

METHOD AND DEVICE FOR BROADCASTING MESSAGES

TECHNICAL FIELD

[0001] The present application relates to the field of blockchain technologies, and in particular, to a method and device for broadcasting messages.

5

BACKGROUND

[0002] In the field of blockchain technologies, a blockchain node that is responsible for performing consensus verification on a service is referred to as a consensus node. In the trend of blockchain globalization, consensus nodes are often distributed around the world.

10

[0003] Each consensus node usually needs to broadcast messages (for example, a service message or a consensus message) to consensus nodes in a service handling phase, a consensus verification phase, and other phases. A globalized heterogeneous network can cause instability in a message transmission process, for example, the Chinese network and the American network are heterogeneous networks, and a consensus node in the American network may fail to receive a message that is broadcast by a consensus node in the Chinese network. Therefore, to ensure that each other consensus node receives a message at a relatively high success rate, a consensus node that broadcasts the message usually creates one thread for each other consensus node, and executes, by using the thread, a task of sending the message to the another consensus node.

20

[0004] Specifically, after sending the message to the another consensus node by using the thread, the consensus node that broadcasts the message waits to receive a notification (for announcing that the another consensus node has received the message) returned by the another consensus node. If receiving no notification within a specified time by using the thread, the consensus node that broadcasts the message sends the message to the another consensus node again by using the thread, until the consensus node that broadcasts the message receives the notification. Alternatively, if the consensus node that broadcasts the message still receives no notification by using the thread after sending the message for multiple times, the consensus node that

25

broadcasts the message cancels the thread. As such, it can be ensured that the another consensus node receives the message at a relatively high success rate.

5 **[0005]** However, actually, when executing, by using the thread created by the consensus node that broadcasts the message, the task of sending the message to the another consensus node, the consensus node that broadcasts the message usually needs to repeatedly send the message for multiple times. It consumes a large number of computing resources of the consensus node that broadcasts the message, causing an excessive burden on the consensus node that broadcasts the message.

SUMMARY

10 **[0006]** Implementations of the present application provide a method and device for broadcasting messages, to alleviate a problem that the existing method for broadcasting messages causes an excessive burden on a consensus node that broadcasts a message.

15 **[0007]** To alleviate the previous technical problem, the implementations of the present application are implemented as follows:

20 **[0008]** An implementation of the present application provides a method for broadcasting messages, including: obtaining, by a consensus node of a blockchain network, a first broadcast message; creating a first thread for a message processing device corresponding to the consensus node, and sending the first broadcast message to the message processing device by using the first thread, so that the message processing device sends the first broadcast message to a message processing device corresponding to another consensus node; and cancelling the first thread.

25 **[0009]** An implementation of the present application provides another method for broadcasting messages, including: receiving, by a message processing device, a first broadcast message sent by a consensus node corresponding to the message processing device; creating a second thread for a message processing device corresponding to each other consensus node, and sending the first broadcast message to the message processing device corresponding to the another consensus node by using the second thread; and cancelling the second thread in response to determining that the message processing device corresponding to the another consensus node receives the first broadcast message.

30 **[0010]** An implementation of the present application provides a device for

broadcasting messages, including: an acquisition module, configured to obtain a first broadcast message; a sending module, configured to create a first thread for a message processing device corresponding to the device, and send the first broadcast message to the message processing device by using the first thread, so that the message processing device sends the first broadcast message to a message processing device corresponding to another consensus node; and a cancellation module, configured to cancel the first thread.

[0011] An implementation of the present application provides another device for broadcasting messages, including: a first receiving module, configured to receive a first broadcast message sent by a consensus node corresponding to the device; a sending module, configured to create a second thread for a message processing device corresponding to each other consensus node, and send the first broadcast message to the message processing device corresponding to the another consensus node by using the second thread; and a cancellation module, configured to cancel the second thread in response to determining that the message processing device corresponding to the another consensus node receives the first broadcast message.

[0012] It can be learned from the previous technical solutions provided in the implementations of the present application that, in the implementations of the present application, one message processing device is allocated to each consensus node, and a work of broadcasting a message between consensus nodes is handed over to message processing devices for implementation. For a certain consensus node, a message processing device corresponding to the consensus node can create a thread for a message processing device corresponding to another consensus node, to execute a message broadcasting task, until the message processing device corresponding to the another consensus node receives a broadcast message. As such, the thread for executing the message broadcasting task consumes only a computing resource of the message processing device without consuming a resource of the consensus node. Therefore, no burden is caused on the consensus node, so that the consensus node can perform consensus verification on a service more efficiently.

BRIEF DESCRIPTION OF DRAWINGS

[0013] To describe the technical solutions in the implementations of the present application or in the existing technology more clearly, the following briefly describes

the accompanying drawings needed for describing the implementations or the existing technology. Apparently, the accompanying drawings in the following description merely show some implementations of the present application, and a person of ordinary skill in the art can still derive other drawings from these accompanying drawings without creative efforts.

[0014] FIG. 1 is a flowchart illustrating a method for broadcasting messages, according to an implementation of the present application;

[0015] FIG. 2 is a schematic diagram illustrating an example of comparing the existing method for broadcasting messages with a method for broadcasting messages provided in the present application;

[0016] FIG. 3 is a schematic diagram illustrating a device for broadcasting messages, according to an implementation of the present application; and

[0017] FIG. 4 is a schematic diagram illustrating another device for broadcasting messages, according to an implementation of the present application.

DESCRIPTION OF IMPLEMENTATIONS

[0018] Implementations of the present application provide a method and device for broadcasting messages.

[0019] To make a person skilled in the art understand the technical solutions in the present application better, the following clearly and comprehensively describes the technical solutions in the implementations of the present application with reference to the accompanying drawings in the implementations of the present application. Apparently, the described implementations are merely some but not all of the implementations of the present application. All other implementations obtained by a person of ordinary skill in the art based on the implementations of the present application without creative efforts shall fall within the protection scope of the present application.

[0020] The technical solutions provided in the implementations of the present application are described in detail below with reference to the accompanying drawings.

[0021] FIG. 1 is a flowchart illustrating a method for broadcasting messages, according to an implementation of the present application. The method for broadcasting messages includes the following steps.

[0022] S101. A consensus node obtains a first broadcast message.

[0023] In this implementation of the present application, the consensus node is a blockchain node that participates in consensus verification on a service in a blockchain network. In a typical blockchain network, all blockchain nodes can participate in consensus verification as consensus nodes. However, in a special blockchain network (for example, a blockchain network that includes a consortium-chain subnetwork and a public-chain subnetwork), not all blockchain nodes can participate in consensus verification as consensus nodes.

[0024] It is an application scenario of the implementations of the present application that in a service handling phase, a consensus verification phase, and other phases, consensus nodes need to broadcast or send messages to each other.

[0025] The service handling phase means that a certain consensus node receives a service message, and broadcasts the service message to other consensus nodes, so that all consensus nodes receive the service message. It facilitates subsequent consensus verification on the service message. The consensus verification phase is a phase in which all consensus nodes comply with a certain consensus algorithm to reach a consensus on a verification result. In the consensus verification phase, all the consensus nodes need to send verification messages to each other, to reach a consensus.

[0026] In this implementation of the present application, the first broadcast message can be the service message, the verification message, or a message that is broadcast or sent by a certain consensus node to another consensus node.

[0027] It is worthwhile to note that the method can be performed by any consensus node. In the service processing phase, the consensus node can receive a service message sent by a client or a non-consensus node, and use the service message as the obtained first broadcast message. In the consensus verification phase, the consensus node can generate a verification message after verifying the service message, and use the verification message as the obtained first broadcast message.

[0028] S102. The consensus node creates a first thread for a message processing device corresponding to the consensus node, and sends the first broadcast message to the message processing device by using the first thread.

[0029] S103. The consensus node cancels the first thread.

[0030] In this implementation of the present application, the consensus node can send the first broadcast message to the message processing device corresponding to

the consensus node, and completely hand a task of broadcasting or sending the first broadcast message over to the message processing device corresponding to the consensus node for processing, so that the message processing device ensures that a message processing device of another consensus node successfully receives the first broadcast message. As such, the consensus node does not need to consume computing resources to ensure that all other consensus nodes receive the first broadcast message as many as possible.

[0031] FIG. 2 is a schematic diagram illustrating an example of comparing the existing method for broadcasting messages with a method for broadcasting messages provided in the present application. As shown in FIG. 2, the message processing device corresponding to the consensus node is a computing device independent of the consensus node. Message exchange between consensus nodes is performed by message processing devices respectively corresponding to the consensus nodes.

[0032] That is, the consensus node only needs to create the first thread for the message processing device corresponding to the consensus node, and can cancel the first thread after sending the first broadcast message to the message processing device corresponding to the consensus node by using the first thread, without consuming a computing resource to continuously maintain the first thread.

[0033] The first thread is a thread created by the consensus node for the message processing device corresponding to the consensus node, and the second thread described below is a thread created by the message processing device for the message processing device corresponding to the another consensus node.

[0034] Actually, the consensus node uses an asynchronous call technology to create the first thread that is used to send the first broadcast message to the message processing device corresponding to the consensus node. After sending the first broadcast message by using the first thread, the consensus node can directly cancel the first thread without caring whether the message processing device receives the first broadcast message.

[0035] It is worthwhile to note that the consensus node and the message processing device corresponding to the consensus node are usually not distributed in a heterogeneous network. That is, information exchange between the consensus node and the message processing device corresponding to the consensus node is stable. Therefore, the message processing device usually can successfully receive, at one time, a message sent by the consensus node corresponding to the message processing

device.

[0036] Further, to completely ensure that the first broadcast message is not lost in a process of transmitting the first broadcast message from the consensus node to the corresponding message processing device, a regional private line can be used to
 5 connect the consensus node to the message processing device corresponding to the consensus node, to alleviate, as much as possible, network disturbance that may exist between the consensus node and the message processing device corresponding to the consensus node.

[0037] In addition to instructing the message processing device corresponding to
 10 the consensus node to broadcast the message, the consensus node can further receive, by using the message processing device corresponding to the consensus node, a message that is broadcast by the another consensus node.

[0038] Specifically, the consensus node can receive a second broadcast message sent by the message processing device corresponding to the consensus node. The
 15 second broadcast message is broadcast by the message processing device corresponding to the another consensus node. The message processing device can receive the second broadcast message that is broadcast by the message processing device corresponding to the another consensus node, and then send the second broadcast message to the consensus node.

[0039] It is worthwhile to note that both the first broadcast message and the
 20 second broadcast message are broadcast messages that are broadcast by consensus nodes, and both the first broadcast message and the second broadcast message can be service messages, verification messages, etc.

[0040] S104. The message processing device creates a second thread for a
 25 message processing device corresponding to each other consensus node, and sends the first broadcast message to the message processing device corresponding to the another consensus node by using the second thread.

[0041] S105. The message processing device cancels the second thread in
 30 response to determining that the message processing device corresponding to the another consensus node receives the first broadcast message.

[0042] In the existing technology, the consensus node is responsible for message broadcasting. The consensus node creates a thread that occupies a computing resource of the consensus node, and uses the thread to ensure, as many as possible, that all other consensus nodes receive a message that is broadcast by the consensus node.

[0043] In this implementation of the present application, the consensus node only needs to send the first broadcast message to the message processing device corresponding to the consensus node. As a computing device independent of the consensus node, the message processing device can be independently responsible for message broadcasting. As such, the consensus node is released, so that the consensus node can concentrate limited computing resources to perform consensus verification work more efficiently.

[0044] In this implementation of the present application, the message processing device creates the second thread for the message processing device corresponding to each other consensus node, and uses the second thread to consume a computing resource of the message processing device to wait to confirm that the message processing device corresponding to each other consensus node successfully receives the first broadcast message.

[0045] Specifically, the message processing device can send the first broadcast message to the message processing device corresponding to each other consensus node by using the second thread created for the message processing device corresponding to each other consensus node; count the number of received response signals, where the response signal is returned by the message processing device corresponding to the another consensus node after receiving the first broadcast message; and resend the first broadcast message by using each second thread in response to determining that the counted number is less than a specific value, until the counted number of received response signals is not less than the specific value, where the specific value is an integer greater than 0. The specific value is configurable. A higher specific value can ensure that more consensus nodes receive the first broadcast message, and therefore the consensus node broadcasts the message at a higher success rate. The specific value is not specifically limited in the present application.

[0046] Further, if the counted number is less than the specific value, the message processing device can continue sending, by using a second thread corresponding to a message processing device that returns no response signal, the first broadcast message to the message processing device that returns no response signal, until the total number of received response signals counted in previous times is not less than the specific value. Alternatively, the message processing device can send the first broadcast message to all other message processing devices again by using all second threads, until the number of received response signals counted at a next time is not

less than the specific value.

[0047] Further, after the message processing device sends the first broadcast message to another certain message processing device for a specified number of times by using a second thread, if the message processing device still receives, by using the
 5 second thread, no response signal returned by the another message processing device, the message processing device can stop sending the first broadcast message to the another message processing device. After a specified period, the message processing device continues sending the first broadcast message to the another message processing device.

[0048] For example, the message processing device can first broadcast the first broadcast message to other message processing devices, then determine another message processing device that returns no response signal (that is, a message processing device that fails to receive the first broadcast message), and then resend the first broadcast message to the message processing device that fails to receive the
 15 first broadcast message. These operations are repeated. After the message processing device repeatedly sends the first broadcast message to the same another message processing device for three times, it indicates that there is a temporary transmission problem to be resolved (the problem may be caused due to a failure of the message processing device that fails to receive the first broadcast message or due to a large
 20 network fault). Therefore, the message processing device first stops sending the first broadcast message to the another message processing device, and then continues sending the first broadcast message to the another message processing device after waiting for a long time (the transmission problem may have been resolved).

[0049] In this implementation of the present application, the message processing
 25 device cancels the second thread only after ensuring that the message processing device corresponding to each other consensus node receives the first broadcast message. It is worthwhile to note that the message processing device can use various message delivery policies such as delay delivery and compensation delivery (a subject that fails to receive the message actively requests to compensate for the message) to
 30 ensure that the message processing device corresponding to each other consensus node receives the first broadcast message. A message delivery policy used by the message processing device is not specifically limited in the present application.

[0050] It can be learned from the previous descriptions that, the message processing device actually uses a synchronization call technology to create, for the

message processing device corresponding to each other consensus node, one second thread that is used to send the first broadcast message to the message processing device corresponding to each other consensus node. After sending the first broadcast message by using the second thread, the message processing device still uses the second thread to wait to confirm that the message processing device corresponding to each other consensus node receives the first broadcast message. If the message processing device corresponding to each other consensus node fails to receive the first broadcast message, the message processing device resends the first broadcast message to the message processing device corresponding to each other consensus node by using the second thread. The second thread is cancelled only until the message processing device corresponding to each other consensus node certainly receives the first broadcast message or a condition for stopping sending is triggered.

[0051] S106. The message processing device corresponding to the another consensus node sends the first broadcast message to corresponding another consensus node.

[0052] As previously described, the another consensus node can receive, by using the message processing device corresponding to the another consensus node, a message that is broadcast by another consensus node.

[0053] According to the method for broadcasting messages shown in FIG. 1, one message processing device is allocated to each consensus node, and a work of broadcasting a message between consensus nodes is handed over to message processing devices for implementation. For a certain consensus node, a message processing device corresponding to the consensus node can create a thread for a message processing device corresponding to each other consensus node, to execute a message broadcasting task, until the message processing device corresponding to the another consensus node receives a broadcast message. As such, the thread for executing the message broadcasting task consumes only a computing resource of the message processing device without consuming a resource of the consensus node. Therefore, no burden is caused on the consensus node, so that the consensus node can perform consensus verification on a service more efficiently.

[0054] Based on the method for broadcasting messages shown in FIG. 1, an implementation of the present application further correspondingly provides a device for broadcasting messages. As shown in FIG. 3, the device for broadcasting messages includes: an acquisition module 301, configured to obtain a first broadcast message; a

sending module 302, configured to create a first thread for a message processing device corresponding to the device, and send the first broadcast message to the message processing device by using the first thread, so that the message processing device sends the first broadcast message to a message processing device corresponding to another consensus node; and a cancellation module 303, configured to cancel the first thread.

[0055] The device is connected to the message processing device by using a regional private line.

[0056] The device further includes a receiving module 304, configured to receive a second broadcast message sent by the message processing device corresponding to the device, where the second broadcast message is broadcast by the message processing device corresponding to the another consensus node.

[0057] Based on the method for broadcasting messages shown in FIG. 1, an implementation of the present application further correspondingly provides another device for broadcasting messages. As shown in FIG. 4, the device for broadcasting messages includes: a first receiving module 401, configured to receive a first broadcast message sent by a consensus node corresponding to the device; a sending module 402, configured to create a second thread for a message processing device corresponding to each other consensus node, and send the first broadcast message to the message processing device corresponding to the another consensus node by using the second thread; and a cancellation module 403, configured to cancel the second thread in response to determining that the message processing device corresponding to the another consensus node receives the first broadcast message.

[0058] The cancellation module 403 is configured to count the number of received response signals by using each second thread, where the response signal is returned by the message processing device corresponding to the another consensus node after receiving the first broadcast message; and resend the first broadcast message by using each second thread in response to determining that the counted number is less than a specific value, until the counted number of received response signals is not less than the specific value, where the specific value is an integer greater than 0.

[0059] The device further includes a second receiving module 404, configured to receive a second broadcast message that is broadcast by the message processing device corresponding to the another consensus node; and send the second broadcast

message to the consensus node.

[0060] In the 1990s, whether a technical improvement is a hardware improvement (for example, an improvement to a circuit structure such as a diode, a transistor, or a switch) or a software improvement (an improvement to a method procedure) can be clearly distinguished. However, as technologies develop, current improvements to many method procedures can be considered as direct improvements to hardware circuit structures. Almost all designers program an improved method procedure into a hardware circuit, to obtain a corresponding hardware circuit structure. Therefore, a method procedure can be improved by using a hardware entity module. For example, a programmable logic device (PLD) (for example, a field programmable gate array (FPGA)) is such an integrated circuit, and a logical function of the PLD is determined by a user through device programming. The designer performs programming to "integrate" a digital system into a PLD without requesting a chip manufacturer to design and manufacture an application-specific integrated circuit chip. In addition, at present, instead of manually manufacturing an integrated chip, this type of programming is mostly implemented by using "logic compiler" software. The "logic compiler" software is similar to a software compiler used to develop and write a program. Original code needs to be written in a particular programming language before being compiled. The language is referred to as a hardware description language (HDL). There are many HDLs, such as Advanced Boolean Expression Language (ABEL), Altera Hardware Description Language (AHDL), Confluence, Cornell University Programming Language (CUPL), HDCal, Java Hardware Description Language (JHDL), Lava, Lola, MyHDL, PALASM, and Ruby Hardware Description Language (RHDL). The very-high-speed integrated circuit hardware description language (VHDL) and Verilog are most commonly used at present. A person skilled in the art should also understand that a hardware circuit that implements a logical method procedure can be easily obtained provided that the method procedure is logically programmed and programmed into an integrated circuit by using the foregoing several hardware description languages.

[0061] A controller can be implemented in any appropriate way. For example, the controller can be in a form of a microprocessor, a processor, a computer readable medium that stores computer readable program code (such as software or firmware) that can be executed by the microprocessor or the processor, a logic gate, a switch, an application-specific integrated circuit (ASIC), a programmable logic controller, or a

built-in microprocessor. Examples of the controller include but are not limited to the following microprocessors: ARC 625D, Atmel AT91SAM, Microchip PIC18F26K20, and Silicone Labs C8051F320. A memory controller can alternatively be implemented as a part of control logic of a memory. A person skilled in the art also knows that, in addition to implementing the controller by using the computer readable program code, method steps can be logically programmed to enable the controller to implement the same function in the form of the logic gate, the switch, the application-specific integrated circuit, the programmable logic controller, or the built-in microcontroller. Therefore, the controller can be considered as a hardware component, and a device that is included in the controller and configured to implement various functions can also be considered as a structure in the hardware component. Alternatively, the device configured to implement various functions can even be considered as both a software module for implementing the method and a structure in the hardware component.

[0062] The system, device, module, or unit illustrated in the previous implementations can be specifically implemented by using a computer chip or an entity, or by using a product having a certain function. A typical implementation device is a computer. Specifically, the computer can be, for example, 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 device, a game console, a tablet computer, a wearable device, or a combination of any of these devices.

[0063] For ease of description, the previous device is described by dividing the device into various units based on functions. Certainly, when the present application is implemented, functions of the units can be implemented in one or more pieces of software and/or hardware.

[0064] A person skilled in the art should understand that the implementations of the present disclosure can be provided as a method, a system, or a computer program product. Therefore, the present disclosure can use a form of hardware only implementations, software only implementations, or implementations with a combination of software and hardware. In addition, the present disclosure can use a form of a computer program product that is implemented on one or more computer-usable storage media (including but not limited to a magnetic disk storage, a CD-ROM, an optical memory, etc.) that include computer-usable program code.

[0065] The present disclosure is described with reference to the flowcharts and/or

block diagrams of the method, the device (system), and the computer program product according to the implementations of the present disclosure. It is worthwhile to note that computer program instructions can be used to implement each process and/or each block in the flowcharts and/or the block diagrams and a combination of processes and/or blocks in the flowcharts and/or the block diagrams. These computer program instructions can be provided for a general-purpose computer, a dedicated computer, an embedded processor, or a processor of another programmable data processing device to generate a machine, so that the instructions executed by the computer or the processor of the another programmable data processing device generate a device for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0066] These computer program instructions can alternatively be stored in a computer readable memory that can instruct a computer or another programmable data processing device to work in a specific way, so that the instructions stored in the computer readable memory generate an artifact that includes an instruction device. The instruction device implements a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0067] These computer program instructions can alternatively be loaded onto a computer or another programmable data processing device, so that a series of operations and steps are performed on the computer or the another programmable device, thereby generating computer-implemented processing. Therefore, the instructions executed on the computer or the another programmable device provide steps for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0068] In a typical configuration, a computing device includes one or more processors (CPUs), one or more input/output interfaces, one or more network interfaces, and one or more memories.

[0069] The memory can include a non-persistent memory, a random access memory (RAM), a nonvolatile memory, and/or another form of memory in computer readable media, for example, a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of the computer readable medium.

[0070] The computer readable medium includes persistent, non-persistent, movable, and unmovable media that can store information by using any method or technology. The information can be a computer readable instruction, a data structure,

a program module, or other data. Examples of a computer storage medium include but are not limited to a parameter random access memory (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), another type of random access memory (RAM), a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or another optical storage, a magnetic cassette, a magnetic tape, a magnetic disk memory or another magnetic storage device, or any other non-transmission medium that can be used to store information that is accessible to the computing device. Based on the definition in this specification, the computer readable medium does not include transitory computer readable media (transitory media) such as a modulated data signal and carrier.

[0071] It is worthwhile to further note that, the terms "include", "comprise", or their any other variants are intended to cover a non-exclusive inclusion, so that a process, a method, a product, or a device that includes a list of elements not only includes those elements but also includes other elements that are not expressly listed, or further includes elements inherent to such process, method, product, or device. Without more constraints, an element preceded by "includes a ..." does not preclude the existence of additional identical elements in the process, method, product, or device that includes the element.

[0072] A person skilled in the art should understand that the implementations of the present application can be provided as a method, a system, or a computer program product. Therefore, the present application can use a form of hardware only implementations, software only implementations, or implementations with a combination of software and hardware. In addition, the present application can use a form of a computer program product that is implemented on one or more computer-usable storage media (including but not limited to a magnetic disk storage, a CD-ROM, an optical memory, etc.) that include computer-usable program code.

[0073] The present application can be described in the general context of computer executable instructions executed by a computer, for example, a program module. Generally, the program module includes a routine, a program, an object, a component, a data structure, etc. for executing a specific task or implementing a specific abstract data type. The present application can alternatively be practiced in distributed computing environments. In these distributed computing environments,

tasks are performed by remote processing devices that are connected through a communications network. In the distributed computing environments, the program module can be located in both local and remote computer storage media including storage devices.

- 5 **[0074]** The implementations in the present specification are described in a progressive way. For same or similar parts of the implementations, mutual references can be made to the implementations. Each implementation focuses on a difference from other implementations. Particularly, a system implementation is basically similar to a method implementation, and therefore is described briefly. For related parts,
10 references can be made to some descriptions in the method implementation.

- [0075]** The previous descriptions are merely implementations of the present application, and are not intended to limit the present application. A person skilled in the art can make various modifications and changes to the present application. Any modification, equivalent replacement, improvement, etc. made without departing from
15 the spirit and principle of the present application shall fall within the scope of the claims in the present application.

CLAIMS

1. A method for broadcasting messages, the method comprising:
 - obtaining, by a first consensus node of a blockchain network, a first broadcast message;
 - creating, by the first consensus node, a first thread for a first message processing device corresponding to the first consensus node;
 - sending, by the first consensus node, the first broadcast message to the first message processing device by using the first thread;
 - cancelling, by the first consensus node, the first thread directly after sending the first broadcast message, without determining whether the first message processing device receives the first broadcast message; and
 - receiving, by the first consensus node, a second broadcast message sent by the first message processing device, wherein the second broadcast message was broadcast by a message processing device corresponding to another consensus node.
2. The method according to claim 1, wherein the first consensus node is connected to the first message processing device by using a regional private line to ensure that the first broadcast message is not lost in a process of transmitting the first broadcast message from the first consensus node to the first message processing device.
3. The method according to claim 1, comprising:
 - counting, by the first message processing device, a number of received response signals by using each second thread, wherein the received response signals are returned by respective other message processing device corresponding to the respective thread after receiving the first broadcast message; and
 - resending, by the first message processing device, the first broadcast message by using each second thread in response to determining that the number of the received response signals is less than a specific value, until a counted number of received response signals is not less than the specific value, wherein the specific value is an integer greater than 0.
4. The method according to claim 1, wherein the first consensus node creates the first thread using an asynchronous call technology.

5. The method according to claim 1, wherein the blockchain network comprises a consortium-chain subnetwork and a public-chain subnetwork and not all blockchain nodes can participate in consensus verification as consensus nodes.

6. The method according to claim 1, wherein the blockchain network comprises the first consensus node and other consensus nodes, each of the other consensus nodes having a respective corresponding other message processing device, all message processing devices being computing devices independent of their corresponding consensus nodes, wherein message exchange between consensus nodes is performed by message processing devices respectively corresponding to the consensus node.

7. The method according to claim 6, further comprising:

sending, by the first message processing device, the first broadcast message to other message processing devices corresponding to the other consensus nodes by creating a respective second thread for each of the other message processing devices and sending the first broadcast message to the other message processing devices by using the respective second threads; and

for each of one or more second threads, determining that the other message processing device to which the first broadcast message was sent received the first broadcast message, and in response to the determining, cancelling the second thread by the first message processing device.

8. A consensus node and a message processing device for broadcasting messages, comprising a plurality of modules configured to perform the method of any one of claims 1 to 7.