METHODS AND DEVICES FOR PROVIDING ATOMIC TRANSACTION ON BLOCKCHAIN

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for providing an atomic transaction on a blockchain.

BACKGROUND

[0002] Blockchain systems, also known as distributed ledger systems (DLSs) or consensus systems, may enable participating entities to store data securely and immutably. Blockchain systems may include any DLSs, without referencing any particular use case, and may be used for public, private, and consortium blockchain networks. A public blockchain network is open for all entities to use the system and participate in the consensus process. A private blockchain network is provided for a particular entity, which centrally controls read and write permissions. A consortium blockchain network is provided for a select group of entities, which control the consensus process, and includes an access control layer.

[0003] A blockchain system is implemented using a peer-to-peer (P2P) network, in which the nodes communicate directly with each other, e.g., without the need of a fixed, central server. Each node in the P2P network may initiate communication with another node in the P2P network.

[0004] A blockchain system maintains one or more blockchains. A blockchain is a data structure for storing data, such as transactions, that may prevent tampering and manipulation of the data by malicious parties. The blockchain system may

support execution of one or more smart contracts. Each smart contract may be a computer protocol in the form of computer code that is incorporated into a blockchain, to facilitate, verify, or enforce the negotiation or performance of a contract.

[0005] Blockchain use cases may require two transactions to be processed atomically, meaning that the two transactions either both succeed, or both fail, with no partial processing allowed. However, current implementations of blockchain systems do not have built-in mechanisms to support atomic transactions.

[0006] To achieve atomicity, a separate, customized smart contract may need to be deployed to serve as a proxy contract. For example, if a first user wants to exchange a first asset owned by the first user for a second asset owned by a second user, the first user may need to create a first transaction that operates on a first contract to transfer the first asset from the first user to the second user. The first user may then send the first transaction to a proxy contract. Likewise, the second user may need to create a second transaction that operates on a second contract to transfer the second asset from the second user to the first user. The second user may then send the second transaction to the proxy contract. After having received both transactions, the proxy contract may invoke calls to the first contract and the second contract to carry out the two transactions.

[0007] As a number of proxy contracts increases, consumption of storage spaces and computation resources will also increase. The proxy contracts may also need to acquire authorizations to carry out certain operations defined by the first and second contracts, further increasing the number of transactions needed to perform atomic transactions.

SUMMARY

[0008] In one aspect, a computer-implemented method for providing an atomic transaction on a blockchain includes: receiving a transaction submitted to the blockchain; determining whether the transaction is a fulfilling transaction; in response to a determination that the transaction is a fulfilling transaction, identifying an offering transaction that corresponds to the fulfilling transaction; determining whether the offering transaction is valid and whether the fulfilling transaction satisfies an execution condition specified in the offering transaction, the execution condition being specified in a data field of the offering transaction, the execution condition specifying a smart contract and further specifying a method call defined in the smart contract to be invoked by the fulfilling transaction; and in response to a determination that the offering transaction is valid and that the fulfilling transaction satisfies the execution condition specified in the offering transaction, executing the fulfilling transaction and the offering transaction.

[0009] In another aspect, a device for providing an atomic transaction on a blockchain includes: one or more processors; and one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon that are executable by the one or more processors to: receive a transaction submitted to the blockchain; determine whether the transaction is a fulfilling transaction; in response to a determination that the transaction is a fulfilling transaction, identify an offering transaction that corresponds to the fulfilling transaction; determine whether the offering transaction is valid and whether the fulfilling transaction satisfies an execution condition specified in the offering transaction, the execution condition being specified in a data field of the offering transaction, the execution condition specifying a smart contract and further

specifying a method call defined in the smart contract to be invoked by the fulfilling transaction; and in response to a determination that the offering transaction is valid and that the fulfilling transaction satisfies the execution condition specified in the offering transaction, execute the fulfilling transaction and the offering transaction.

[0010] In still another aspect, a non-transitory computer-readable medium has stored therein instructions that, when executed by a processor of a device, cause the device to perform a method for providing an atomic transaction on a blockchain. The method includes: receiving a transaction submitted to the blockchain; determining whether the transaction is a fulfilling transaction; in response to a determination that the transaction is a fulfilling transaction, identifying an offering transaction that corresponds to the fulfilling transaction; determining whether the offering transaction is valid and whether the fulfilling transaction satisfies an execution condition specified in the offering transaction, the execution condition being specified in a data field of the offering transaction, the execution condition specifying a smart contract and further specifying a method call defined in the smart contract to be invoked by the fulfilling transaction; and in response to a determination that the offering transaction is valid and that the fulfilling transaction satisfies the execution condition specified in the offering transaction, executing the fulfilling transaction and the offering transaction.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments. In the following description, which refers to the drawings, the same numbers in different drawings represent the same or similar elements unless otherwise represented.

[0012] FIG. 1 is a schematic diagram of a blockchain system, according to an embodiment.

[0013] FIG. 2 is a schematic diagram of a computing device for implementing a node in a blockchain system, according to an embodiment.

[0014] FIG. 3 is a block diagram depicting a protocol for providing an atomic transaction on a blockchain, according an embodiment.

[0015] FIG. 4 is a flow chart of a method for providing an atomic transaction on a blockchain, according an embodiment.

[0016] FIG. 5 is a block diagram of an apparatus for providing an atomic transaction on a blockchain, according to an embodiment.

DETAILED DESCRIPTION

[0017] Embodiments of the specification provide methods and devices for providing atomic transactions on blockchains. The methods and devices may add certain data fields to a data structure of blockchain transactions to support execution of atomic transactions. Such atomic transactions may be executed atomically on a blockchain without requiring the blockchain to deploy proxy contracts. The methods and devices may also allow the new data fields to be set to nil. Transactions with new data fields set to nil may be executed on the blockchain in a conventional manner, thereby minimizing the impact of the changes to the data structure on conventional transactions.

[0018] Embodiments disclosed in the specification have one or more technical effects. In some embodiments, the methods and devices add certain data fields to the data structure of blockchain transactions. This allows the methods and devices to support execution of atomic transactions on a blockchain without requiring the blockchain to deploy proxy contracts, which in turn reduces consumptions of storage spaces and computation resources. This also allows every contract defined on the blockchain to have the ability to interact with other contracts, regardless of when the contract is deployed and how the contract is implemented. In some embodiments, the methods and devices may process transactions with the new data fields set to nil. This allows the methods and devices to support creations and executions of transactions that do not utilize the atomic feature disclosed herein. This also minimizes the impact of the changes to the data structure on conventional

transactions. In this manner, the methods and devices can be utilized in existing blockchain systems without significant modifications.

[0019] A blockchain is a data structure that stores data, e.g., transactions, in a way that may prevent tampering and manipulation of the data by malicious parties. The transactions stored in this manner may be immutable and subsequently verified. A blockchain includes one or more blocks. Each block is linked to a previous block immediately before it in the blockchain by including a cryptographic hash of the previous block. Each block also may include a timestamp, its own cryptographic hash, and one or more transactions. The transactions, which generally have already been verified by the nodes of the blockchain system, may be hashed and encoded into a data structure, such as a Merkle tree. In a Merkle tree, data at leaf nodes of the tree is hashed, and all hashes in each branch of the tree may be concatenated at a root of the branch. This process continues up the tree to the root of the entire tree, which stores a hash that is representative of all data in the tree. A hash purporting to be of a transaction stored in the tree can be quickly verified by determining whether it is consistent with the structure of the tree.

[0020] A blockchain system includes a network of computing nodes that manage, update, and maintain one or more blockchains. The network may be a public blockchain network, a private blockchain network, or a consortium blockchain network. For example, numerous entities, such as hundreds, thousands, or even millions of entities, can operate in a public blockchain network, and each of the entities operates at least one node in the public blockchain network. Accordingly, the public blockchain network can be considered a public network with respect to the participating entities. Sometimes, a majority of entities (nodes) must sign every block for the block to be valid and added to the blockchain of the blockchain network.

Examples of public blockchain networks include particular peer-to-peer payment networks that leverage a distributed ledger, referred to as blockchain.

[0021] In general, a public blockchain network may support public transactions. A public transaction is shared with all of the nodes in the public blockchain network, and is stored in a global blockchain. A global blockchain is a blockchain replicated across all nodes, and all nodes are in perfect state consensus with respect to the global blockchain. To achieve consensus (e.g., agreement to the addition of a block to a blockchain), a consensus protocol is implemented in the public blockchain network. Examples of consensus protocols include proof-of-work (POW) (e.g., implemented in the some crypto-currency networks), proof-of-stake (POS), and proof-of-authority (POA).

[0022] In general, a private blockchain network may be provided for a particular entity, which centrally controls read and write permissions. The entity controls which nodes are able to participate in the blockchain network. Consequently, private blockchain networks are generally referred to as permissioned networks that place restrictions on who is allowed to participate in the network, and on their level of participation (e.g., only in certain transactions). Various types of access control mechanisms can be used (e.g., existing participants vote on adding new entities, a regulatory authority can control admission).

[0023] In general, a consortium blockchain network may be private among the participating entities. In a consortium blockchain network, the consensus process is controlled by an authorized set of nodes, one or more nodes being operated by a respective entity (e.g., a financial institution, insurance company). For example, a consortium of ten (10) entities (e.g., financial institutions, insurance companies) can operate a consortium blockchain network, each of which operates at least one node

in the consortium blockchain network. Accordingly, the consortium blockchain network can be considered a private network with respect to the participating entities. In some examples, each entity (node) must sign every block in order for the block to be valid, and added to the blockchain. In some examples, at least a sub-set of entities (nodes) (e.g., at least 7 entities) must sign every block in order for the block to be valid, and added to the blockchain.

[0024] FIG. 1 illustrates a schematic diagram of a blockchain system 100, according to an embodiment. Referring to FIG. 1, the blockchain system 100 may include a plurality of nodes, e.g., nodes 102-110, configured to operate on a blockchain 120. The nodes 102-110 may form a network 112, such as a peer-to-peer (P2P) network. Each of the nodes 102-110 may be a computing device, such as a computer or a computer system, configured to store a copy of the blockchain 120, or may be software running on the computing device, such as a process or an application. Each of the nodes 102-110 may have a unique identifier.

[0025] The blockchain 120 may include a growing list of records in the form of data blocks, such as blocks B1-B5 in FIG. 1. Each of the blocks B1-B5 may include a timestamp, a cryptographic hash of a previous block, and data of the present block, which may be transactions such as monetary transactions. For example, as illustrated in FIG. 1, block B5 may include a timestamp, a cryptographic hash of block B4, and transaction data of block B5. Also, for example, a hashing operation may be performed on the previous block to generate the cryptographic hash of the previous block. The hashing operation may convert inputs of various lengths into cryptographic outputs of a fixed length through a hash algorithm, such as SHA-256.

[0026] The nodes 102-110 may be configured to perform an operation on the blockchain 120. For example, when a node, e.g., the node 102, wants to store new

data onto the blockchain 120, that node may generate a new block to be added to the blockchain 120 and broadcast the new block to other nodes, e.g., the nodes 104-110, in the network 112. Based on legitimacy of the new block, e.g., validity of its signature and transactions, the other nodes may determine to accept the new block, such that the node 102 and the other nodes may add the new block to their respective copies of the blockchain 120. As this process repeats, more and more blocks of data may be added to the blockchain 120.

[0027] FIG. 2 illustrates a schematic diagram of a computing device 200 for implementing a node, e.g., the node 102 (FIG. 1), in a blockchain system, according to an embodiment. Referring to FIG. 2, the computing device 200 may include a communication interface 202, a processor 204, and a memory 206.

[0028] The communication interface 202 may facilitate communications between the computing device 200 and devices implementing other nodes, e.g., nodes 104-110 (FIG. 1), in the network. In some embodiments, the communication interface 202 is configured to support one or more communication standards, such as an Internet standard or protocol, an Integrated Services Digital Network (ISDN) standard, etc. In some embodiments, the communication interface 202 may include one or more of a Local Area Network (LAN) card, a cable modem, a satellite modem, a data bus, a cable, a wireless communication channel, a radio-based communication channel, a cellular communication channel, an Internet Protocol (IP) based communication device, or other communication devices for wired and/or wireless communications. In some embodiments, the communication interface 202 may be based on public cloud infrastructure, private cloud infrastructure, hybrid public/private cloud infrastructure.

[0029] The processor 204 may include one or more dedicated processing

units, application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), or various other types of processors or processing units. The processor 204 is coupled with the memory 206 and is configured to execute instructions stored in the memory 206.

[0030] The memory 206 may store processor-executable instructions and data, such as a copy of the blockchain 120 (FIG. 1). The memory 206 may include any type of volatile or non-volatile memory devices, or a combination thereof, such as a static random-access memory (SRAM), an electrically erasable programmable read-only memory (EEPROM), an erasable programmable read-only memory (EPROM), a programmable read-only memory (PROM), a read-only memory (ROM), a magnetic memory, a flash memory, or a magnetic or optical disk. When the instructions in the memory 206 are executed by the processor 204, the computing device 200 may perform an operation on the blockchain 120.

[0031] FIG. 3 illustrates a block diagram depicting a protocol 300 for providing an atomic transaction on a blockchain according to an embodiment. Referring to FIG. 3, multiple users may have accounts on a Blockchain, e.g., the blockchain 120 (FIG. 1). The Blockchain may be implemented to support various types of users, or parties, including, e.g., individuals, businesses, banks, financial institutions, hospitals, as well as other types of companies, organizations, and the like.

[0032] For illustrative purposes, a first user 302 and a second user 304 are depicted in FIG. 3. Two smart contracts are also depicted in FIG. 3, namely, Contract A and Contract B. Smart contracts are computer protocols implemented in the form of computer code that are incorporated into the Blockchain, to facilitate, verify, or enforce the negotiation or performance of contracts. For example, users of the Blockchain may program agreed terms into a smart contract using a

programming language, such as C++, Java, Solidity, Python, etc., and when the terms are met, the smart contract may be automatically executed on the Blockchain, e.g., to perform a transaction. Also for example, the smart contract may include a plurality of subroutines, functions, or methods, each of which may be a sequence of program instructions that performs a specific task. The smart contract may be operational code that is fully or partially executed without human interaction. In some embodiments, a first smart contract, e.g., Contract A, may be incorporated into the Blockchain to manage some assets (e.g., a certain type of digital currency) recorded on the Blockchain. A second smart contract, e.g., Contract B, may also be incorporated into the Blockchain to manage some assets (e.g., a certain type of commodity) recorded on the Blockchain. For illustrative purposes, dashed line rectangles are used to indicate transactions that may invoke one or more methods defined in Contract A and Contract B. It is to be understood that the dashed line rectangles are not indicative of the data structure maintained by the Blockchain.

[0033] Suppose that the first user 302 owns a first asset managed by Contract A and wants to exchange the first asset for a second asset managed by Contract B. In this case, the first user 302 may create a first transaction 306 offering to exchange the first asset for the second asset. The first transaction 306 may be defined using a data structure depicted in FIG. 3, which in turn defines an event that can be carried out on the Blockchain. As will be described in detail below, the first transaction 306 may define the event to be an invocation of a method, e.g., “methodA,” defined in a smart contract, e.g., “Contract A,” if certain execution conditions are satisfied. If the second user 304 owns the second asset and is willing to carry out the exchange with the first user 302, the second user 304 may create a second transaction 308 accepting/fulfilling the offer. The second transaction 308 may be defined using the

same data structure. Both users 302 and 304 may be interested in carrying out the exchange in an atomic manner, meaning that they may be interested in making sure the two transactions 306 and 308 either both succeed, or both fail. To that end, both users 302 and 304 may follow the protocol 300 depicted in FIG. 3 to carry out the first transaction 306 and the second transaction 308.

[0034] In some embodiments, the first user 302 may specify various data fields 306-1 through 306-10 when creating the first transaction 306. The data fields may include a transaction identifier 306-1, a timestamp 306-2, a “from” data field 306-3, a “to” data field 306-4, a timeout parameter 306-5, a status indicator 306-6, a method selector 306-7, an input data filed 306-8, a target transaction identifier 306-9, and one or more execution conditions 306-10.

[0035] In some embodiments, the transaction identifier 306-1 may specify a value, e.g., a hashed string, that uniquely identifies the first transaction 306. The timestamp 306-2 may specify the time when the first transaction 306 is created. The “from” data field 306-3 may identify the transaction sender, which is the first user 302 in this case. The “to” data field 306-4 may identify the transaction receiver, which may be set to Contract A, i.e., the contract that manages the first asset. The timeout parameter 306-5 may specify a time after which the first transaction 306 is set to expire. The status indicator 306-6 may indicate the status of the first transaction 306, including, e.g., “pending,” “success,” “reverted,” or “expired.” The method selector 306-7 may specify a method defined in Contract A that is to be invoked if/when the first transaction 306 is executed. In some embodiments, the method may include a transfer operation provided by Contract A to transfer the first asset from the first user 302 to a specified user (e.g., the second user 304). The input data field 306-8 may specify one or more input parameters to be provided to the method

specified in the method selector 306-7. In some embodiments, the input data field 306-8 may include serialized input data, which may be decoded to function as input parameters to the method when the first transaction 306 is executed.

[0036] The target transaction identifier 306-9 may indicate which transaction is to be executed atomically together with the first transaction 306. The transaction that is to be executed atomically together with the first transaction 306 may be referred to as the “target” transaction of the first transaction 306. In some embodiments, the target transaction identifier 306-9 may be left blank (e.g., set to nil) if the first user 302 does not know which transaction is the target transaction of the first transaction 306. This may happen, for example, when the first user 302 is creating the first transaction 306 as an offering transaction that offers to exchange the first asset for the second asset. At the time of creation of the first transaction 306, the second transaction 308 may not have been created yet, so the target transaction identifier 306-9 of the first transaction 306 may be left blank.

[0037] The execution conditions 306-10 may specify one or more conditions that must be satisfied for the first transaction 306 to execute. In some embodiments, the conditions may include certain requirements established for the target transaction of the first transaction 306. In other words, the target transaction of the first transaction 306 may be required to satisfy the execution conditions 306-10 set forth by the first transaction 306 for the target transaction to execute atomically together with the first transaction 306. Continuing with the example above, where the first user 302 wants to exchange a first asset managed by Contract A for a second asset managed by Contract B, the first user 302 may specify as an execution condition 306-10 that the target transaction of the first transaction 306 must be a transaction that operates on assets managed by Contract B. This execution

condition may be expressed as “targetTx.to == contractB” in FIG. 3. In this manner, if the second user 304 wants to accept the first user 302’s offer to exchange the first asset for the second asset, the second user 304 can create the second transaction 308 with the “to” data field 308-4 set to Contract B. Otherwise, the second transaction 308 may not satisfy the execution condition 306-10 of the first transaction 306.

[0038] The first user 302 may also require the target transaction of the first transaction 306 to invoke a particular method provided by Contract B when the target transaction is executed. For example, in some embodiments, the first user 302 may require the target transaction of the first transaction 306 to invoke a method defined in Contract B that performs a transfer operation when the target transaction is executed. This execution condition may be expressed as “targetTx.method == methodB” in FIG. 3. In this manner, if the second user 304 wants to accept the first user 302’s offer to exchange the first asset for the second asset, the second user 304 can create the second transaction 308 with the method selector 308-7 set to methodB. Otherwise, the second transaction 308 may not satisfy the execution condition 306-10 of the first transaction 306.

[0039] The first user 302 may also specify additional execution conditions 306-10. For example, as depicted in FIG. 3, the first user 302 may require the target transaction of the first transaction 306 to call a specified method, e.g., “methodB,” with a first parameter having a value greater than 10 and a second parameter having a value less than 20. Target values for other parameters may be specified in similar manners.

[0040] In some embodiments, the first user 302 may establish the target values based on the offer provided by the first user 302. For instance, suppose that

the first user 302 offers to exchange one unit of the first asset for at least eleven units of the second assets, and further suppose that methodB takes a parameter, param_B(1), as the total units of the second asset to be transferred, then the first user 302 may specify as an execution condition 306-10 that param_B(1) must be greater than 10. In this manner, if the second user 304 wants to accept the first user 302's offer to exchange the first asset for the second asset, the second user 304 can create the second transaction 308 with parameter param_B(1) set to be greater than 10. Otherwise, the second transaction 308 may not satisfy the execution conditions 306-10 of the first transaction 306.

[0041] It is to be understood that the execution conditions 306-10 depicted above are provided merely as examples and are not meant to be limiting. It is contemplated that other execution conditions 306-10 may be specified in similar manners, and that the target values may include numbers, strings, as well as values expressed in other types of data structures. It is noted that while specific implementations may vary, the purpose of implementing the execution conditions 306-10 may be the same, which is to provide the first user 302 with the ability to specify conditions (e.g., in terms of invoking *which* method of *which* smart contract with *what* parameters) required of the target transaction of the first transaction 306.

[0042] Once the first transaction 306 is created, the first user 302 may submit the first transaction 306 for recordation on the Blockchain. In some embodiments, the first transaction 306 submitted to the Blockchain may be added to a block 310, which is subject to a consensus process. For illustrative purposes, black dots are used to indicate transactions contained in various blocks of the Blockchain (these transactions are contained in the blocks similar to the transactions contained in block B5 of FIG. 1), and the first transaction 306 is depicted as being contained in the

block 310 as one of the transactions. In some embodiments, adding the first transaction 306 to the block 310 may trigger a transaction execution process 316, which may determine, at step 318, that the first transaction 306 is an offering transaction because its target transaction identifier is set to nil and its execution condition is not nil. The transaction execution process 316 may then set the status of the first transaction 306 to “pending” at step 320. Alternatively, in some embodiments, the first transaction 306 submitted to the Blockchain may remain in a pending transaction pool maintained by the Blockchain without being added to any blocks. In some embodiments, the first transaction 306 may remain in the pending transaction pool until it is fulfilled or expired.

[0043] Once the first transaction 306 is recorded on the Blockchain, the first transaction 306 may become visible to other users, including the second user 304, who may be interested in creating the second transaction 308 to accept/fulfill the offer specified in the first transaction 306. The second user 304 may specify various data fields 308-1 through 308-10 when creating the second transaction 308. The data fields may include a transaction identifier 308-1, a timestamp 308-2, a “from” data field 308-3, a “to” data field 308-4, a timeout parameter 308-5, a status indicator 308-6, a method selector 308-7, an input data filed 308-8, a target transaction identifier 308-9, and one or more execution conditions 308-10.

[0044] The transaction identifier 308-1 may specify a value, e.g., a hashed string, that uniquely identifies the transaction 308. The timestamp 308-2 may specify the time when the transaction 308 is created. The “from” data field 308-3 may identify the transaction sender, which is the second user 304 in this case. The “to” data field 308-4 may identify the transaction receiver, which may be set to Contract B, i.e., the contract that manages the second asset. The timeout parameter 308-5

may specify a time after which the second transaction 308 is set to expire. The status indicator 308-6 may indicate the status of the transaction 308, including, e.g., “pending,” “success,” “reverted,” or “expired.” The method selector 308-7 may specify a method defined in Contract B that is to be invoked if/when the second transaction 308 is executed. In some embodiments, the method may include a transfer operation provided by Contract B to transfer the second asset from the second user 304 to a specified user (e.g., the first user 302). The input data field 308-8 may specify one or more input parameters to be provided to the method specified in the method selector 308-7. In some embodiments, the input data field 308-8 may include serialized input data, which may be decoded to function as input parameters to the method when the transaction 308 is executed.

[0045] The target transaction identifier 308-9 may indicate which transaction is to be executed atomically together with the second transaction 308. In this case, the second user 304 may set the target transaction identifier 308-9 to be the transaction identifier 306-1 of the first transaction 306, indicating that it is the first transaction 306 that is to be executed atomically together with the second transaction 308. In this manner, the second transaction 308 may be considered to have been created to fulfill the first transaction 306. The second transaction 308 may therefore be referred to as a fulfilling transaction.

[0046] In some embodiments, the second transaction 308, serving as the fulfilling transaction, may leave the execution conditions 308-10 blank (e.g., set to nil). In such embodiments, the second transaction 308 may accept the execution conditions 306-10 set forth in the first transaction 306. However, for the second transaction 308 to be a valid fulfilling transaction, the second transaction 308 must satisfy the execution conditions 306-10 set forth in the first transaction 306. In some

embodiments, validity of the second transaction 308 may be verified after the second transaction 308 is submitted to the Blockchain. In some embodiments, the second transaction 308 submitted to the Blockchain may be added to a block 312, which is subject to a consensus process.

[0047] In some embodiments, adding the second transaction 308 to the block 312 may trigger the transaction execution process 316, which may determine, at step 318, that the second transaction 308 is a fulfilling transaction because its target transaction identifier is not nil. The transaction execution process 316 may then proceed to step 322.

[0048] At step 322, the transaction execution process 316 may identify the target transaction specified by the target transaction identifier 308-9 of the second transaction 308. Because the transaction execution process 316 has already determined that the second transaction 308 is a fulfilling transaction, the transaction execution process 316 may recognize the target transaction specified by the target transaction identifier 308-9 to be the offering transaction that corresponds to the fulfilling transaction. Continuing with the example above, the target transaction identifier 308-9 of the second transaction 308 may identify the first transaction 306 as the target transaction, allowing the transaction execution process 316 to retrieve the target transaction identifier 308-9 and identify the first transaction 306 as the offering transaction. The transaction execution process 316 may then cause the second transaction 308 to be executed and committed together with the first transaction 306 if (1) the first transaction 306 is still valid and (2) the second transaction 308 satisfies the execution conditions 306-10 set forth in the first transaction 306.

[0049] In some embodiments, the transaction execution process 316 may

determine whether the first transaction 306 is still valid based on the timeout parameter 306-5 of the first transaction 306. For example, the transaction execution process 316 may compute a difference between the current timestamp and the timestamp 306-2 of the first transaction 306 and compare the difference against the timeout parameter 306-5 of the first transaction 306. If the difference is greater than the timeout parameter 306-5, the transaction execution process 316 may determine the first transaction 306 to have expired and therefore no longer valid. Alternatively or additionally, the transaction execution process 316 may determine whether the first transaction 306 is still valid based on the status indicator 306-6 of the first transaction 306. For example, if the transaction execution process 316 determines that the status indicator 306-6 indicates that the first transaction 306 is still "pending," the transaction execution process 316 may determine the first transaction 306 to be still valid. In some embodiments, if the transaction execution process 316 determines that the first transaction 306 is no longer valid, the transaction execution process 316 may revert the second transaction 308 (e.g., by setting the status indicator 308-6 of the second transaction 308 to "reverted") so that any changes made by the second transaction 308 (if any) can be reverted on the Blockchain.

[0050] In some embodiments, if the transaction execution process 316 determines that the first transaction 306 is still valid, the transaction execution process 316 may proceed to determine whether the second transaction 308 satisfies the execution conditions 306-10 set forth in the first transaction 306. Continuing with the example above, the execution conditions 306-10 may require the second transaction 308 to be sent to Contract B and that the second transaction 308 to call methodB when the second transaction 308 is executed. Therefore, if the transaction execution process 316 determines that the "to" data field 308-4 of the second

transaction 308 is not set to Contract B, or the method selector 308-7 of the second transaction 308 is not set to methodB, the transaction execution process 316 may determine that the second transaction 308 does not satisfy the execution conditions 306-10. In some embodiments, the execution conditions 306-10 may further set forth certain requirements for parameters param_B(1) and param_B(2). Therefore, if the transaction execution process 316 determines that the input data field 308-8 of the second transaction 308 is setting the parameters param_B(1) and param_B(2) to values that do not conform to the requirements set forth in the execution conditions 306-10, the transaction execution process 316 may determine that the second transaction 308 does not satisfy the execution conditions 306-10.

[0051] In some embodiments, if the transaction execution process 316 determines that the second transaction 308 does not satisfy the execution conditions 306-10, the transaction execution process 316 may revert the second transaction 308 (e.g., by setting the status indicator 308-6 of the second transaction 308 to “reverted”) so that any changes made by the second transaction 308 can be reverted on the Blockchain. On the other hand, if the transaction execution process 316 determines that the second transaction 308 satisfies the execution conditions 306-10, the transaction execution process 316 may proceed to execute both the first transaction 306 and the second transaction 308. If the first transaction 306 and the second transaction 308 both execute successfully, the transaction execution process 316 may commit the execution of both transactions on the Blockchain (e.g., by incorporating the transactions 306 and 308 into the Blockchain and setting the status indicators 306-6 and 308-6 of the transactions 306 and 308 to “success”). In this manner, the result of the execution may be included in a new block, e.g., block 314, which may then be added to the Blockchain. On the other hand, if either the first

transaction 306 or the second transaction 308 is executed unsuccessfully, the transaction execution process 316 may revert both transactions 306 and 308 (e.g., by setting the status indicators 306-6 and 308-6 of the transactions 306 and 308 to "reverted") so that any changes made by either the first transaction 306 or the second transaction 308 can be reverted on the Blockchain. In this manner, atomicity can be achieved because the two transactions 306 and 308 either both succeed, or both fail, with no partial processing allowed.

[0052] In some embodiments, the transaction execution process 316 may further support executions of transactions that are neither offering transactions nor fulfilling transactions. Such transactions may have target transaction identifiers and execution conditions both set to nil. When the transaction execution process 316 determines that a transaction is neither an offering transaction nor a fulfilling transaction, the transaction execution process 316 may proceed to step 324 and treat the transaction as a "normal" transaction. In some embodiments, the transaction execution process 316 may handle a normal transaction in a conventional manner. For example, the transaction execution process 316 may execute the transaction and update the status of the transaction as the transaction would normally execute on the Blockchain, thereby minimizing the impact of the new data fields (e.g., the target transaction identifier and the execution conditions) have on conventional transactions.

[0053] It is to be understood that while the examples above referenced user-created transactions 306 and 308, such implementations are merely provided as examples and are not meant to be limiting. In some embodiments, the transactions 306 and 308 may be user-created or system-generated. It is also to be understood that while the examples above referenced Contract A and Contract B as two different

smart contracts incorporated into the Blockchain, such implementations are merely provided as examples and are not meant to be limiting. In some embodiments, it may be permissible for the first user 302 to set the target transaction, e.g., targetTx.to, to Contract A, in which case the fulfilling transaction will need to be a transaction that is sent to Contract A as well. Furthermore, it is to be understood that the declarations of the functions, variables, and transactions described above are merely presented as examples and are not meant to be limiting.

[0054] FIG. 4 illustrates a flow chart of a method 400 for providing an atomic transaction on a blockchain, according to an embodiment. The method 400 may be performed by one or more nodes in a blockchain system, e.g., the nodes 102-110 in the blockchain system 100 (FIG. 1). The nodes 102-110 in the blockchain system 100 may perform operations on a blockchain, e.g., the blockchain 120 (FIG. 1). The blockchain 120 may be implemented as the Blockchain in the examples described above.

[0055] At step 402, a node, e.g., the node 102, may receive a transaction submitted to the blockchain 120. At step 404, the node 102 may determine whether the transaction is an offering transaction, a fulfilling transaction, or a normal transaction.

[0056] In some embodiments, the node 102 may determine that the transaction is an offering transaction if the transaction specifies one or more execution conditions but does not specify a target transaction identifier. If the transaction specifies a target transaction identifier, then the node 102 may recognize the transaction as a fulfilling transaction. If the transaction specifies neither a target transaction identifier nor an execution condition, then the node 102 may recognize the transaction as a normal transaction.

[0057] In some embodiments, if the node 102 determines that the transaction is an offering transaction, the node 102 may set the status of the transaction as pending. If the node 102 determines that the transaction is a normal transaction, the node 102 may execute the transaction in a conventional manner. If the node 102 determines that the transaction is a fulfilling transaction, the node 102 may proceed to step 406.

[0058] At step 406, in response to a determination that the transaction is a fulfilling transaction (e.g., the second transaction 308, FIG. 3), the node 102 may identify an offering transaction (e.g., the first transaction 306, FIG. 3) that corresponds to the fulfilling transaction. In some embodiments, the node 102 may identify the offering transaction based on the target transaction identifier (e.g., the target transaction identifier 308-9, FIG. 3) specified in the fulfilling transaction. The node 102 may retrieve the target transaction identifier specified in the fulfilling transaction and identify the offering transaction based on the target transaction identifier.

[0059] At step 408, the node 102 may determine whether the offering transaction is valid and whether the fulfilling transaction satisfies an execution condition specified in the offering transaction.

[0060] In some embodiments, the node 102 may determine whether the offering transaction is valid based on a timeout parameter specified in the offering transaction. For example, the node 102 may compute a difference between the current timestamp and a timestamp recorded when the offering transaction was created. The node 102 may compare the difference against the timeout parameter specified in the offering transaction. If the difference is greater than the timeout parameter, the node 102 may consider the offering transaction to have expired and

therefore no longer valid. Alternatively or additionally, the node 102 may determine whether the offering transaction is valid based on a status indicator specified in the offering transaction. For example, if the node 102 determines that the status indicator specified in the offering transaction indicates that the offering transaction is still pending, the node 102 may determine the offering transaction to be valid. In some embodiments, if the node 102 determines that the offering transaction is no longer valid, the node 102 may revert the fulfilling transaction so that any changes made by the fulfilling transaction can be reverted on the blockchain 120.

[0061] In some embodiments, the offering transaction may specify one or more execution conditions and the node 102 may determine whether the fulfilling transaction satisfies each of the execution conditions specified in the offering transaction. The execution conditions may specify, e.g., a smart contract that the fulfilling transaction should be sent to, a method call that the fulfilling transaction should invoke, or a target value for a parameter that should be contained in the fulfilling transaction. It is to be understood that the target value may be specified as a particular value, a range of values, a minimum value, a maximum value, or the like.

[0062] At step 410, in response to a determination that the offering transaction is valid and that the fulfilling transaction satisfies the execution condition specified in the offering transaction, the node 102 may execute the fulfilling transaction and the offering transaction. In some embodiments, the node 102 may determine whether the fulfilling transaction and the offering transaction are executed successfully. If both the fulfilling transaction and the offering transaction are executed successfully, the node 102 may commit the execution of both the fulfilling transaction and the offering transaction on the blockchain 120. On the other hand, if at least one of the fulfilling transaction and the offering transaction is executed unsuccessfully, the node

102 may revert both the fulfilling transaction and the offering transaction so that any changes made by either the fulfilling transaction or the offering transaction can be reverted on the blockchain 120. In this manner, atomicity can be achieved because the fulfilling transaction and the offering transaction may either both succeed, or both fail, with no partial processing allowed.

[0063] FIG. 5 is a block diagram of an apparatus 500 for providing an atomic transaction on a blockchain, according to an embodiment. The apparatus 500 may be an implementation of a software process, and may correspond to the method 400 (FIG. 4). Referring to FIG. 5, the apparatus 500 may include a receiving module 502, a determination module 504, an identification module 506, and an execution module 508.

[0064] The receiving module 502 may receive a transaction submitted to a blockchain. The receiving module 502 may provide the received transaction to the determining module 504.

[0065] The determining module 504 may determine whether the transaction is an offering transaction, a fulfilling transaction, or a normal transaction. If the transaction is a normal transaction, the determining module 504 may provide the transaction to the execution module 508 for execution. If the transaction is an offering transaction, the determination module 504 may set the status of the transaction as pending. If the transaction is a fulfilling transaction, the determination module 504 may provide the transaction to the identification module 506 for further processing.

[0066] The identification module 506 may identify an offering transaction that corresponds to the fulfilling transaction. In some embodiments, the identification module 506 may identify the offering transaction based on the target transaction

identifier specified in the fulfilling transaction. The identification module 506 may retrieve the target transaction identifier specified in the fulfilling transaction and identify the offering transaction based on the target transaction identifier. The identification module 506 may then provide the identified offering transaction to the determination module 504.

[0067] The determination module 504 may determine whether the offering transaction is valid and whether the fulfilling transaction satisfies an execution condition specified in the offering transaction. In response to a determination that the offering transaction is valid and that the fulfilling transaction satisfies the execution condition specified in the offering transaction, the determination module 504 may provide the offering transaction and the fulfilling transaction to the execution module 508 for execution. In some embodiments, if both the offering transaction and the fulfilling transaction are executed successfully, the execution module 508 may commit the execution of both the offering transaction and the fulfilling transaction on the blockchain. On the other hand, if at least one of the offering transaction and the fulfilling transaction is executed unsuccessfully, the execution module 508 may revert both the offering transaction and the fulfilling transaction so that any changes made by either the offering transaction and the fulfilling transaction can be reverted on the blockchain. In this manner, atomicity can be achieved because the offering transaction and the fulfilling transaction may either both succeed, or both fail, with no partial processing allowed.

[0068] Each of the above described modules may be implemented as software, or hardware, or a combination of software and hardware. For example, each of the above described modules may be implemented using a processor executing instructions stored in a memory. Also, for example, each the above

described modules may be implemented with one or more application specific integrated circuits (ASICs), digital signal processors (DSPs), digital signal processing devices (DSPDs), programmable logic devices (PLDs), field programmable gate arrays (FPGAs), controllers, micro-controllers, microprocessors, or other electronic components, for performing the described methods. Further for example, each of the above described modules may be implemented by using a computer chip or an entity, or implemented by using a product having a certain function. In one embodiment, the apparatus 500 may be a computer, and the computer may be a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email receiving and sending device, a game console, a tablet computer, a wearable device, or any combination of these devices.

[0069] For an implementation process of functions and roles of each module in the apparatus 500, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[0070] In some embodiments, a computer program product may include a non-transitory computer-readable storage medium having computer-readable program instructions thereon for causing a processor to carry out the above-described methods.

[0071] The computer-readable storage medium may be a tangible device that can store instructions for use by an instruction execution device. The computer-readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the

computer-readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing.

[0072] The computer-readable program instructions for carrying out the above-described methods may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or source code or object code written in any combination of one or more programming languages, including an object oriented programming language, and conventional procedural programming languages. The computer-readable program instructions may execute entirely on a computing device as a stand-alone software package, or partly on a first computing device and partly on a second computing device remote from the first computing device. In the latter scenario, the second, remote computing device may be connected to the first computing device through any type of network, including a local area network (LAN) or a wide area network (WAN).

[0073] The computer-readable program instructions may be provided to a processor of a general-purpose or special-purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the above-described methods.

[0074] The flow charts and diagrams in the figures illustrate the architecture,

functionality, and operation of possible implementations of devices, methods, and computer program products according to various embodiments of the specification. In this regard, a block in the flow charts or diagrams may represent a software program, segment, or portion of code, which comprises one or more executable instructions for implementing specific functions. It should also be noted that, in some alternative implementations, the functions noted in the blocks may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the diagrams and/or flow charts, and combinations of blocks in the diagrams and flow charts, may be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

[0075] It is appreciated that certain features of the specification, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the specification, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination or as suitable in any other described embodiment of the specification. Certain features described in the context of various embodiments are not essential features of those embodiments, unless noted as such.

[0076] Although the specification has been described in conjunction with specific embodiments, many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the following claims embrace all such alternatives, modifications and variations that fall within the terms of the claims.

What Is Claimed Is:

1. A computer-implemented method for providing an atomic transaction on a blockchain, the method comprising:

receiving a transaction submitted to the blockchain;

determining whether the transaction is a fulfilling transaction;

in response to a determination that the transaction is a fulfilling transaction, identifying an offering transaction that corresponds to the fulfilling transaction;

determining whether the offering transaction is valid and whether the fulfilling transaction satisfies an execution condition specified in the offering transaction, the execution condition being specified in a data field of the offering transaction, the execution condition specifying a smart contract and further specifying a method call defined in the smart contract to be invoked by the fulfilling transaction; and

in response to a determination that the offering transaction is valid and that the fulfilling transaction satisfies the execution condition specified in the offering transaction, executing the fulfilling transaction and the offering transaction.

2. The method of claim 1, further comprising:

determining whether the transaction is an offering transaction; and

in response to a determination that the transaction is an offering transaction, setting a status of the transaction as pending.

3. The method of claim 1, further comprising:

determining whether the transaction is a normal transaction; and

in response to a determination that the transaction is a normal transaction, executing the transaction.

4. The method of any preceding claim, wherein determining whether the transaction is a fulfilling transaction further comprises:

determining whether the transaction specifies a target transaction identifier; and

in response to a determination that the transaction specifies a target transaction identifier, recognizing the transaction as a fulfilling transaction.

5. The method of claim 4, wherein identifying the offering transaction that corresponds to the fulfilling transaction further comprises:

retrieving the target transaction identifier specified in the fulfilling transaction; and

identifying the offering transaction based on the target transaction identifier.

6. The method of any preceding claim, wherein determining whether the offering transaction is valid further comprises:

determining whether the offering transaction is valid based on a timeout parameter specified in the offering transaction.

7. The method of any preceding claim, wherein determining whether the offering transaction is valid further comprises:

determining whether the offering transaction is valid based on a status indicator specified in the offering transaction.

8. The method of any preceding claim, wherein the execution condition specified in the offering transaction specifies a target value and how the target value compares against a parameter contained in the fulfilling transaction.

9. The method of any preceding claim, further comprising:

determining whether the fulfilling transaction and the offering transaction are executed successfully; and

in response to a determination that both the fulfilling transaction and the offering transaction are executed successfully, committing the execution of both the fulfilling transaction and the offering transaction on the blockchain.

10. The method of claim 9, further comprising:

in response to a determination that at least one of the fulfilling transaction and the offering transaction is executed unsuccessfully, reverting both the fulfilling transaction and the offering transaction.

11. A device for providing an atomic transaction on a blockchain, comprising:

one or more processors; and

one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon that are executable by the one or more processors to perform the method of any of claims 1 to 10.

12. An apparatus for providing atomic transaction on a blockchain, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 10.

13. A non-transitory computer-readable medium having stored therein instructions that, when executed by a processor of a device, cause the device to perform the method of any of claims 1 to 10.