracts that supports Turing-complete programming languages, and it is also a relatively influential community in the blockchain field. Many companies and enthusiasts have used Ethereum as the underlying platform to implement applications covering market forecasting, supply chain source verification [15], crowdfunding-based financing, securities, and derivatives trading [16]. Hyperledger is an open-source project on advancing blockchain digital technology and transaction verification jointly initiated by the Linux Foundation in 2015 with more than 30 members, including major financial, technology companies, and related open-source organizations. The emergence of Hyperledger officially announced that blockchain technology is no longer just an open-source community technology but has been recognized by mainstream institutions and the market, which is of great significance to the development of blockchain. Some other open-source projects have their own focus. Taking Hawk as an example, the main feature of the project is to protect the data privacy of participating users, and users' transfer records are no longer recorded in the block in a public form. Smart contract is a concept proposed by Ethereum, a set of commitments defined in digital form, including agreements on which contract participants can execute these commitments. The execution process is transparent, traceable, and unchangeable. If Bitcoin is a distributed ledger, its transaction symbol is Bitcoin. Ethereum is a distributed platform without transaction symbols. It adds a core smart contract concept on the basis of Bitcoin [17]. The original Bitcoin mechanism was not scalable. It was an application that could only run Bitcoin

transactions. If a new developer wants to develop new applications on the basis of the blockchain, the blockchain needs to be completely redone. Ethereum is a platform provided to application vendors. Developers only need to pay a certain amount of Ether, build different contracts on this platform, and run different operations and different businesses. It no longer just records transactions but adds more parameter information to the contract. By establishing its own blockchain, a clear "state tree" representing the current balance of each address and a "transaction table" representing the transactions between the current block and the previous block are stored in each block [18, 19]. Ethereum solves the scalability problem. The business scope supported by Ethereum includes voting, financial exchanges, intellectual property management, crowdfunding, and artificial intelligence [20]. Ethereum provides a platform that enables storage and Turing's complete scripting language to make it possible to encode a complete currency in a single contract. Compared with Bitcoin without smart contracts, Ethereum with smart contract features has a broader future development direction, which can support more applications and larger transaction volumes. Coupled with its own high-efficiency characteristics, it will gain a longer-term advantage. But Ethereum also has obvious shortcomings; that is, the code of the application itself and the code generated by the application are in the same block, and the increase in the amount of accumulated data for a long time will cause the expansion of the block. At the same time, in the process of using contracts to form applications, multiple applications and contracts may be controlled by the same institution, which in turn will cause the same economic phenomenon as existing financial institutions. The entire system of Ethereum is controlled by the state or an institution. In this way, the freedom of blockchain transactions no longer exists. The blockchain can support many businesses more easily and conveniently, and it is no longer necessary to design all the blockchain storage by yourself. In order to support more applications, the Ethereum contract provides the simplest possible contract form so that any programmer can write a program that can run on Ethereum. Smart contracts will be allowed to store data in persistent memory. Projects such as Bitcoin and Ethereum are in the form of public chains. The characteristics of the public chain are no confidentiality, no traceability, long confirmation time, no finality, and low throughput, which cannot meet the needs of commercial use, so a blockchain in the form of a private chain appears. Hyperledger proposes to support blockchain based on private chains and alliance chains. It is a new commercial blockchain application that pays more attention to privacy and confidentiality than blockchain projects in the form of public chains. Hyperledger is an enterprise-level distributed ledger technology based on blockchain. It is used to build commercial application platforms for various industries. It has modularity and high performance and reliability and provides business-friendly licenses [21]. Compared with Bitcoin's digital currency system and Ethereum's general public chain platform, Hyperledger provides a general alliance chain platform, which is mainly managed by the Linux Foundation. It uses public or private networks, which can provide developers with privacy, also inherit the concept of Ethereum smart contract, and support the use of Go, Java, and other development languages to write smart contracts. In Hyperledger, this kind of smart contract is called on-chain code to distinguish between the contract on the blockchain and the written contract, which allows enterprises to use consensus mechanisms and services in a plug-and-play manner. Since the launch of the Hyperledger project, new companies have continued to join the Hyperledger camp [22], including Samsung's IT service subsidiary, Chinese heavy machinery manufacturer Sany Heavy Industries, Huawei, and other large enterprises. The fabric subproject, which has made a huge contribution to the Hyperledger, provides basic privacy, confidentiality, and auditability and uses the Practical Byzantine Fault Tolerance consensus algorithm [23] to provide confidentiality, authority management, and control functions for transactions. At the same time, the functions of consensus mechanism and bookkeeping are separated, in which nodes can be dynamically expanded and contracted, with upgradeable smart contracts, which are expected to increase throughput. And another project Sawtooth Lake, which was originally a blockchain platform mainly contributed and led by Intel, supports a new consensus mechanism Proof of Elapsed Time. In terms of blockchain payment channel network, there are some studies in academia and industry, among which the two projects of Lightning Network [24] (the Lightning Network and Raiden Network [25]) are the most famous. The Lightning Network uses the idea of a payment channel to transfer transactions originally on the Bitcoin blockchain to the off-chain, that is, the Lightning Network, to complete fast instant transactions and return the balance in the channel to the block when the payment channel is closed. The Lightning Network uses the revocable sequence maturity contract RSMC to complete two-way payments while ensuring the security of the balances of both users. In order to realize the transaction between two users without a direct channel, the Lightning Network has designed a hash-based time lock contract HTLC to implement routing on the payment channel network. The Raiden Network uses the same solution as the Lightning Network, but it is a payment channel network for the Ethereum blockchain. Both of these projects are currently under development, and some users have already joined the Lightning Network and Raiden Network, but their availability and security have yet to be verified. To sum up, the existing research on payment channels or state channels mainly focuses on the design of channels and networks, channel routing, rebalancing, and so on; these researches are all in the initial stage, and most of them are based on theoretical research. Therefore, research in the direction of the blockchain payment channel network is very necessary, which will help solve the scalability problem of the blockchain itself and turn the blockchain into an electronic currency system suitable for large-scale applications. In this paper, we mainly study multiparty payment channels based on blockchain and smart contracts and conduct simulation experiments to study the impact of different factors on payment channels.

3. Method

In this section, we will discuss and analyze the stages of the creation of a two-party payment channel, the updates, types (off-chain update; on-chain update), and methods, besides channel closure, as well as the performance of the two-party payment channel. Then, we will introduce the multiparty payment channel, which is designed based on the two-party payment channel.

3.1. Payment Channel Design

3.1.1. Channel Creation Stage. In the channel creation phase, one user sends a request to create a two-party payment channel to another user and attaches his own signature. If the other party agrees to create, the message will be returned with a signature. After the signatures of both parties are collected, a consensus to create a channel is reached, and the requester initiates a request to create a channel to the smart contract. The smart contract will verify the signatures of the two users and check some necessary conditions (e.g., the accounts of both users on the blockchain must have sufficient balance to establish a channel, and a certain fee can be paid). Assuming that the payment channel to be created for both parties is t and the user sending the request is P, then the other user $P_2 = t.\text{counterparty}(P_1)$. When requesting to create a channel, the message sent by P_1 is

$$\text{tpc} - \text{creat} = ''\text{creat TPC}t, \delta_1''. \tag{1}$$

This means that P_1 requests to establish a new two-party payment channel t with P_2, where t represents an instance of the channel and δ_1 is the signature of P_1. If P_2 agrees to create the channel after verifying the signature, it will return a message of the following form:

$$\text{tpc} - \text{creat} - \text{agree} = ''\text{creat TPC}\, t, \delta_2'', \tag{2}$$

where δ_2 is the signature of P_2 on the message. If P_2 does not agree to create the channel, it returns a rejection message:

$$\text{tpc} - \text{creat} - \text{reject} = ''\text{reject to creat TPC}\, t''. \tag{3}$$

If the two parties reach a consensus to create a channel, that is, P1 receives the message and signature agreed by P_2, then P_1 can call the create TPC function provided in the smart contract through RPC and other methods. The smart contract verifies the signatures of both parties and completes the mortgage of funds and the creation of channels. At this time, the deposit required to establish the channel will be deducted from the blockchain accounts of P and P_2 through $L.\text{remove}(P1, t.b_1^0)$ and $L.remove(P_2, t.b_2^0)$ and add an instance of the payment channel of both parties to the storage space of the channel through $C.\text{addTPC}(t)$. It should be noted that the stage when users reach a consensus to create a channel is a process of interaction under the blockchain. Therefore, the delay is extremely low, and the interaction in a more common network is only affected by the network status. When calling a smart contract to create a channel, both users need to mortgage the assets on their respective blockchain accounts into the smart contract. As a result, transactions on the blockchain will occur, leading to a long delay in the process. However, in the entire life cycle of the payment channel between the two parties, the process of creating the channel only needs to be executed once, so a long delay here is acceptable.

3.1.2. Channel Update Stage. There are two types of channel updates: off-chain updates and on-chain updates. The procedures of the two update methods will be described in detail as follows:

(1) Off-chain update is the key to both parties' payment channels to reduce transaction delay and increase transaction throughput. Because the process is carried out under the blockchain and no blockchain transaction is generated, there is no need to wait for the transaction to be packaged and confirmed by the block. Compared with the blockchain transaction, a lot of transaction time is saved. In the off-chain update phase of the channel, one user sends a request to update the payment channel of both parties to another user and attaches its own signature. If the other party agrees to update, it will return the message and attach the signature. After the signatures of both parties are collected, a consensus to update the channel is reached. Two users can update the local channel to a new state and initiate a settlement request to the smart contract when settlement is needed. Suppose the payment channel of both parties to be updated is t, and the user who sends the request is P_1. When requesting to update the channel t, the message sent by P is

$$\text{tpc} - \text{update} = ''\text{update TPC}\, t \,\text{to t}', \delta_1''. \tag{4}$$

This means that P_1 requests to update the two-party payment channel t between $P_2 = t.\text{counterparty}(P_1)$, where t' represents a new channel state and δ_1 is the signature of P_1. This message needs to satisfy $t.b_1^0 + t.b_2^0 = t'.b_1^{now} + t'.b_2^{now}$, because the total balance on a payment channel between two parties will not change. In addition, $t\prime.version > t.version$ should also be satisfied because the version number is an increasing integer if $t'.\text{version} < \text{tversion}$; then, it shows that the requester sent an old version of the channel status, which is a malicious fraud. If P_2 verifies the content and signature of the message and agrees to create the channel, it will return a message as follows:

$$\text{tpc} - \text{update} - \text{agree} = ''\text{update TPC}\, t \,\text{to}\, t', \delta_2'', \tag{5}$$

where δ_2 is the signature of P_2 on the message. If P_2 does not agree to create the channel, it returns a rejection message:

$$\text{tpc} - \text{update} - \text{reject} = ''\text{reject to update TPC}\, t''. \tag{6}$$

(2) The on-chain update is to call the smart contract interface to update the latest state of the channel to the blockchain. If the two parties reach a consensus to update the channel, that is, P_1 receives the message and signature agreed by P_2, then P_1 can call the updateTPC function provided in the smart contract through RPC and other methods. The smart contract verifies the signatures of both parties and checks some necessary conditions, such as whether the payment channel exists, whether the balance distribution is reasonable, and whether the channel version is correct, to verify the legitimacy of the updated content. If the verification is successful, you can update t to t' through C.update TPC (t, t'). Generally speaking, the update operation of the smart contract only needs to be performed once before the channel is closed. The remaining multiple updates can only be updated off-chain between the two parties of the channel. After the two parties reach an agreement, the latest channel version can be kept locally.

3.1.3. Channel Closing Phase. In the channel closing phase, one user sends a request to close the payment channel of both parties to another user and attaches his own signature. If the other party agrees to create it, the message will be returned and the signature attached. After collecting the signatures of both parties, a consensus is reached to close the channel. The requester initiates a request to close the channel to the smart contract. The smart contract will verify the signatures of the two users and check some necessary conditions, such as the version number and balance of the channel and distribution. Assume that the two payment channels to be closed are t, and the user sending the request is P_1. When requesting to close the channel t, the message sent by P_1 is

$$\text{tpc} - \text{close} = "\text{close TPC}\, t, \delta_2''. \tag{7}$$

This means that P_1 requests to close the two-party payment channel t between $P_2 = t$.counterparty (P_1), and δ_1 is the signature of P_1. If P_2 verifies the content and signature of the message and agrees to close the channel, it will return a message as follows:

$$\text{tpc} - \text{close} - \text{agree} = ''\text{close TPC}\, t, \delta_2'', \tag{8}$$

where δ_2 is the signature of P_2 on the message. If P_2 does not agree to create the channel, it returns a rejection message:

$$\text{tpc} - \text{close} - \text{reject} = ''\text{reject to close TPC}\, t''. \tag{9}$$

If the two parties reach a consensus to close the channel, that is, P_1 receives the message and signature that P_2 agrees to close, then P_1 can call the closeTPC function provided in the smart contract through RPC and so on, and the smart contract verifies the signatures of both parties and completes the refund of the deposit and the channel. At this time, by calling L.add $(P_1, t.b_1^{now})$ and L.add $(P_2, t.b_2^{now})$, the balance in the channel will be returned to the blockchain accounts of P_1 and P_2 according to the latest balance distribution status, and the instance of the payment channel of both parties in the storage space is deleted through C.removeTPC (t). The refund operation involves a blockchain account, which is an on-chain transaction, so a certain handling fee is required when refunding the deposit.

3.2. Performance Analysis of the Payment Channel between the Two Parties. For a two-party payment channel, the main performance indicator is transaction delay, and the transaction delay on the payment channel is mainly affected by the number of off-chain interactions between the two parties of the channel, the number of transactions on the chain, and the number of signatures. Next, in this section, we will analyze the performance of the above-mentioned two-party payment channels and analyze the four stages of channel creation, off-chain update channel, on-chain update channel, and channel closure. The three stages of creation, on-chain update, and closing of the channel need to interact with the smart contract. In addition to completing these functions, the smart contract needs to check the legitimacy of the caller each time, for example, whether the channel exists, whether the balance distribution is correct, whether the signatures of both users are correct, and whether the channel version is correct; these operations ensure the security of user assets. In addition, because these three stages need to call the interface of the smart contract and change the channel state in the contract, there will be transactions on the blockchain, so the delay is longer. There is no need to interact with the smart contract when the channel is updated under the chain. It only needs to reach a consensus to update the channel state between the two users. Both users keep a copy of the same latest channel state and have the other's signature on the channel state, so both parties can initiate an on-chain update request to the smart contract at any time. In each stage, the main factors that affect performance are the number of on-chain transactions, the number of off-chain interactive messages, and the number of signatures on the payment channel of both parties. When creating a two-party payment channel, the two users at both ends of the channel need to interact to reach a consensus on creating the channel. The process includes a TPC-create request message and a TPC-create-agree return message, including two signatures in total. After reaching a consensus to create a channel, two users pledge a certain amount to the smart contract to create the channel, so two on-chain transactions will occur. When updating the channel state off-chain, there is no need to call the smart contract interface, and there is no time-consuming on-chain transaction. Two users request updates through the TPC-update message and agree to update the channel through the TPC-update-agree message. This process includes two interactive messages and two signatures, and the off-chain update is the advantage of the payment channel. When the channel is updated on the chain, that is, the latest state of the payment channel is updated to the smart contract, the update interface of the smart contract needs to be called once, so there will be an on-chain transaction. Similarly, when two users interact, two

messages and two signatures are generated. When closing the payment channel of both parties, the balance needs to be returned to the blockchain accounts of users at both ends of the channel according to the latest channel status in the contract, so two on-chain transactions will occur. In the same way, reaching a consensus to close the channel requires an interaction, including a request for closing the channel and a reply to two messages, which contain the signatures of the two users.

3.3. Multiparty Payment Channel Design. Similar to a two-party payment channel, the life cycle of a multiparty payment channel mainly includes three stages: creation, update, and closure. This section will combine the model and definition to introduce in detail the specific content of each stage in the life cycle of the multiparty payment channel. In addition, in the following description, we will follow the definitions of related symbols and functions of the blockchain in the first two sections.

The multiparty payment channel in this paper is designed based on the two-party payment channel. The smart contract opens the interface to the user. The user calls the interface provided by the smart contract through RPC and other methods to complete the creation, update, and close of the multiparty payment channel. This paper designs three basic smart contract functions, and the specific content is as follows:

$$\text{createMPC}(\text{parties}, \text{tpcs}, \delta) \longrightarrow \{i\ d, 0\}. \tag{10}$$

This function is used to request the creation of a new multiparty payment channel from the smart contract. parties is a collection of blockchain user addresses, representing all users who will join the multiparty payment channel. tpcs is a collection of the identifiers of a two-party payment channel, which means all the two-party payment channels that are added to the multiparty payment channel. Once a multiparty payment channel is added, the busy field of the payment channel of both parties will be set to 1, indicating that the channel has been added to a multiparty payment channel and cannot be used separately. δ represents the collection of signatures of all users, and the smart contract will verify these signatures and some necessary conditions. After the verification is successful, a new multiparty payment channel is created, and the identifier id of the channel is returned to the user. If the creation fails, it returns 0.

$$\text{updateMPC}(\text{id}, \text{mpc}', \delta) \longrightarrow \{1, 0\}. \tag{11}$$

This function is used to request the smart contract to update the multiparty payment channel whose identifier is id. The input parameter mpc′ represents the new state of the multiparty payment channel, including the new balance distribution and new version number of the two parties' payment channels. After the smart contract is successfully verified, the status of the multiparty payment channel is updated according to mpc′. If the update is successful, it returns 1; otherwise, it returns 0.

$$\text{closeMPC}(\text{id}, \delta) \longrightarrow \{1, 0\}. \tag{12}$$

After the channel parties reach a consensus, they request the smart contract to close the multiparty payment channel with the identifier id. After the smart contract completes the verification step, it will release all the two-party channels in the channel, that is, set the busy value of all the two-party payment channels in the multiparty payment channel to 0, which means that the channel is idle and can be used as a separate two-party payment channel or join other multiparty payment channels. The smart contract will delete the instance of the multiparty payment channel in the storage space and return 1 to the user if the closure is successful; otherwise, it returns 0.

3.3.1. Channel Creation Phase. In the creation phase of a multiparty payment channel, a user sends a request message to the server to create a multiparty payment channel with multiple other users and attaches its own signature, and the server notifies other users. Assuming that the user Pi initiates a request, the request message can be defined as

$$\text{mpc} - \text{create} = ''\text{create an MPC}\, m, \delta_i''. \tag{13}$$

In the message mpc – create, δ_i is the signature of the requester P_i for the message, and m represents an instance of a multiparty payment channel. For example, *m.parties* represents the addresses of all users participating in the channel, and *m.*tpcs represents that the multiparty payment channel needs to be used, the payment channel for both parties. The server will forward the request to all other users in *m.*parties. For each user P_j who receives the request, if he agrees to create the channel, it will return a consent message to the server:

$$\text{mpc} - \text{create} - \text{agree} = ''\text{create an MPC}\, m, \delta_j''. \tag{14}$$

Otherwise, the user P_j will return a rejection message:

$$\text{mpc} - \text{create} - \text{reject} = ''\text{reject to create}\, m''. \tag{15}$$

If the consent of all users is obtained, that is, the server has collected the signatures of all users for the creation request message, it can call the interface for creating multiparty payment channels provided by the smart contract.

The createMPC(*m.*parties, *m.*tpcs, δ) function initiates a request to the smart contract, where δ is the set of signatures of all users. The smart contract will perform a series of verifications. If the verification is passed, a new multiparty payment channel instance will be created, the busy field of all two-party payment channels in *m.*tpcs will be set to 1, and the identifier of the channel will be returned to the user.

3.3.2. Channel Update Stage. The update phase of the multiparty payment channel, that is, the multiparty transaction phase, also needs to reach a consensus on channel update among all users. Due to the complexity of the multiparty payment channel update stage, the following content will be divided into four parts: multiparty

transaction generation, channel selection, multiparty transaction optimization, and multiparty payment channel update.

(1) Multiparty transaction generation: The multiparty payment channel in this paper is built on the payment channel layer above the blockchain network layer, and its purpose is to provide services to the upper application layer. Therefore, multiparty transactions are generated in the application layer. For example, in a multiparty game where four people participate in $\{P_1, P_2, P_3, P_4\}$, the rules of the game are that each round has a winner, and other users pay 1 token to the winner. Then, every time a round of the game needs to be settled, a multiparty transaction will be generated at this time. For example, if P_3 is the winner of this round, a multiparty transaction mptx = $\{(P_1, P_3, 1), (P_2, P_3, 1), (P_4, P_3, 1)\}$, meaning that each of P_1, P_2, and P_4 pays one token to P_3. Obviously, a multiparty transaction contains multiple two-party transactions. If $mptx$ is split into three separate two-party transactions, theoretically, the same settlement as a multiparty transaction can be completed, but the atomicity of the multiparty transaction cannot be guaranteed. For example, if the two parties' transactions $(P_1, P_3, 1)$ and $(P_2, P_3, 1)$ in $mptx$ are successfully settled and the transaction $(P_4, P_3, 1)$ cannot be successfully settled due to insufficient channel balance or routing failure, then the multiparty transaction mptx only partially settled and its atomicity is destroyed. Such a transaction is obviously unreasonable, and the wrong settlement result is difficult to roll back. Therefore, multiparty transactions need to ensure atomicity; that is, all two-party transactions included in a multiparty transaction either all settle successfully or all settlement fails.

(2) Channel selection: The function of channel selection is similar to routing in order to select the appropriate channel to complete multiparty transactions. The following will illustrate the importance of channel selection through an example.

As shown in Figure 1, it is a multiparty payment channel established by five users. The numbers at both ends of the arrow indicate the balances of the users at both ends of the channel. It can be seen that, in the multiparty payment channels, some channels are redundant, and not all two-party payment channels need to be updated. For example, when P_1 needs to pay a token to P_3, the payment can be made directly through the channel $P_1 \leftrightarrow P_3$, or the transaction can be made directly through the path $P_1 \leftrightarrow P_5 \leftrightarrow P_3$. Assuming that after a round of the game, P_1 pays a token to P_3 through $P_1 \leftrightarrow P_3$, the balance of the channel becomes $[P_1: 0, P_3: 4]$. At this time, the channel is an extremely unbalanced channel because P is no longer possible to pay to the other party through the

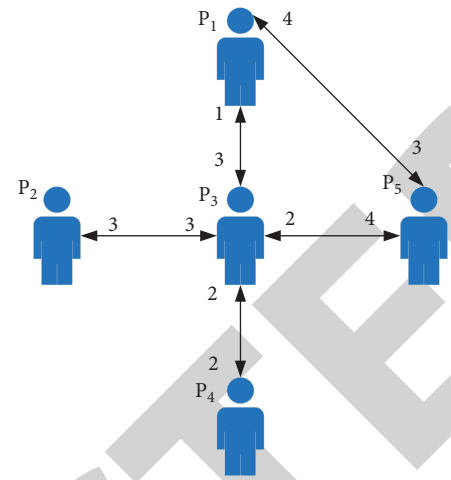

Figure 1: Schematic diagram of channel selection.

channel $P_1 \leftrightarrow P_3$. In the next round of settlement, if P1 needs to pay to P3, the channel $P_1 \leftrightarrow P_5 \leftrightarrow P_3$ will be selected for the transaction. Therefore, since each multiparty transaction will cause the balance status of multiple payment channels in the topology to change, we need to select the appropriate channel to complete a transaction. In order to measure the quality of a channel, this paper defines the following concepts:

$$\mathrm{Dist}_{ij} = \frac{b_{ij+}b_{ji}}{b_{ij} + b_{ji} + \sum_{k \in N_{ij}} \left(b_{ik} + b_{ki} + b_{jk} + b_{kj}\right)}, \tag{16}$$

$$\mathrm{Bal}_{ij} = \frac{\left|b_{ij} + b_{ji}\right|}{b_{ij} + b_{ji}}, \tag{17}$$

$$W_{ij} = k_1 * \left(1 - \mathrm{Dist}_{ij}\right) + k_2 * \mathrm{Bal}_{ij}. \tag{18}$$

In formula (16), Dist_{ij} represents the local proportion of the total channel balance between i and j, where b_{ij} represents the amount i can pay to j, b_{ji} represents the amount j can pay to i, $b_{ij+}b_{ji}$ is the total balance of channel ij, N_{ij} represents the set of neighbor nodes of channel ij, and a neighbor node of channel ij refers to the node that has established both the payment channel with i and the payment channel with j. Dist_{ij} can indicate the local importance of the balance of the channel. The larger the value, the higher the proportion of the balance of the channel ij. The larger the value is, the more the channel should be selected. In formula (17), Bal_{ij} represents the balance of the channel subscription, the numerator $|b_{ij} + b_{ji}|$ represents the absolute value of the difference between the balances at both ends of the channel, and the denominator $b_{ij} + b_{ji}$ represents the total balance of the channel. The smaller the value, the more the balance at both ends. The smaller the gap, the more balanced the channel. A balanced channel can support two-way transactions. Formula (18) is a value obtained by combining Dist_{ij} and Bal_{ij}, which is used to express the weight of channel ij. k_1 and k_2 are two adjustable parameters. The smaller the W_{ij} value, the better the quality of the channel.

3.3.3. Channel Closing Phase. In the closing phase of the multiparty payment channel, a user P_i initiates a request to close the channel to the server. This message can be recorded as

$$\text{mpc} - \text{close} = ''\text{close MPC mpc}, \delta_i''. \tag{19}$$

After receiving the request, the server will notify other users in the channel, namely, mpc.otherparties (P_i). If other users P agree to close, they will return a message:

$$\text{mpc} - \text{close} - \text{agree} = ''\text{close MPC mpc}, \delta_j''. \tag{20}$$

Otherwise, it will return a message that refuses to close:

$$\text{mpc} - \text{close} - \text{reject} = ''\text{reject to close MPC mpc}''. \tag{21}$$

After a certain period of time, after the server collects all users' replies, if all users agree to close MPC, the instance of the multiparty payment channel will be deleted, and the busy value of all the two-party payment channels involved will be set to 0, indicating that the payment channel of both parties has withdrawn from the environment of multiparty payment channels and can be used as an independent two-party payment channel or join other multiparty payment channels. After closing the channel, an mpc − closed message is used to notify all users that the channel has been closed. If it is not successfully closed, an mpc − close − failed message is returned.

3.4. Multiparty Payment Channel Performance Analysis. This section will analyze the performance of the multiparty payment channel according to the various stages of its life cycle. If n users participate in the multiparty payment channel, its performance is shown in Table 1. In the channel creation phase, 1 user initiates an application, the server sends a request to other n-1 users, n-1 users will return a response message to the request, and the server will return m messages to n users, notifying them of the result of creating a multiparty payment channel, so there are a total of $3n - 1$ messages in the interactive process at this stage. The smart contract is called to generate a transaction on the chain at a time, and each user needs to provide its signature, so there is a total of m signatures.

It should be noted that the multiparty payment channel proposed in this paper can theoretically support any number of users to establish a multiparty payment channel, but it may be limited in actual implementation. For example, in Ethereum, each block has a maximum gas limit, called a gas limit. The more the users in the channel, the more the signatures the smart contract needs to verify and therefore the more the gas it consumes. Only when the gas value consumed to create a multiparty payment channel is less than the limit value, can the multiparty payment channel be successfully created. In other blockchains without a gas limit, the number of users is not limited.

From the above analysis, it can be seen that, on the multiparty payment channel designed in this paper, except for the on-chain transactions necessary to create, update, and close the multiparty payment channel, the rest of the interaction process can be performed off-chain, so instant multiparty transactions can be realized. Compared with direct transactions through Ethereum, it can greatly reduce transaction delays, increase transaction throughput, and ensure the atomicity of multiparty transactions.

4. Experiments and Results

In this section, our aim is to view experiments and results done in order to mainly reduce transaction delay and increase the throughput of blockchain transactions, where we study the influence of factors, transaction amount, transaction distribution, and gas consumption.

4.1. Experimental Configuration. In this experiment, we mainly study three indicators of transaction delay, transaction success rate, and smart contract gas consumption. In the blockchain payment channel network, transaction delay is a key indicator. The purpose of payment channels is to significantly reduce transaction delays and increase the throughput of blockchain transactions. The transaction success rate is also an important indicator, which is related to the availability of the payment channel network. Higher transaction success rate means that the channel can withstand more transactions, and there will be no imbalance in a short period of time that will cause the channel to be unavailable. In this experiment, we studied the influence of factors such as network topology, transaction amount, and transaction distribution on the success rate of transactions. In addition, the gas consumption of smart contracts based on Ethereum is also an important indicator. In Ethereum, transaction fees are calculated according to the amount of gas consumed by the contract. Therefore, this paper also sets up several sets of experiments to study the gas consumption of different transaction methods. In order to verify the feasibility of the multiparty payment channel proposed in this paper in different networks, this paper uses random topologies of different scales to simulate the payment channel network. These random topologies are generated using NetworkX's algorithm. In addition, this experiment also intercepted network topologies of different scales from the real topology of the Lightning Network to conduct experiments.

4.2. Transaction Delay. Transaction delay is an important performance indicator for blockchain transactions and an important factor that limits the throughput of blockchain transactions. In Ethereum, the average block generation time is about 15 seconds, and the transaction throughput is about 15 TPS. Through the payment channel, we can transfer transactions on the blockchain to off-chain to achieve low-latency, high-throughput transactions. We set the block generation time of the simulated blockchain to 15 seconds, which is consistent with the real Ethereum block generation time. We separately studied the transaction delays of direct transactions through Ethereum, two-party payment channel transactions, and multiparty payment channel transactions.

Table 1: The performance of each stage of the multiparty payment channel of n users.

Function	Number of on-chain transactions	Number of off-chain transactions	Number of signatures
Created channel	1	$3n-1$	n
Off-chain update channel	0	$5n+1$	Relevant to the actual
On-chain update channel	1	0	n
Closed channel	1	$3n-1$	n

Figure 2 shows the delay of direct transactions via Ethereum varies with the number of transactions. This is achieved by calling a specific interface. This interface can specify the account addresses of the payer and the payee and the transaction amount to initiate an Ethereum transaction. It can be seen from the figure that the more Ethereum transactions, the greater the transaction delay. Because each transaction needs to call the interface once, the average delay of each transaction is about 15 seconds, which is consistent with the block time.

Figure 3 shows the change of the delay of the payment channel transaction between the two parties with the number of transactions. In the simulation experiment, two users are set up, the payment channel between the two parties is established based on the smart contract of the two parties, and the off-chain interaction is completed through socket communication. The communication delay of the network is set to 50 ms. By calling the interface opened by the smart contract to complete the creation, update, and closing of the channel, it can be seen from the figure that the entire transaction process consists of four parts: creating a channel, off-chain transaction, updating the channel on the chain, and closing the channel. Among them, the creation, chain update, and closing of the channel need to call the smart contract interface to generate Ethereum's on-chain transaction, so the delay is longer. The delay in the off-chain transaction part is relatively low because off-chain transactions do not need to call the smart contract interface and will not generate on-chain transactions. Therefore, as the number of transactions increases, the total transaction delay will not increase significantly because once the channel is established, multiple off-chain transactions can be performed, which is extremely low compared to on-chain transactions.

Figure 4 shows the delay of multiparty payment channel transactions with the number of transactions. In this experiment, three users are set up. A multiparty payment channel is established based on a multiparty payment channel smart contract. The communication delay of the network is also set to 50 ms, and users complete off-chain transactions through socket communication. It should be noted that, as the number of users increases, the delay of multiparty payment channel transactions may increase because the number of off-chain interactions will increase.

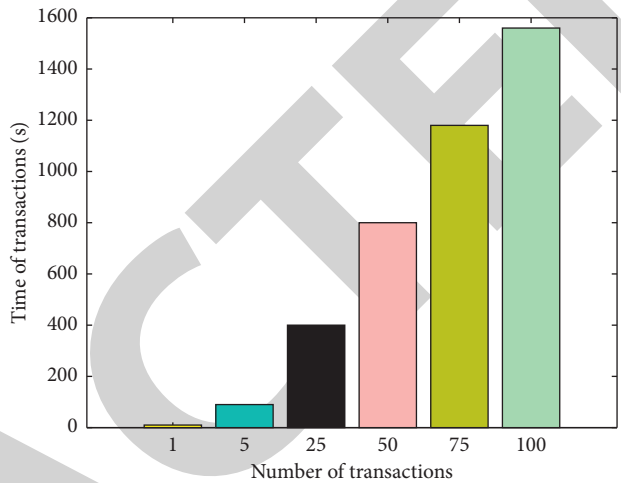

Figure 2: The relationship between the number of transactions and time in Ethereum.

4.3. Transaction Success Rate

4.3.1. The Impact of Network Topology on the Success Rate of Transactions. In the research on the payment channel network, some scholars mentioned that the hub-and-spoke topology is used to establish a multiparty payment channel; that is, all users are connected to the same payment hub, and each user conducts transactions with other users through the hub. In order to study the impact of randomized topology and hub-and-spoke topology on the success rate of transactions, this paper sets up several sets of experiments to run on different networks while ensuring that the number of nodes in the network is the same as the total amount of channels. For the same batch of transactions, statistics on their transaction success rate indicators and the experimental results are shown in Figures 5–7. In this experiment, we set up three different scale pair topologies, consisting of 10 nodes, 50 nodes, and 100 nodes, respectively, and ran multiple sets of transactions with different numbers to study the changes in the success rate of transactions, where "ran" represents a randomly generated topology, and "hub" represents a hub-and-spoke topology. It can be seen from the results in the figure that, in networks of different sizes, the transaction success rate on random topology is higher than that on hub-and-spoke topology. The reason is that, in the hub-and-spoke network, each user only establishes a two-party payment channel with the hub node. With more and more transactions, the channel will be exhausted. In a random network topology, a user's payment channels are usually established with multiple nodes, so when one channel is exhausted, other channels can be selected for transactions. In addition, it can also be seen that, as the number of transactions increases, the success rate of transactions in both networks will gradually decrease.

In addition to the network topology, the number of nodes in the network will also have a certain impact on the transaction success rate. Therefore, this paper sets up several sets of experiments to study the performance of transaction

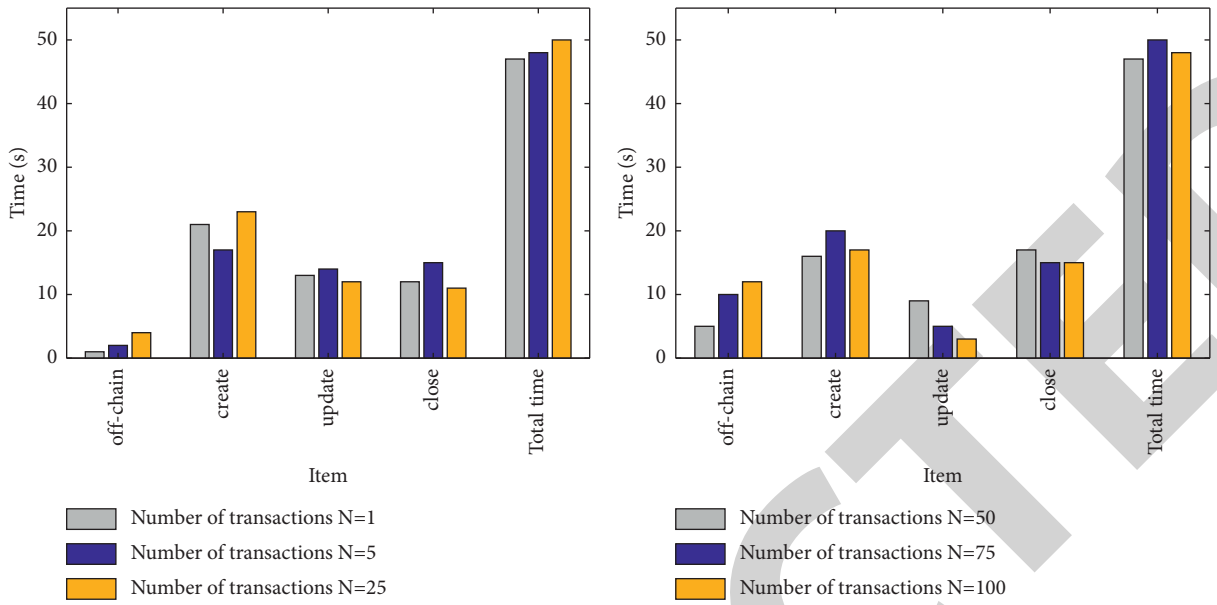

Figure 3: Comparison of dual payment channel transactions.

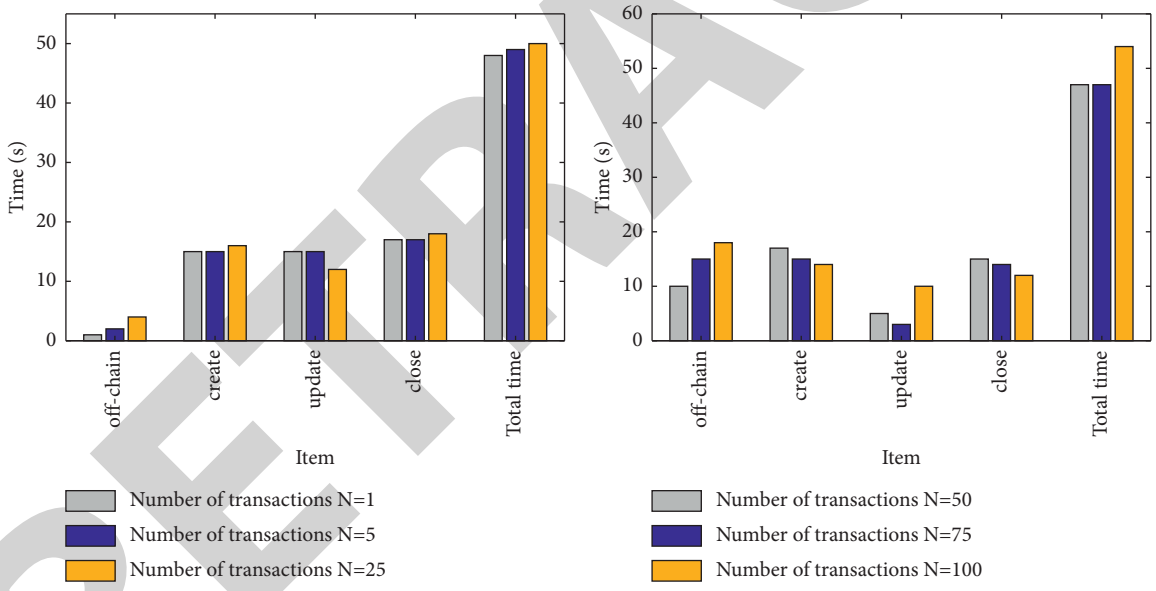

Figure 4: Comparison of multipayment channel transactions.

success rates in networks with different numbers of nodes. The experimental results are shown in Tables 2–4. Table 2 shows the performance when the quota of each transaction is less than 5 ethers. Table 3 is the performance when the quota of each transaction is less than 10 ethers, and Table 4 is the performance when the quota of each transaction is less than 50 ethers. From these three sets of experiments, it can be seen that, in the case of the same number of transactions, the more the nodes in the network, the higher the transaction success rate because the more the nodes, the more the channels in the network. Each transaction has different paths that can be used for trading. When some channels are exhausted, alternative channels can be selected for trading, so the success rate of the transaction is higher.

4.3.2. The Impact of Transaction Quota on Transaction Success Rate. In addition to the network topology, the amount of each transaction also has an impact on the success rate of transactions in the network. Therefore, this experiment has set up several sets of experiments to explore the impact of transactions with different amounts on the same network. The experimental results are shown in Figure 8. It can be seen that the larger the amount of each transaction, the lower the success rate of transactions in the network. The larger the transaction amount is, the faster the channel will be consumed, and many channels will reach an unbalanced state sooner, so that some channels are no longer available, and the number of transactions that can withstand in the network is relatively small.

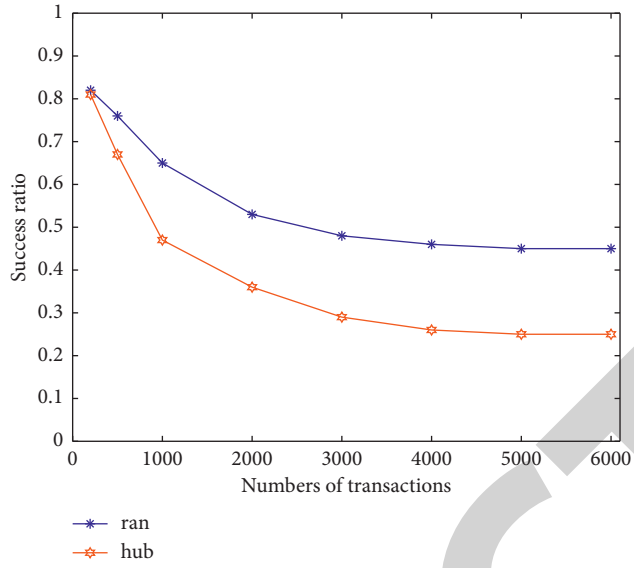

Figure 5: The transaction success rate of the 10-node topology.

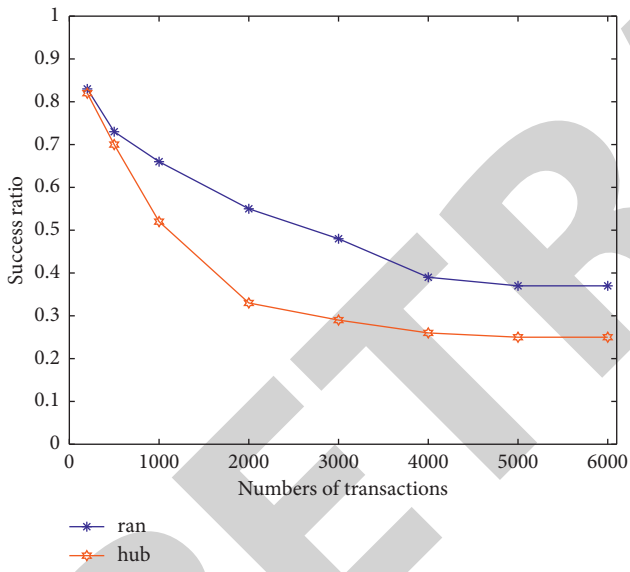

Figure 6: The transaction success rate of the 50-node topology.

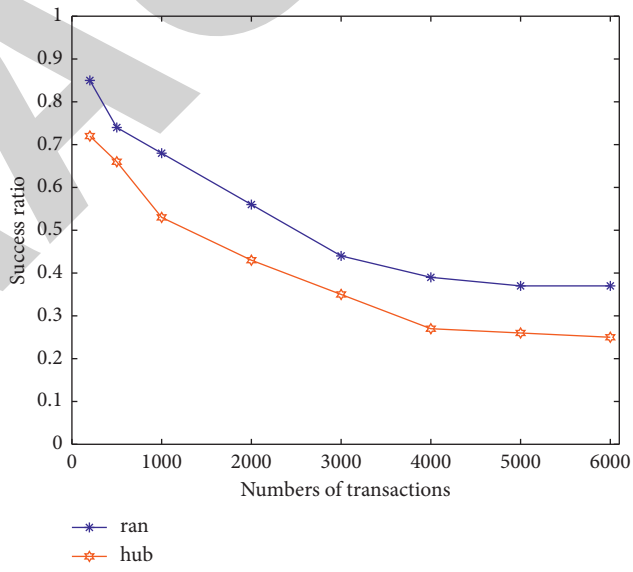

Figure 7: The transaction success rate of the 100-node topology.

4.4. Gas Consumption. Gas consumption may be the most concerned metric for users because it involves transaction fees. Therefore, this paper conducts experiments of different scales on the topology of the Lightning Network and the randomly generated topology to study the impact of different payment methods on gas consumption. In this experiment, the channel chain update interface of the smart contract is called once every ten state updates. In actual use, it can be called once before closing the channel to reduce gas consumption. The result is shown in Figure 9. "Ours" in the figure means using the multiparty payment channel proposed in this paper to complete multiparty transactions, "Dual" means using two-party payment channels to complete multiparty transactions, and "Eth" means not using payment channels. To conduct transactions directly on the Ethereum blockchain network, from the results in Figure 9, it can be seen that the multiparty payment channel transaction method proposed in this paper has great advantages in gas consumption, which benefits from the channel selection and multiparty transaction optimization proposed in this paper. The strategy reduces the number of interactions between users and also reduces the number of smart contract calls, making smart contracts consume less gas. The transaction method of the payment channel of both parties will reuse some of the same channels, so the performance on gas is slightly worse. If the transaction is performed directly through the Ethereum blockchain, each transaction needs to call the Ethereum transaction interface each time the gas consumption is fixed and the payment delay is long.

Table 2: Influence of the number of nodes on success rate when payment amount <5.

Number of transactions	Success ratio (%)				
	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
0	100	100	100	100	100
500	95.8	98	96.2	98.5	99
1000	92.5	95.3	94.8	96.4	97.2
1500	85.3	90	87.5	92.5	94.5
2000	81.2	87.5	85	90.9	91.3
2500	73.5	82.6	80.5	86.4	89.8
3000	68.2	75	75	82.6	86.2
3500	65.7	70.8	72	78.8	80.9
4000	60	68.5	70	75.9	78.7

Table 3: Influence of the number of nodes on success rate when payment amount <10.

Number of transactions	Success ratio (%)				
	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
0	100	100	100	100	100
500	89.2	91.2	92.5	93.8	95
1000	82.5	85.3	85.8	86.5	87.4
1500	75.3	76	77.5	81.2	84.5
2000	61.6	67.3	71.5	75.9	76.1
2500	53.2	62.2	68.2	72.4	73.5
3000	38.2	43.5	50	52.9	56.4
3500	25.5	30.8	39.6	42.5	44.7
4000	20	28.8	30	35.8	38.3

Table 4: Influence of the number of nodes on success rate when payment amount <50.

Number of transactions	Success ratio (%)				
	$N = 100$	$N = 150$	$N = 200$	$N = 250$	$N = 300$
0	100	100	100	100	100
500	85.2	86.8	88	89.5	92
1000	72.1	75.5	78.6	80	85.4
1500	55.8	56.2	61.5	66.3	69.5
2000	41.6	47.1	51.2	55.4	57.3
2500	31.2	32.5	38.5	42.1	43.6
3000	20.5	23.8	27	31.5	35.7
3500	18.5	20	19.6	22.5	26.5
4000	17.6	18.8	18	20	24

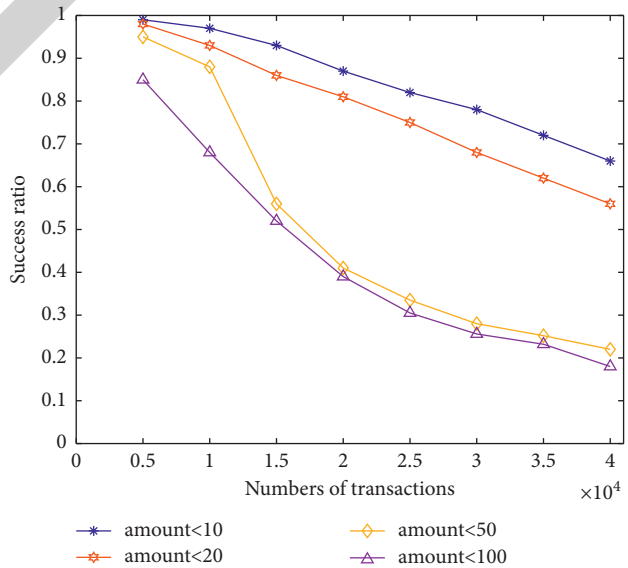

Figure 8: The impact of transaction amount on success rate.

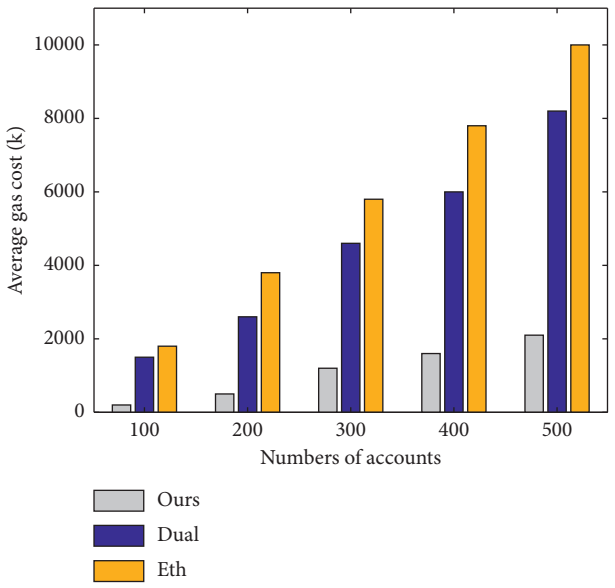

Figure 9: Comparison of gas consumption in different methods.

5. Conclusion

In recent years, blockchain technology has developed rapidly, but its shortcomings cannot be ignored. Aiming at the shortcomings of high transaction delay and low throughput of the blockchain network, this paper designs a new multiparty payment channel based on blockchain and smart contracts, which aims to reduce the delay of blockchain transactions and improve blockchain transactions. Throughput, this paper mainly carried out the following tasks: (1) This paper researched the literature and projects related to the blockchain payment network and analyzed the current implementation of the blockchain payment network and its defects. (2) Based on the smart contract, the two-party payment channel was designed, the process of its creation, update, and closing stages was introduced in detail, the subchannel was designed to support the parallel transaction of the payment channel, and the performance of the two-party payment channel was analyzed. (3) Based on the two-party payment channel, a multiparty payment channel based on blockchain and smart contracts is proposed, which extends the application scenario of the payment channel from a two-person application to a multiperson online application, so that the payment channel can be applied to multiperson online games, crowdfunding, auctions, and other application scenarios, and a trading strategy based on channel selection and multiparty transaction optimization is designed to ensure the balance of the channel and reduce the communication overhead and gas consumption during transactions. (4) This paper developed smart contracts for multiparty payment channels, performed blockchain simulation and client-side simulation, and researched the performance indicators of multiparty payment channels such as transaction delay, transaction success rate, and gas consumption. Through comparative experiments, the feasibility and advantages of the scheme proposed in this paper are verified.

Data Availability

The author did not obtain analytical permission from the data provider because of trade confidentiality.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] X. W. Xu, C. Pautasso, and L. M. Zhu, "The blockchain as a software connector," in *Proceedings of the 2016 Working IEEE/IFIP Conference on Software Architecture*, pp. 182–191, Venice, Italy, April 2016.

[2] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: usin block chain to protect personal data," in *Proceedings of the 2015 Security and Privacy Workshops*, pp. 180–184, San Jose, CA, USA, May 2015.

[3] M. Swan, "The quantified self: fundamental disruption in big data science and biological discovery," *Big Data*, vol. 1, no. 2, pp. 85–99, 2013.

[4] M. Swan, "Blockchain thinking: the brain as a decentralized autonomous corporation," *IEEE techn#ology and Society Magazine*, vol. 34, pp. 27–29, 2015.

[5] C. Dwork, "Differential privacy," *Automata, Languages and Programming*, vol. Part II, pp. 1–12, 2006.

[6] L. Sweeney, "k-Anonymity: a model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[7] N. H. Li and T. C. Li, "t-closeness: privacy beyond k-anonymity and l-diversity," in *Proceedings of the 2007 IEEE International Conference on Data Engineering*, pp. 106–115, Istanbul, Turkey, April 2007.

[8] K. McKusick and S. Quinlan, "Gfs," *Communications of the ACM*, vol. 53, no. 3, pp. 42–49, 2010.

[9] F. Chang, J. Dean, and S. Ghemawat, "Bigtable: a distributed storage system for structured data," *ACM Transactions on Computer Systems*, vol. 26, no. 2, p. 4, 2008.

[10] A. Schaub, R. Bazin, O. Hasan, and L. Brunie, "A trustless privacy-preserving reputation system," in *Proceedings of the ICT Systems Security and Privacy Protection*, pp. 398–411, Ghent, Belgium, May 2016.

[11] N. Szabo, "Formalizing and securing relationships on public network," vol. 2, 1997.

[12] M. Swan, "Blockchain: blueprint for a new economy," O'Reilly Media, Inc., Sebastopol, CA, USA, 2015.

[13] I. Toshiya, O. Yuji, and S. Hiroki, "A language-dependent cryptographic primitive," *Journal of Cryptology*, vol. 10, no. 1, pp. 37–49, 1997.

[14] A. Kosba, A. Miller, and E. Shi, "Hawk: the blockchain model of cryptography and privacy-preserving smart contracts," in *Proceedings of the Security and Privacy. IEEE*, pp. 839–858, San Jose, CA, USA, May 2016.

[15] T. Swanson: Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems, 2015.

[16] N. P. Smart: Secure multi-party computation. 2013.2013.

[17] D. Morris, "Bitcoin is not just digital currency, it's napster for finance," *CNN Money*, vol. 12, pp. 12-13, 2014.

[18] L. Luu, D. Chu, and H. Olickel, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 254–269, Vienna, Austria, Ocutober 2016.

[19] T. Chen, X. Q. Li, and X. P. Luo, "Under-optimized smart contracts devour your money in: proceedings of the 2016 IEEE International Conference on Software Analysis," *Evolution and Reengineering*, vol. 2017pp. 442–446, Suita, Japan, March 2016.
[20] O. S. Cryptocurrencies, "Smart contracts, and artificial intelligence," *AI matters*, vol. 1, no. 2, pp. 19–21, 2014.
[21] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, pp. 1–9, 1997.
[22] D. Chaum, "Blind signatures for untraceable payments," *Advances in Cryptology*, pp. 199–203, 1983.
[23] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, 2002.
[24] F. Beres, I. A. Seres, and A. A. A. Benczur, "Cryptoeconomic traffic analysis of Bitcoin's lightning network," 2019, https://arxiv.org/abs/1911.09432.
[25] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, vol. 2008, Article ID 21260, 2008.