applied sciences

MDPI

Article

Efficient Cross-Chain Transaction Processing on Blockchains

Wenqi Wang, Zhiwei Zhang *, Guoren Wang and Ye Yuan

School of Computer Science & Technology, Beijing Institute of Technology, Beijing 100811, China; 3120191051@bit.edu.cn (W.W.); wanggr@bit.edu.cn (G.W.); yuan-ye@bit.edu.cn (Y.Y.)
* Correspondence: zwzhang@bit.edu.cn

Abstract: Blockchain has received great attention in academia and industry due to its decentralization and immutability. From the perspective of transaction processing, blockchain is a distributed shared ledger and database with the characteristics of decentralization, traceability, and transparency. These features ensure the security of blockchain's reliability. However, because a blockchain network requires complex consensus verification between users, it causes problems such as a high cost of data exchange and a low system throughput. Such problems are aggravated when executing a cross-chain transaction, as it is particularly important to ensure the atomicity and isolation of transactions across the blockchain. Considering this, in this paper, we propose the cross-chain transaction processing flow of EOVPC and efficient transaction processing based on version control. Different from the existing cross-chain transaction approaches based on locking, we propose optimistic approaches in which the updated data can be used immediately, with a rolling back procedure that guarantees atomicity. We conducted extensive experiments, which show that our approaches can improve the throughput and success rate significantly.

Keywords: blockchain; cross-chain interoperation; optimistic cross-chain consensus

check for updates

Citation: Wang, W.; Zhang, Z.; Wang, G.; Yuan, Y. Efficient Cross-Chain Transaction Processing on Blockchains. *Appl. Sci.* **2022**, *12*, 4434. https://doi.org/10.3390/app12094434

Academic Editor: Eui-Nam Huh

Received: 27 March 2022
Accepted: 25 April 2022
Published: 27 April 2022

1. Introduction

Blockchain has been developing rapidly in recent years. Different from traditional databases, blockchain has the characteristics of decentralization, common maintenance, openness and transparency, immutability, and traceability. These characteristics enable blockchain to better adapt to a data management system whose security requirement is high. Each chain can be taken as an unit and provide services for data maintenance and sharing. The data flow within the blockchain is safe and effective and works in real time. However, as the increase in blockchain applications, there is increasing demand for cross-chain transactions between multiple chains. The data flow between blockchains is difficult to operate due to the lack of trust between blockchains. Because of this, blockchain cross-chain processing has become a hot research topic in the field of blockchain.

Recently, cross-chain transaction processing mainly aims at asset transfer and asset exchange between two blockchains. These cross-chain transaction processing methods are all for the asset circulation on the blockchain of "currency", but it is difficult to realize cross-chain interoperation based on a smart contract, which limits the application scope of cross-chain blockchain. At the same time, most of these methods are difficult to implement, and tedious transaction processing makes transaction execution efficiency low.

The motivation of this paper is to base cross-chains more on smart contracts, rather than only assets, so that cross-chain interoperability can be more flexible. At the same time, the efficiency of the cross chain needs to be improved. To achieve these goals, we propose our cross-chain transaction processing, known as the OVC procedure. It is based on optimistic version control for the data for all involved chains. We have conducted extensive experiments to show the effectiveness of our cross-chain method and the performance of different cross-chain methods with different working loads.

The contributions of this paper are as follows: This paper studies and compares the existing four mainstream blockchain cross-chain methods, and designs and implements a method to complete cross-chain interoperation among multiple blockchains by using cross-chain scheduling. This cross-chain method is then optimized to become more efficient. The work completed includes (1) the design of a cross-chain scheduling module to realize the cross-chain method of a blockchain for cross-chain interoperability among multiple blockchains, (2) the design of an EOVPC architecture to ensure the atomicity of cross-chain transactions, (3) the design of cross-chain blocks and cross-chain databases to support the traceability and reproduction of cross-chain operations, (4) the optimization of the design of the cross-chain method for a special case of cross chains, and (5) the design and implementation of the cross-chain test platform as well as a test and analysis of the cross-chain system.

2. Related Work

Communication between blockchains involves two blockchains: the source and the target [1]. The source chain initiates a transaction that can be executed on the target chain. At present, "cross chain" refers to "asset transfer" and "asset exchange". A cross-chain asset transfer involves three atomic transactions: (1) a source chain asset lock, (2) transaction confirmation on the blockchain, and (3) the creation of assets on the target chain [2–4]. According to a technical report from the National Institute of Standards and Technology (NIST), blockchain interoperability is defined as "the composition of distinguishable blockchain systems, each representing a unique distributed data ledger, in which atomic transaction execution can span multiple heterogeneous blockchain systems, where data is recorded in a system [where] the blockchain is semantically reachable, [is] verifiable, and can be referenced by another possible external transaction" [5].

In the general communication protocol proposed by Abebe et al., interoperability is defined as the "transmission or exchange of data or value between different ledgers with guaranteed validity" [6]. Pillai and Biswas noted that "the purpose of cross-communication is not to directly change the state of another blockchain system. Instead, cross-communication should trigger some set of functions on the other system that are expected to operate within its own network" [7].

Hardjono et al. define blockchain transaction activity as "the completion of an application-level transaction that is independent of the participating blockchain system" [8]. Both transactions and sub-transactions are guaranteed to be completed within a specified time, i.e., with a "best effort to deliver".

Two blockchain cross-chain protocols include the cross-chain communication protocol (CCCP) and the cross-blockchain communication protocol (CBCP). Rafael Belchior et al. made the distinction that the CCCP process communicates against a homogeneous blockchain. The CBCP targets heterogeneous blockchain communication [1]. Zamyatin et al. demonstrated that "there is no asynchronous cross-chain communication (CCC) protocol that tolerates [the] abnormal behavior of nodes" [9]. The authors use the simplification of the fair exchange problem to show that correct cross-chain communication is just as difficult as the fair exchange problem [10]. As a result of the proposed theorem, the authors state that, "without trusted third parties, there can be no CCC protocol for tolerating misbehaving nodes". Trusted third parties can be centralized or decentralized. For example, a centralized trusted party is a trusted validator [11], and a decentralized trusted party can be another blockchain. Lafourcade and Lombard Plate [12] formalize the issue of blockchain interoperability, arguing that a fully decentralized source blockchain cannot verify the existence of data on the target blockchain through practical efforts. More specifically, no protocol assumes that a complete client can implement its interoperable functions, such as asset transfers, without the assistance of third parties.

For these two cross-chain operations, there are several mainstream cross-chain technologies: (1) notary schemes, (2) hash-locking, (3) sidechains/relays, and (4) distributed private key control.

The notary scheme involves a notary, an entity that monitors multiple chains, triggering transactions in the chain when an event occurs on another chain [13]. Exchanges facilitate signaling among market participants by managing order books and matching buyers and sellers. If the trust anchor is placed on a centralized party that holds the user's private key or funds, the notary is a centralized exchange. Otherwise, if the exchange does not perform trades on behalf of the user and only provides matching services, it is considered a decentralized exchange [14].

In layman's terms: A trusted third party who tells Chain B what is happening to Chain A, or who tells Chain B whether certain information is true.

For hash locking-based approaches, Hashed Time-Lock Contracts (HTLCs) originally emerged as an alternative to centralized switching because they support cross-chain atomic operations [15]. HTLC technology uses hash locks [16] and time locks [17] to enforce the atomicity of operations, usually between two sides. The trader makes a trade by promising to provide proof of encryption to the other party before the time runs out.

An HTLC is used in Bitcoin for conditional or cross-chain payments, namely atomic exchange [18–20]. Atomic swapping can be thought of as a distributed commitment [21], able to fend off a Byzantine adversary. Therefore, atomic cross-chain switching is a distributed atomic transaction [20], which is settled on the chain.

Side chains are two existing blockchains that are interoperable [22,23], extensible (for example, through blockchain sharing [24]), and upgradable [25]. Side chains allow for different interactions between participating blockchains, the most common being the transfer of assets between the main and side chains (bidirectional hooking) [26,27]. A relay solution is an SPV [28] client of the source blockchain running on the target blockchain, capable of verifying transactions [29].

The relay/side chain does not rely on a trusted third party to help it verify the transaction; rather, it is verified by the receiving chain itself after receiving the transmit chain data. A can read B, but B cannot read A; if the A-C chain can read all chains, it should also be an "on-chain" intermediary, and the whole process is "A-C-B". The trunk is more like the dispatch center of multiple chains. When a "side chain" links many main chains, it becomes a trunk chain. Side chains need to anchor each other's data.

The private keys of various assets are controlled by distributed nodes, and the original chain assets are mapped to the cross-chain to ensure the interconnection of various assets in the blockchain system. The core of distributed private key control is distributed control management; that is, the ownership and use rights of assets are separated, and the control rights of digital assets on the original chain are safely transferred to the decentralized system.

A distributed private key blocks a chain inside the private key, which is divided into N and put it into N shares. To share it with N participants, everyone can grasp some of the private key participants and can only collect K of them after the distribution of the private key, to restore the integrity of a private key, to restore a full private key, to be able to access the private key assets, or to perform an unlock. In other words, distributed private key control transfers account control through the distributed storage of user account private keys on the network.

A comparison of the four existing cross-chain methods is shown in Table 1.

By summarizing the above four mainstream cross-chain methods, it can be found that they are all value exchanges for "coin" and designed to provide interoperability between cryptocurrency systems [13], aiming to improve the transaction processing capacity of Bitcoin-dominated tokens. Besides the hash locking method, other methods include connecting the token network with the public chain network for token transaction performance improvement and transaction verification. However, blockchain is no longer limited to the field of cryptocurrency and is expected to become a data storage and management technology that is as widely used as a database. Therefore, the blockchain cross chain should not be limited to the transfer of assets and should focus on transaction interoperation between blockchains. With the popularity of the alliance chain, cross-chain technology should also be applied to the scope of the alliance chain. In this paper, we will design and implement the cross-chain interoperation of a blockchain based on Hyperledger Fabric, an important project in the federation chain.

Table 1. Comparison of the four existing cross-chain methods.

	Notary Schemes	**Hash Locking**	**Sidechains/ Relays**	**Distributed Private Key Control**
Asset exchange	Yes	Yes	Yes	Yes
Asset transfers	Yes	No	Yes	Yes
Support smart contracts of different blockchains	Hard	No	Hard	Hard
The difficulty of implementation	Medium	Easy	Hard	Medium

3. Cross-Chain Processing-Based EOVPC

In this section, we introduce our approaches for efficient cross-chain transaction processing. A cross-chain transaction could be referred to as the transfer of assets or the sharing of data between chains. For each committed cross-chain transaction, all blockchains not only must ensure the success of the operations related to itself, but also need to consider the situations of other cross-chain transactions. When all operations succeed, then the cross-chain transaction can commit, and the data read and written by each chain can be valid. Otherwise, the cross-chain transaction fails, and the modified data is invalid.

Since there may be multiple cross-chain transactions involved in collaborative cross-chain transactions, it is necessary to coordinate the completion of sub-transactions on each sub-chain, and judge the final result of the collaborative cross chain according to the execution results of each chain. Here, we use a sub-transaction to represent all operations related to one chain. It is possible that the end result of the collaborative cross chain may be inconsistent with the execution result of the sub-transaction, in which case the cross-chain transaction needs to restore the state of the chain to what it was before the cross-chain transaction was executed. Therefore, the execution results of sub-transactions cannot be applied in practice before the final cross-chain results arrive. Different from the existing pessimistic consensus, in which the data related to cross-chain transaction will be locked until the cross-chain transaction processing finishes, we chose to use an optimistic cross-chain consensus. In our approach, we allow the results of each sub-transaction to be valid as soon as the sub-transaction can be finished. This method will increase the performance significantly, because the following transaction can use the results of each sub-transaction immediately. On the other hand, it needs to deal with failed cross-chain transactions. Considering this, we propose a pre-commit period for sub-transactions, in which the data is valid but can be rolled back in the future.

The collaborative cross chain will carry out two cycles of information transmission. The first cycle is the execution cycle of the cross-chain transaction, and the other cycle is the consensus cycle of the cross-chain transaction results, and the cross-chain result consensus cycle is broadcast to each chain. In the first cycle, a cross-chain transaction will chain up and update the ledger before it collaborates with the cross-chain result consensus, but the relevant updates are not valid for other transactions until the collaborative cross-chain transaction ends. This stage is called pre-commit. The collaborative cross-chain transaction flow is increased from the EOV used by Fabric to the EOVPC.

In the process of multi-chain collaboration and the cross chain, cross-chain scheduling is responsible for the statistics of the cross-chain transaction response information, consensus, and announcement of the final cross-chain result. Cross-chain scheduling will notify each blockchain of the consensus result of the collaborative cross chain, and each blockchain will confirm the pre-commitment or rollback of the pre-commitment result of the previous cross-chain transaction according to the consensus cross-chain result. The cross-chain database maintained by each node is updated according to the results of each cross-chain stage.

For multi-chain collaborative cross-chain transactions, cross-chain transactions are specifically packaged as cross-chain blocks during the consensus ordering of Orderer nodes on each blockchain.

3.1. EOVC & EOVPC

In the transaction flow of Fabric, a client proposing a transaction to an endorsement node, which simulates the execution and returns the endorsement result to the client. As shown in Figure 1, the client counts all endorsement results, and if the results are consistent, the simulation is valid, encapsulated into transactions, sent to the ordering node to agree on the transaction order, and encapsulated into blocks. The sorting node sends the encapsulated blocks to the Leader node of each organization in the blockchain. The Leader node simply verifies the block transaction and distributes it to neighboring nodes in the organization. Each node forwards the block to the neighbor node that has not yet obtained the block, so as to achieve the purpose of distributing the block to all nodes in the organization. When each node receives a new block, in addition to verifying the format of each part of the block and chain code rules, the most important thing is to check the read and write conflicts for transactions in the block. Valid transactions will update the world state and various databases, and the chain on the block will save the file system.

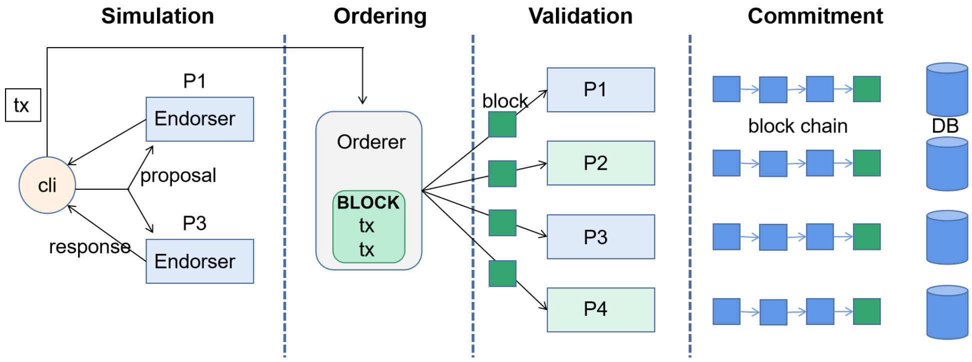

Figure 1. EOVC.

The EOVPC architecture can still handle cross-chain local transactions, and the transaction flow remains unchanged. For cross-chain transactions, the blockchain-oriented SDK receives cross-chain transaction requests from cross-chain scheduling, and the blockchain executes the EOVPC process to complete the cross-chain transaction and collaborative cross-chain transaction processing.

As shown in Figure 2, first, the cross-chain transaction request is sent to the endorser node on the blockchain for transaction simulation by the SDK of the chain, which is specialized in dealing with cross-chain transactions. The node records the original value of relevant data and records it during the simulation. Since the node identifies the transaction as a special cross-chain transaction, the node starts to construct a cross-chain transaction dependency graph with the cross-chain transaction as the head node. This dependency diagram documents a DAG that starts with a cross-chain transaction and uses cross-chain transaction-related data as dependencies for subsequent transactions. While this cross-chain transaction dependency graph exists, all new transactions on the chain check the dependencies and update the dependency graph dynamically.

In the ordering phase, the ordering node identifies the cross-chain transaction and immediately terminates and encapsulates the current block. The cross-chain transaction is individually encapsulated into a cross block, which is pulled and distributed by the Leader node and verified.

In the pre-commitment phase, the node identifies cross-chain blocks and updates the data accordingly. Finally, cross-chain transaction results are returned to the cross-chain SDK. At this point, the cross-chain transaction is complete.

In the following, the second phase of cross-chain transaction processing is described. Cross-chain scheduling collects the cross-chain transaction feedback of each chain cross-chain SDK and determines the success or failure of collaborative cross-chain transactions as the result of a collaborative cross-chain consensus. The cross-chain scheduling will send the collaborative cross-chain consensus result to the ordering node, and the ordering node will identify and encapsulate the result message to the confirmation block, which will be transmitted to each node in the chain at the fastest speed.

Each node on the chain determines the cross-chain transaction result according to the confirmation block received, so as to determine whether to delete the cross-chain transaction dependency graph with the cross-chain transaction as the root node or roll back the related transaction according to the sub-dependency graph. Meanwhile, the cross-chain database updates the cross-chain transaction status according to the cross-chain consensus results.

At this point, a collaborative cross-chain transaction is completed.

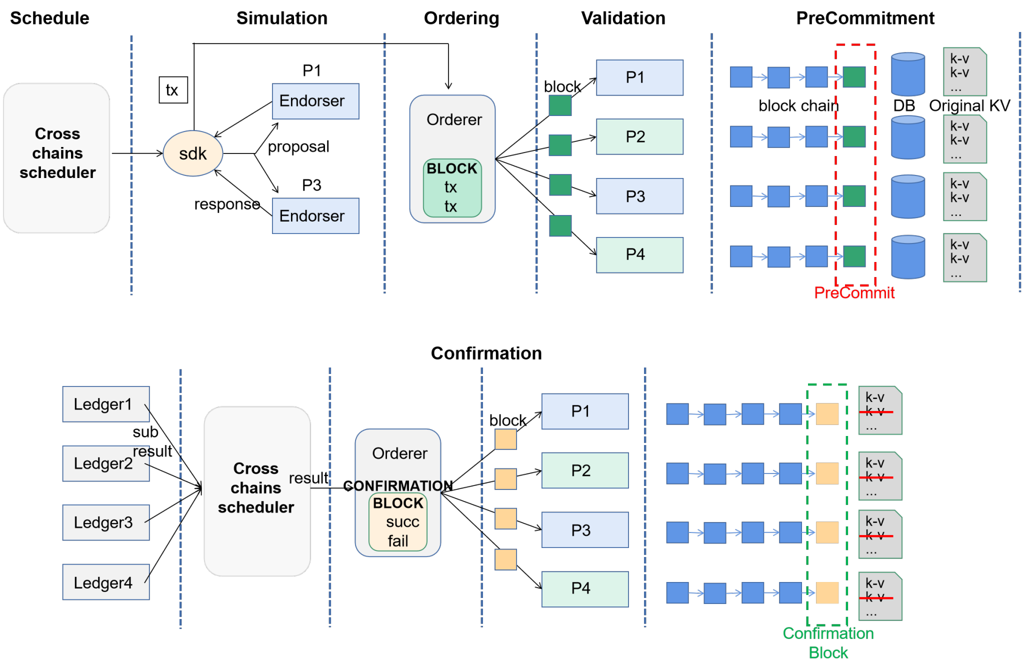

Figure 2. EOVPC.

3.1.1. Simulation Processing

In order to make the result of cross-chain transaction used earlier, and when cross-chain transaction is rolled back, other transactions that depend on this transaction can also be rolled back accurately, we design a cross-chain transaction dependency graph. We will rely on the validity of the cross-chain transaction, which is called transaction validity dependency, and the post-order transaction depends on cross-chain transactions because of the derivative data of the cross-chain correlation data or cross-chain data. This graph showsa dag-chain transaction as a starting point, the post-order transaction is dependent on cross-chain transactions, and the new update is updated based on the time order, which is not required for transactions that are unrelated to the cross-chain transaction. The transaction that exists on the chain transaction is recorded in the simulation phase, and the original value of its read-write collection is recorded. As shown in Figure 3, the transaction T0 is a local transaction that is already on the chain but has not yet been verified by cross-chain results. The letters A to E represent data used by the transaction. The five transactions from T0 to T4 read and write data respectively. T1, T2, T3, and T4 is directly or indirectly dependent on T0.

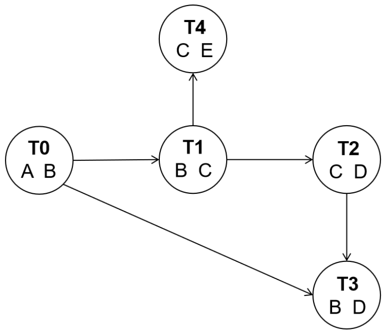

Figure 3. Cross-chain transaction dependency.

After the completion of the cross-chain transaction, each node processes each transaction on the graph according to the cross-chain transaction dependency graph. If the cross-chain transaction succeeds, the cross-chain transaction dependency graph will be deleted, and the original data of the transaction related values stored on the graph will be released. If a cross-chain transaction fails, the cross-chain transaction and its subsequent dependent transactions are rolled back according to the dependency graph.

3.1.2. Pre-Commitment Processing

The execution results of cross-chain sub-transactions cannot take effect immediately before the collaborative cross-chain result consensus. Considering this, for the update operation for the world state, they can be carried out in two stages in the cross-chain transactions: (A) after the completion of all the sub-transactions in a cross-chain transaction but before the collaborative cross-chain consensus result and (B) after the collaborative cross-chain consensus results.

In Case A, due to the delay of blockchain cross-chain transactions processing, the data used by the following transactions are not the latest data updated in real time. Naturally, there is a weakness in the simulation of subsequent transactions with expired data, resulting in the reduction in transaction success rate. This situation is even more serious when the blockchain needs to be rolled back, because the rollback operation will cause the subsequent transactions of the previously successfully committed rollback transactions to become read dirty, resulting in transaction execution result errors. This situation will accumulate with the high-frequency use of relevant data.

Case B chooses to update the world state after consensus cross-chain results, which is relatively simple. After the cross-chain consensus result is coordinated, the consensus result is sent to each chain. The state is updated if the chain succeeds; otherwise, i the world state is operated on. This method can better avoid the sequential transaction read dirty state caused by cross-chain transaction rollback. After the completion of the cross-chain transaction, the node does not need to update the status database and history database immediately, but it still needs to update other blockchain system databases and file systems, such as the block index database. The database is updated again after the cross-chain transaction has been confirmed successfully.

3.1.3. Confirmation Processing

Cross-chain scheduling constructs a special "cross-chain result consensus" block. The task of this block is to bring collaborative cross-chain results to each node without the need for an up-chain, so this block is a simple one that encapsulates only cross-chain transaction identification and consensus results. Cross-chain results are obtained by cross-chain scheduling and sent directly to the Orderer node. The Orderer node identifies the cross-chain consensus result message, encapsulates it into a cross-chain consensus result block, and sends it to the Leader node, who then distributes the cross-chain consensus result block to all nodes in the organization. Compared with the normal transaction flow EOVC, this process reduces the process of simulating endorsements and verifying commits.

The specific process of cross-chain result consensus is as follows: cross-chain scheduling records all cross-chain transactions included in the forwarding process of cross-chain transactions at the beginning of collaborative cross-chain transactions, and continuously monitors each cross-chain transaction until the end of the collaborative cross-chain transactions. When any cross-chain transaction returns a transaction failure message or a cross-chain timeout, the scheduling sends the cross-chain failure message to the Orderer nodes. The Orderer nodes encapsulate the cross-chain failure message into the confirmation block and send it to the nodes in the chain. The cross-chain transaction success message is encapsulated only when the cross-chain scheduler receives all cross-chain transaction success messages for collaborative cross-chain transactions. Each node on the blockchain accepts or rolls back transactions recorded in the cross-chain dependency graph.

This method can be applied to cross chains among multiple chains. In other words, this method supports collaborative cross-chain transactions composed of several sub-transactions. For example, there are three chains, C0, C1, and C2, and there is a collaborative cross-chain transaction involving multiple chains: C1 needs to write C1 and C2, and C1 needs to read data from C2. This collaborative cross-chain transaction consists of three sub-transactions. After analyzing and judging the cross-chain request, the cross-chain scheduler sends the cross-chain request to C1 and C2, and waits for the feedback of the three sub-transactions. After collecting all cross-chain transaction feedback, the cross-chain scheduler determines whether the collaborative cross-chain transaction is successful or not and then feeds back the cross-chain result to all chains, and each chain performs subsequent operations according to the cross-chain result.

It can be seen from the cross-chain process that the cross-chain method using the cross-chain scheduler is still applicable to the cross chain of different consensus algorithms or the cross-chain between heterogeneous blockchains. This is because the key point of cross-chain transaction flow lies in the system's differential treatment of cross-chain transaction and the synchronization of cross-chain transaction results. However, since each blockchain system is rigorous and complex and has its own unique transaction processing process, the blockchain also needs to be transformed when the cross-chain method mentioned in this paper is applied to each blockchain.

4. Optimizations for Cross-Chain Scheduling

The multi-chain cooperative cross-chain system architecture adds a cross-chain scheduling module over the original Fabric system. The cross-chain scheduling module is located between the client and the Fabric network and interacts with the Peer module and Orderer module of the Fabric system to complete cross-chain transaction-related instructions. The architecture of a cross-chain system that includes cross-chain scheduling is shown in Figure 4.

Two interoperable blockchains need to recognize each other's identity and determine the operation authority of each other's blockchains. For example, Chain A and Chain B are two federated chain blockchains. In the case of isolation between chains, Chain A needs to perform read and write operations on Chain B, while Chain B can only perform read operations on Chain A but has no right to write operations on Chain A. In this case, Chain A and Chain B need to register with cross-chain scheduling separately. Each chain that completes the registration will have interactive interface and access permissions. Blockchains that have been registered tell the cross-chain scheduler how much they are allowed to interoperate with other blockchains, namely read-only, write-only, and read/write (both). Default read/write (both) is unavailable.

In this approach, traditional local transactions are still supported. Cross-chain scheduling is transparent to local transactions. For cross-chain transactions, cross-chain scheduling determines whether the target chain permits access to the source chain. If access is available, cross-chain scheduling encapsulates the access request and cross-chain information into a cross-chain proposal and routes it to the target chain. If there is no access, the cross-chain scheduler will not continue forwarding and will construct a cross-chain failure reply without cross-chain permission. Starting from the forwarding of cross-chain requests by cross-chain scheduling, the cross-chain scheduling records outstanding cross-chain transactions and starts timing the cross-chain forwarding time.

Before the cross-chain timeout, cross-chain scheduling waits for the results of sub-transactions of the target chains then summarizes the results to judge whether the cross-chain is successful, and sends the results to each relevant blockchain for subsequent operations.

If the cross-chain timeout occurs, the default cross-chain failure occurs, and the failure response is sent to all relevant blockchains by the cross-chain scheduling package to restore the cross-chain transaction-related operations.

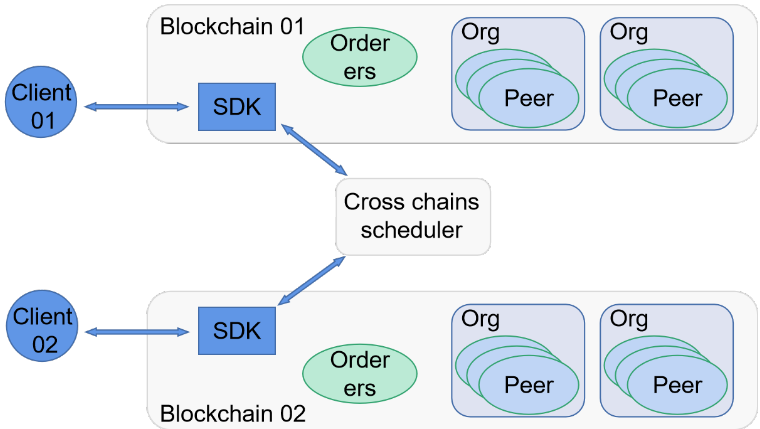

Figure 4. Cross-chain architecture.

The optimized cross-chain approach utilizes cross-chain scheduling to adopt an optimistic cross-chain consensus when there is only one sub-transaction. In other words, the success of a cross-chain transaction is only related to one sub-transaction, so in this case, there is no need for cross-chain scheduling to collect sub-transactions results and then distribute the result messages to all participating chains again. The result of the target chain is valid immediately after the sub-transaction is processed, and no revalidation is required. At the same time, the target chain feeds back the result of cross-chain transaction execution to the source chain through cross-chain scheduling.

The cross-chain scheduler also plays the role of cross-chain transaction sequencing. The relative order of cross-chain transactions should be the same across different blockchains. In other words, if two cross-chain transactions T0 and T1 have sub-transactions on Chain A and Chain B, the order of T0 and T1 must be the same in both chains. The relationship between the cross-chain scheduler and each block chain is equivalent to the relationship between the ordering node and each peer in a fabric. The order of cross-chain transactions depends on the ordering of cross-chain scheduling.

The cross-chain method based on the EOVPC process improves the efficiency of cross-chain transaction processing by using cross-chain scheduling. First, cross-chain transactions are encapsulated into special transaction requests in the cross-chain scheduling module. The blockchain nodes identify cross-chain transactions, finish the ordering of local transactions as quickly as possible, and then individually encapsulate cross-chain transactions into blocks. Blocks with only one transaction will be processed faster. Secondly, in the cross-chain result consensus stage, the cross-chain scheduling specifically provides messages to the ordering nodes and encapsulates special confirmation blocks. Confirmation blocks inform the various nodes of the blockchain in a more direct process, which also improves the efficiency of cross-chain transactions in the result consensus phase.

Compared with the cross-chain method based on the EOVPC, the optimized cross-chain method is more concise in transaction flow. The optimized cross-chain approach is mainly for one-way transactions with only one sub-transaction. This situation eliminates the cross-chain result consensus required for collaborative cross-chain transactions consisting of multiple sub-transactions involving multiple blockchains. This greatly reduces the complexity of the cross chain and improves transaction completion efficiency. At the same time, a cross-chain request containing only one sub-transaction enables the target chain to act on the sequenced transaction immediately after executing the sub-transaction, without waiting for information to be synchronized with the source chain. This also makes the optimized cross-chain approach more efficient.

5. Experimental Evaluation

In the previous sections, we looked at changing several modules of the fabric in several ways to implement its cross-chaining. To test and analyze the performance of the cross-chain blockchain system, first, we focused on the throughput of cross-chain systems and the success rate of cross-chain transactions. Secondly, we focused on the effect of different proportions of cross-chain transactions to local transactions, as well as the effect of different proportions of read and write transactions, on the throughput and transaction success rate of the cross-chain system.

5.1. Setup

The experimental environment consisted of the following: The operating system was a 64-bit Linux, version 5.4.0-90-generic. The server contained 32 CPUs, each of which was an Intel(R) Xeon(R) Silver 4110 with the architecture x86_64. Each CPU ran at 2.1 GHz with 32 K of L1d cache, 32 K of L1i cache, 1024 K of L2 cache, and 11,264 K of L3 cache. The memory device was a DDR4 with 32 G.

Fabric chooses LevelDB to be the world state database. Individual nodes in the fabric run used docker. We set each blockchain to have four peers (commitment nodes) and one ordering node. The blocksize was set to 256.

5.2. Benchmark Framework and Workload

At present, the evaluation test of the blockchain system mainly focused on the throughput and transaction success rate of the system under different conditions. As the development of the blockchain is still in its infancy, there are not many test platforms for the blockchain system. Among single-chain blockchain test platforms, there are test platforms such as the hyperledger-caliper, blockbench, and fabric++-benchmark that can test and compare blockchain systems. However, there are even fewer testing platforms for cross-chain blockchains.

5.2.1. Framework

Due to the lack of mature test platforms for cross-chain blockchain systems, we established our own cross-test platform according to the cross-chain system design designed in this paper. Our blockchain cross-chain test platform can support highly concurrent requests to initiate local transactions and cross-chain transactions, define all blockchains participating in the cross-chain, and support the adjustment of the cross-chain transaction and the local transaction ratio allocation, the read and write transaction ratio, the number of cross-chain transactions, and the transaction trigger time interval of the cross-chain system. Subsequently, we used this cross-chain test platform to conduct a series of collaborative cross-chain and crosschain-op tests and comparisons on our cross-chain system.

5.2.2. Workload

In the experiments, we tested and compared the two cross-chains functions.

First, we tested our cross-chain system with a highly configurable workload. This workload is very suitable for collaborative cross-chain scenarios, which can well simulate collaborative cross-chain transactions between multiple blockchains or a combination of multiple cross-chain transactions, that is, complex cross-chain blockchain interoperability.

In the experiment, we set the proportion of cross-chain transactions to 0, 0.3, 0.5, 0.8, and 1, the proportion of read and write transactions to 0, 0.3, 0.5, 0.8 and 1, and the number of cross-chain transactions to 1, 2, and 3. The influence of a different cross-chain transaction ratio and the number of cross-chain transactions on the performance throughput and success rate of the cross-chain system was obtained. CrossRate represents the ratio of cross-chain transactions to local transactions, and RWRate represents the read/write ratio of the transactions. MultiCross refers to the collaborative cross-chain transaction composed of multiple sub-transactions.

Second, we tested the crosschain-op; that is, we tested and compared the single-span cross-chain transactions of the cooperative cross-chain system and the crosschain-op system, respectively, to observe the effects of the single cross-chain intersection synergistic cross-chain reduction in transaction flow and the reduction in two special blocks, the cross-chain transaction block and the synergistic cross-chain result block, on the cross-chain.

Third, we tested whether more intensive cross-chain transactions will adversely affect the efficiency of the cross-chain system in the same case of collaborative cross-chain transactions. We designed different non-uniform cross-chain transaction densities for testing.

In one run, we fired a fixed number of transactions at a high speed and concurrently. In each experiment, we triggered 10,000 transactions, including blockchain local transactions and cross-chain transactions. The number of sub-transactions of cross-chain transactions as a parameter is a factor we focused on. All transactions in one round of experiments were triggered in a highly concurrent manner, and requests were sent to the blockchain system. The test platform recorded the time it took to complete all transactions as well as the processing results and failure causes of each transaction. We used TPS (the average number of successful transactions per second) and success rate to evaluate the system performance for each round of experiments.

5.3. Impact of the Cross-Chain Transaction Ratio

First we discuss the impact of the proportion of cross-chain transactions to the total number of transactions on the cross-chain blockchain system. In a traditional blockchain system, where all transactions are local, we set the proportion of cross-chain transactions to 0. We conducted a total of five groups of experiments for comparison. The read/write ratios of these 5 groups of experiments were set as 0, 0.3, 0.5, 0.8, and 1. In each group of experiments, the proportion of cross-chain transactions was set as all local transactions (CrossRate = 0), cross-chain transactions accounted for a minority (CrossRate = 0.3), cross-chain and local transactions were balanced (CrossRate = 0.5), and cross-chain transactions accounted for a majority (CrossRate = 0.8). All cross-chain transactions was indicated by a CrossRate of 1. The number of sub-transactions for all cross-chain transactions was set to 2, because a traditional cross-chain operation can be thought of as consisting of two cross-chain transactions. We statistically compared TPS, SuccRate, and cross-chain conflict rate (because the status code of cross-chain conflict was set to 401, here referred to as 401 Rate) for each group of experiments. Experimental statistical results are shown in Figure 5.

As we can see, the results of these 10 groups of experiments had the same variation trend, so it can be seen that CrossRate had the same influence on system performance under different RWRate conditions. In the case of fixed RWRate, as the proportion of cross-chain transactions increased, that is, the CrossRate gradually increased, the number of successfully processed transactions per second (TPS) of the system gradually decreased, the success rate of transaction processing of the system gradually decreased from nearly 100% to about 80%, and the proportion of cross-chain conflicts gradually increased from 0 to nearly 100%.

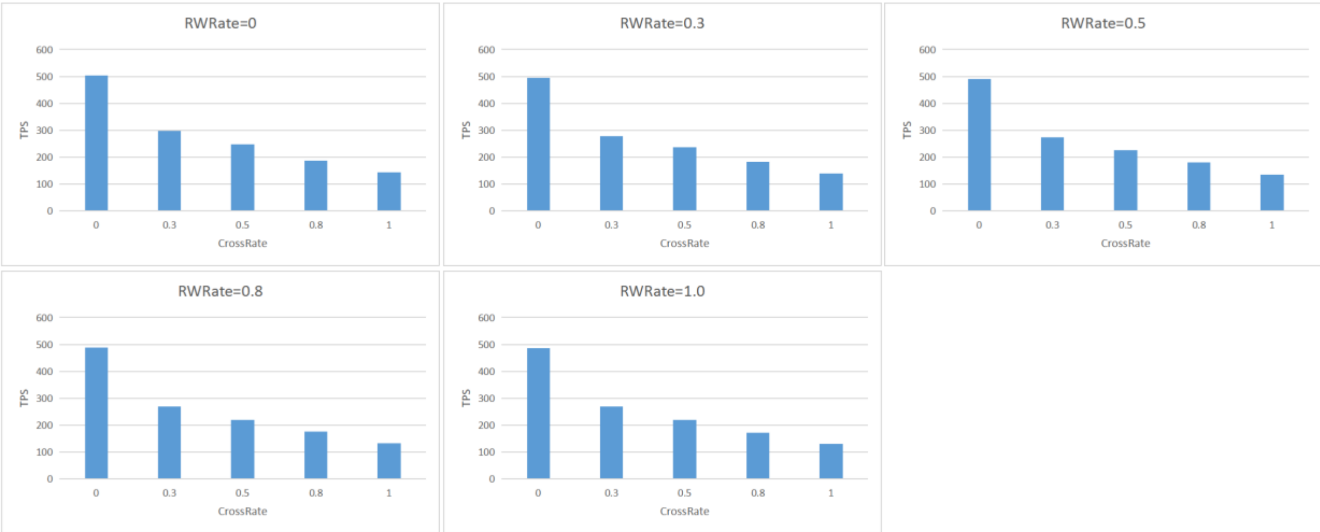

Figure 5. The effect of the cross-chain transaction ratio on TPS.

5.4. Impact of the Number of Sub-Transactions of Crosschains

In our cross-chain system, a collaborative cross-chain transaction can be composed of any number of sub-transactions. Next, we discuss the influence of the number of cross-chain transactions on system performance.

We set the number of cross-chain transactions as 1, 2, and 3, respectively, to conduct three groups of experiments. Different read/write ratios were set for each group of experiments, and experiments were carried out when CrossRate was 0, 0.3, 0.5, 0.8, and 1, and the average throughput, average success rate, and average cross-chain conflict rate at each CrossRate were calculated. The experimental results are shown in Figure 6.

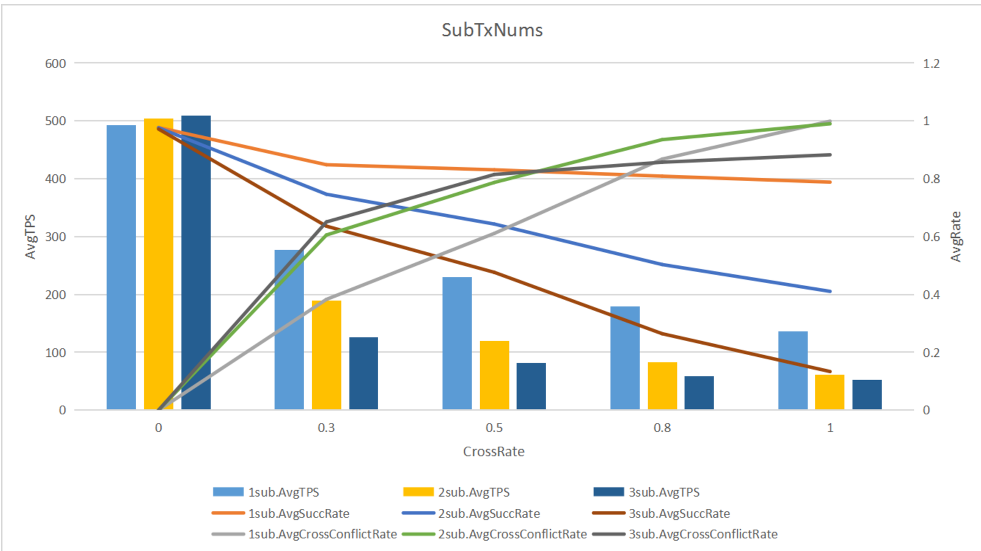

Figure 6. Impact of the number of sub-transactions of cross chains.

According to the experimental results, under the same conditions, more cross-chain transactions led to a lower average TPS, a lower SuccRate, and a higher cross-chain conflict rate. Because more cross-chain transactions represent a collaborative cross-chain transaction and are restricted by more cross-chain transactions, the failure of any sub-transaction will lead to the failure of collaborative cross-chain transactions.

5.5. Comparison of Two Cross-Chain Methods for Single-Span Chain Transactions

In the case of simple single-span chain transactions, we designed a more efficient crosschain-op approach. Next, we compare the performance differences between the two cross-chain approaches for monad transactions.

We performed several experiments to compare MultiCross and crosschain-op with a cross-chain transaction. Both methods set the RWRate to 0, 0.3, 0.5, 0.8, 1, respectively, and experiments were conducted with a CrossRate of 0, 0.3, 0.5, 0.8, and 1. Since there was no cross-chain conflict in crosschain-op, only SuccRate and average TPS are compared. The experimental results are shown in Figure 7.

Figure 7. Comparison of two cross-chain methods.

We can see that the changes of the two cross-chain methods are very different under different parameter changes. The SuccRate of the MultiCross method was not affected by RWRate, but decreased as the CrossRate increased. When the CrossRate was 0, the system SuccRate was between 95% and 100%. As the CrossRate increased, the system SuccRate dropped to about 80%. The crosschain-op SuccRate was not obviously affected by CrossRate but by RWRate. In the optimized crosschain method, SuccRate reached a minimum of 94% when the read/write ratio was balanced. The more unbalanced the read/write ratio was, the closer SuccRate was to 100%. This rule is the same as with traditional fabrics.

The average TPS of the MultiCross method was significantly affected by the CrossRate compared with that of the optimized cross-chain method. The highest mean TPS for the MultiCross method was 510, and the lowest mean TPS was 130. The highest mean TPS of the crosschain-op was 540, and the lowest mean TPS was 510. The average TPS of the MultiCross method decreased with an increasing CrossRate, and the average TPS of the crosschain-op peaked at read-write equalization.

It can be seen that the optimized cross-chain method did make the system better than the MultiCross method in terms of average TPS and SuccRate. However, the crosschain-op supports only one cross-chain transaction, which limits its scope of use. We combined crosschain-op with MultiCross to enable our cross-chainable blockchain system to support the collaborative cross-chain of any number of sub-transactions while providing good performance over cross-chain requests from monadic transactions.

6. Conclusions

In this article, we expand the traditional concept of the cross chain from the scope of "assets" to the logical operation of cross-blockchain, that is cross-chain interoperation. For the purposes of this article, cross-chain interoperability means that one blockchain can make read/write requests to another independent blockchain or invoke the smart contract of this other chain. We designed cross-chain scheduling, which manages and sorts cross-chain transactions and connects isolated federation chains to achieve a blockchain cross chain. This paper introduces a cooperative cross-chain method commonly used for Hyperledger Fabric. A collaborative cross-chain method can support the participation of multiple blockchains and collaborative cross-chain transactions composed of multiple cross-chain transactions. In the collaborative cross-chain approach, we designed a cross-chain flow of EOVPC that is different from the traditional Fabric transaction flow. In order for cross-chain transactions for the cross-chain result consensus to take effect at the fastest speed and prevent cross-chain failure from having an error effect on the subsequent transaction, we propose a cross-chain transaction dependency graph that records a dependence relationship between the local transaction and its predecessor. We also propose an optimized cross-chain method for single cross-chain interoperability. This crosschain-op takes an optimistic cross-chain consensus approach for one-way cross-chain interoperation between two blockchains. Finally, experiments were designed to test the two cross-chain methods at different proportions of cross-chain transactions, read and write transactions, and sub-transactions, and we analyzed the TPS, success rate, and failure ratio of cross-chain transactions of the cross-chain system under different circumstances. The experimental results show that both cross-chain methods can achieve the cross-chain interoperability of the blockchain. The collaborative cross-chain method can be applied to complex multi-chain cooperative cross-chain transactions and can guarantee their atomicity. However, the throughput and transaction success rate of the blockchain will decrease rapidly as the proportion of collaborative cross-chain transactions increases and as the complexity of the cross-chain transactions increases. The crosschain-op method based on the EOVC has an improved throughput and transaction success rate. However, the optimized cross-chain method is more suitable for the case of only one cross-chain transaction. Its application scope is limited.

Author Contributions: Conceptualization, W.W. and Z.Z.; methodology, W.W.; software, W.W.; validation, W.W.; formal analysis, W.W.; investigation, W.W.; resources, W.W.; data curation, W.W.; writing—original draft preparation, W.W.; writing—review and editing, Z.Z., G.W. and Y.Y.; visualization, W.W.; supervision, Z.Z.; project administration, Z.Z.; funding acquisition, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key R&D Program of China Grant No. 2020YFB1707900, NSFC Grant No. 62072035, Open Research Projects of Zhejiang Lab Grant No. 2020KE0AB04 and CCF-Huawei Database System Innovation Research Plan Grant No. CCF-HuaweiDBIR2021007B.

Institutional Review Board Statement: Not applicable. This study is not involving humans or animals.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Belchior, R.; Vasconcelos, A.; Guerreiro, S.; Correia, M. A survey on blockchain interoperability: Past, present, and future trends. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–41. [CrossRef]
2. Belchior, R.; Vasconcelos, A.; Correia, M.; Hardjono, T. Hermes: Fault-tolerant middleware for blockchain interoperability. *Future Gener. Comput. Syst.* **2022**, *129*, 236–251. [CrossRef]
3. Fynn, E.; Bessani, A.; Pedone, F. Smart contracts on the move. In Proceedings of the 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Valencia, Spain, 29 June–2 July 2020; pp. 233–244.
4. Hargreaves, M.; Hardjono, T. Open Digital Asset Protocol. 2020. Available online: https://datatracker.ietf.org/doc/draft-hargreaves-odap/ (accessed on 26 March 2022).
5. Yaga, D.; Mell, P.; Roby, N.; Scarfone, K. Blockchain technology overview. *arXiv* **2019**, arXiv:1906.11078.
6. Abebe, E.; Behl, D.; Govindarajan, C.; Hu, Y.; Karunamoorthy, D.; Novotny, P.; Pandit, V.; Ramakrishna, V.; Vecchiola, C. Enabling enterprise blockchain interoperability with trusted data transfer (industry track). In Proceedings of the 20th International Middleware Conference Industrial Track, Davis, CA, USA, 9–13 December 2019; pp. 29–35.
7. Pillai, B.; Biswas, K.; Muthukkumarasamy, V. Blockchain interoperable digital objects. In *International Conference on Blockchain*; Springer: Cham, Switzerland, 2019; pp. 80–94.
8. Hardjono, T.; Lipton, A.; Pentland, A. Toward an interoperability architecture for blockchain autonomous systems. *IEEE Trans. Eng. Manag.* **2019**, *67*, 1298–1309. [CrossRef]

9. Zamyatin, A.; Al-Bassam, M.; Zindros, D.; Kokoris-Kogias, E.; Moreno-Sanchez, P.; Kiayias, A.; Knottenbelt, W.J. Sok: Communication across distributed ledgers. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 3–36.
10. Asokan, N.; Shoup, V.; Waidner, M. Optimistic fair exchange of digital signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 591–606.
11. Montgomery, H.; Borne-Pon, H.; Hamilton, J.; Bowman, M.; Somogyvari, P.; Fujimoto, S.; Takeuchi, T.; Kuhrt, T.; Belchior, R. Hyperledger Cactus Whitepaper. 2020. Available online: https://github.com/hyperledger/cactus/blob/master/docs/whitepaper/whitepaper.md (accessed on 26 March 2022).
12. Lafourcade, P.; Lombard-Platet, M. About blockchain interoperability. *Inf. Process. Lett.* **2020**, *161*, 105976. [CrossRef]
13. Buterin, V. R3 Report-Chain Interoperability. *R3 Res.* **2016**.
14. Warren, W.; Bandeali, A. 0x: An Open Protocol for Decentralized Exchange on the Ethereum Blockchain. 2017; pp. 4–18. Available online: https://github.com/0xProject/whitepaper (accessed on 26 March 2022).
15. Wiki, B. Hash Time Locked Contracts. 2016. Available online: https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts (accessed on 26 March 2022).
16. Wiki, B. Hashlock. 2016. Available online: https://en.bitcoin.it/wiki/Hashlock (accessed on 26 March 2022).
17. Wiki, B. Timelock. 2016. Available online: https://en.bitcoin.it/wiki/Timelock (accessed on 26 March 2022).
18. Black, M.; Liu, T.; Cai, T. Atomic loans: Cryptocurrency debt instruments. *arXiv* **2019**, arXiv:1901.05117.
19. Decker, C.; Wattenhofer, R. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–18.
20. Herlihy, M. Atomic cross-chain swaps. In Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, Egham, UK, 23–27 July 2018; pp. 245–254.
21. Kumar, R.; Tripathi, R. Content-Based Transaction Access From Distributed Ledger of Blockchain Using Average Hash Technique. In *Opportunities and Challenges for Blockchain Technology in Autonomous Vehicles*; IGI Global: Hershey, PA, USA, 2021; pp. 34–50.
22. Back, A.; Corallo, M.; Dashjr, L.; Friedenbach, M.; Maxwell, G.; Miller, A.; Poelstra, A.; Timón, J.; Wuille, P. Enabling Blockchain Innovations with Pegged Sidechains. 2014; Volume 72. Available online: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains (accessed on 26 March 2022)
23. Gaži, P.; Kiayias, A.; Zindros, D. Proof-of-stake sidechains. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 139–156.
24. Kokoris-Kogias, E.; Jovanovic, P.; Gasser, L.; Gailly, N.; Syta, E.; Ford, B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 583–598.
25. Zamyatin, A.; Stifter, N.; Judmayer, A.; Schindler, P.; Weippl, E.; Knottenbelt, W.J. A wild velvet fork appears! inclusive blockchain protocol changes in practice. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 31–42.
26. Kiayias, A.; Zindros, D. Proof-of-work sidechains. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 21–34.
27. Singh, A.; Click, K.; Parizi, R.M.; Zhang, Q.; Dehghantanha, A.; Choo, K.K.R. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *J. Netw. Comput. Appl.* **2020**, *149*, 102471. [CrossRef]
28. Nakamoto, S.; Bitcoin, A. A Peer-to-Peer Electronic Cash System. 2008; Volume 4. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 26 March 2022).
29. Frauenthaler, P.; Sigwart, M.; Spanring, C.; Schulte, S. Testimonium: A cost-efficient blockchain relay. *arXiv* **2020**, arXiv:2002.12837.