

DATA PROCESSING METHOD AND APPARATUS, AND DISTRIBUTED FIREWALL SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

5 **[0001]** This application claims priorities to Chinese Patent Application No. 202410210863.5, filed with the China National Intellectual Property Administration on February 26, 2024 and entitled "DATA PROCESSING METHOD AND APPARATUS, AND OTHER DEVICES", and to Chinese Patent Application No. 202410387020.2, filed with the China National Intellectual Property Administration on March 29, 2024 and entitled "DATA PROCESSING METHOD AND APPARATUS, AND DISTRIBUTED FIREWALL SYSTEM", both of which are incorporated
10 herein by reference in their entireties.

TECHNICAL FIELD

[0002] This application relates to the field of cloud computing technologies, and in particular, to a data processing method and apparatus, and a distributed firewall system.

BACKGROUND

15 **[0003]** A firewall system is a system including computer hardware and software and deployed at a network border, and serves as a bridge between an internal network and an external network. The firewall system can separate the internal network from the external network and perform access control on data at the network border to prevent unauthorized access and malicious intrusion, thereby ensuring data security of the internal network.

20 **[0004]** The firewall system may include a load balancing node and a plurality of engine nodes. The load balancing node is mainly responsible for balancing traffic among the plurality of engine nodes, and the engine nodes mainly perform access control on packets according to security policies. Currently, the engine nodes mainly perform access control on sessions based on session-related information (for example, a session list). When the firewall system is scaled out or in, or
25 an engine node is faulty, some sessions may be redistributed to other engine nodes. Because the other engine nodes do not have information about these sessions, the sessions may be interrupted, affecting normal service transmission.

SUMMARY

[0005] Embodiments of this application provide a data processing method and apparatus, and a distributed firewall system, to reduce impact on service transmission caused by scale-out or scale-in of the firewall system or a fault of an engine node.

5 **[0006]** According to a first aspect, this application provides a data processing method. The method may be performed by a first engine node in a distributed firewall system, or may be performed by a device including the first engine node, or may be performed by a component (for example, a chip system, a chip, or a functional module) configured in the first engine node. This is not limited. The distributed firewall system may include a plurality of engine nodes, and the first
10 engine node belongs to the plurality of engine nodes.

[0007] An example in which the method is performed by the first engine node is used. The method includes: The first engine node creates a first bitmap file for a second engine node, where the first bitmap file indicates a reception status of each message in at least one message (for example, denoted as M messages, where M is a positive integer) of the second engine node by the
15 first engine node, the M messages include one or more of the following: a first session table or stream reassembly-related first information, and the second engine node belongs to the plurality of engine nodes. When the first bitmap file indicates that N messages in the M messages have not been received, the first engine node sends a request message to the second engine node, where the request message is used to request the N messages, and N is a positive integer less than or equal
20 to M. The first engine node receives the N messages sent by the second engine node. The first engine node modifies the first bitmap file, where a modified first bitmap file indicates that the M messages have been received.

[0008] In the foregoing embodiment of this application, the first engine node can obtain the first session table (and/or the stream reassembly-related first information) from the second engine
25 node, which means that the plurality of engine nodes in the distributed firewall system can share the session table (and/or the stream reassembly-related information). In this way, when the distributed firewall system is scaled in (that is, a quantity of engine nodes is reduced) or an engine node is faulty, because another engine node shares a session table (and/or stream reassembly-related information) of the engine node, the another engine node can continue to serve a session
30 (and/or a data stream) for which the engine node is responsible, thereby reducing impact on a service caused by scale-in of the distributed firewall system or a fault of the engine node. When the distributed firewall system is scaled out (that is, a new engine node is added to the distributed firewall system), according to this embodiment of this application, the engine node can obtain historical data (that is, a session table and/or stream reassembly-related information) of another

engine node in the distributed firewall system. In this way, even if a service task of the another engine node is distributed to the new engine node, the new engine node can continue to serve the service based on the historical data, thereby reducing impact on the service during scale-out of the distributed firewall system.

5 **[0009]** In a possible implementation, the first engine node may further store at least one message to be sent (for example, denoted as H messages), where the H messages include one or more of the following: a second session table or stream reassembly-related second information, and H is a positive integer; and send the H messages to at least one engine node other than the first engine node in the plurality of engine nodes.

10 **[0010]** In the foregoing implementation, the first engine node can store one copy of the to-be-sent message for local use, and share another copy of the message with another engine node in the distributed firewall system, to implement data synchronization (or sharing) between the plurality of engine nodes in the distributed firewall system, thereby reducing impact on service transmission.

15 **[0011]** In a possible implementation, that the first engine node sends the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes may be that the first engine node writes the H messages into a buffer, and send the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes when a first condition is met, where the first condition includes one or more of the following: A data volume of the H messages is greater than or equal to a first threshold, or a buffering duration of the H messages is greater than or equal to a second threshold.

20 **[0012]** In the foregoing implementation, the first engine node may first store the to-be-sent message into the buffer, and then send the message to another engine node in the distributed firewall system after the condition is met.

25 **[0013]** In a possible implementation, the first engine node further receives a first packet, where the first packet belongs to a first session; and when the first session is a new session, determines, based on a connection direction of the first session and an access control list, whether to allow access to the first packet.

30 **[0014]** In the foregoing implementation, if the first session to which the first packet belongs is a new session, the first engine node can directly obtain the connection direction of the first session, and query the access control list based on the connection direction of the first session.

35 **[0015]** In a possible implementation, the first engine node may further modify the first session table when the first session table does not include an identifier of the first session, or the first session table includes an identifier of the first session and third information, with the third information indicating that the connection direction of the first session is indeterminate, where a modified first session table includes the identifier of the first session, fourth information, and the

connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; and send the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes.

[0016] In the foregoing implementation, the first engine node can further modify the first session table, and synchronize the modified first session table to another engine node in the distributed firewall system, to implement data synchronization, thereby reducing impact on service transmission.

[0017] In a possible implementation, the first engine node may further receive a first packet, where the first packet belongs to a first session; when the first session is not a new session, determine a connection direction of the first session based on the first session table, where the first session table does not include an identifier of the first session, and the connection direction of the first session is a sending direction of the first packet; determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet; modify the first session table, where a modified first session table includes the identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate; and send the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes.

[0018] In the foregoing implementation, when the first session is not a new session, the first engine node can query the first session table, determine the connection direction of the first session based on a query result, and query the access control list based on the connection direction of the first session. When the first session table does not include the first session, the first engine node can further modify the first session table, and synchronize the modified first session table to another engine node in the distributed firewall system, to implement data synchronization, thereby reducing impact on service transmission.

[0019] In a possible implementation, the first engine node may further receive a first packet, where the first packet belongs to a first session; when the first session is not a new session, determine a connection direction of the first session based on the first session table; and determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet, where the first session table includes an identifier of the first session, fourth information, and the connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; or the first session table includes an identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate, where the connection direction of the first session includes a forward direction and a reverse direction of the first packet. The forward direction and the reverse direction of the first packet may include a sending direction of

the first packet and a direction opposite (or reverse) to the sending direction of the first packet.

[0020] In the foregoing implementation, when the first session is not a new session, the first engine node can query the first session table, determine the connection direction of the first session based on a query result, and query the access control list based on the connection direction of the first session.

[0021] In a possible implementation, the first engine node may further receive a second packet, where the second packet belongs to a first data stream; perform stream reassembly processing on the second packet when the second packet is a 1st packet in the first data stream; and send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, where the fifth information indicates that reassembly of the first data stream has been completed; or send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, where the sixth information indicates that reassembly of the first data stream has not been completed and a main processing node of the first data stream is the first engine node.

[0022] In the foregoing implementation, the first engine node is a main processing node, and can perform stream reassembly processing on a received packet and send reassembly-related information of the first data stream to another engine node in the distributed firewall system, so that the another engine node obtains the reassembly-related information of the first data stream and performs corresponding processing. For example, after reassembly of the first data stream has been completed, another engine node that receives the packet of the first data stream may discard the packet, to release a storage resource. For another example, if reassembly of the first data stream has not been completed and the main processing node is the first engine node, another engine node that receives the packet of the first data stream may send the packet to the first engine node, to complete reassembly of the first data stream as early as possible.

[0023] In a possible implementation, the first information does not include information about a first data stream, and the first engine node may further receive a second packet, where the second packet belongs to the first data stream; when the second packet is not a 1st packet in the first data stream, determine, based on the first information, that reassembly of the first data stream has not been completed and a main processing node of the first data stream does not exist; and buffer the second packet.

[0024] In the foregoing implementation, the first engine node is a main processing node, and can buffer the second packet, so that the second packet can be sent to the main processing node within a buffer validity period for stream reassembly.

[0025] In a possible implementation, the first engine node may further receive seventh

information, where the seventh information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is a third engine node, and the third engine node belongs to the plurality of engine nodes; and send the second packet to the third engine node.

5 **[0026]** In another possible implementation, the first engine node may further receive a third packet, where the third packet belongs to the first data stream; perform stream reassembly processing on the second packet and the third packet when the third packet is the 1st packet in the first data stream; and send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been
10 completed, where the fifth information indicates that reassembly of the first data stream has been completed; or send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, where the sixth information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is the first engine node.

15 **[0027]** In the foregoing two implementations, within the buffer validity period, if the first engine node determines that the main processing node is another engine node (namely, a third engine node), the first engine node sends the second packet to the third engine node; or if the first engine node determines that the main processing node is itself, the first engine node performs stream reassembly on the third packet and a packet in the buffer.

20 **[0028]** In a possible implementation, the first information indicates that reassembly of a first data stream has not been completed and a main processing node of the first data stream is a third engine node, the first engine node may further receive a second packet, where the second packet belongs to the first data stream; and send the second packet to the third engine node based on the first information.

25 **[0029]** In the foregoing manner, the first information may also indicate that the main processing node is the third engine node. In this way, the first engine node can directly send the second packet to the third engine node.

30 **[0030]** According to a second aspect, this application provides a data processing method, and the method may be performed by a firewall system. The firewall system includes a plurality of engine nodes, and the plurality of engine nodes include a first engine node and a second engine node. The method may include:

35 **[0031]** The second engine node stores M messages to be sent, and sends the M messages to at least one engine node other than the second engine node in the plurality of engine nodes, where the M messages include one or more of the following: a first session table or stream reassembly-related first information, and M is a positive integer.

[0032] The first engine node creates a first bitmap file for the second engine node, and sends a request message to the second engine node when the first bitmap file indicates that N messages in the M messages have not been received, where the first bitmap file indicates a reception status of each message in the M messages by the first engine node, the request message is used to request the N messages, and N is a positive integer less than or equal to M.

[0033] The second engine node receives the request message from the first engine node and sends the N messages to the first engine node.

[0034] The first engine node receives the N messages from the second engine node and modifies the first bitmap file, where a modified first bitmap file indicates that the M messages are already received.

[0035] In a possible implementation, the second engine node writes the M messages into a buffer when sending the M messages to the at least one engine node other than the second engine node in the plurality of engine nodes; and sends the M messages to the at least one engine node other than the second engine node in the plurality of engine nodes when a second condition is met, where the second condition includes one or more of the following: A data volume of the M messages is greater than or equal to a third threshold, or a buffering duration of the M messages is greater than or equal to a fourth threshold.

[0036] In a possible implementation, the first engine node may further store H messages to be sent, where the H messages include one or more of the following: a second session table or stream reassembly-related second information, and H is a positive integer; and send the H messages to at least one engine node other than the first engine node in the plurality of engine nodes. The at least one engine node other than the first engine node in the plurality of engine nodes may further receive the H messages.

[0037] In a possible implementation, the first engine node may further store H messages to be sent, where the H messages include one or more of the following: a second session table or stream reassembly-related second information, and H is a positive integer; and send the H messages to at least one engine node other than the first engine node in the plurality of engine nodes.

[0038] In a possible implementation, the first engine node may write the H messages into a buffer when sending the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes, and send the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes when a first condition is met, where the first condition includes one or more of the following: A data volume of the H messages is greater than or equal to a first threshold, or a buffering duration of the H messages is greater than or equal to a second threshold.

[0039] In a possible implementation, the first engine node further receives a first packet, where

the first packet belongs to a first session; and when the first session is a new session, determines, based on a connection direction of the first session and an access control list, whether to allow access to the first packet.

[0040] In a possible implementation, the first engine node may further modify the first session table when the first session table does not include an identifier of the first session, or the first session table includes an identifier of the first session and third information, with the third information indicating that the connection direction of the first session is indeterminate, where a modified first session table includes the identifier of the first session, fourth information, and the connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; and send the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes. The at least one engine node other than the first engine node in the plurality of engine nodes may further receive the modified first session table.

[0041] In a possible implementation, the first engine node may further receive a first packet, where the first packet belongs to a first session; when the first session is not a new session, determine a connection direction of the first session based on the first session table, where the first session table does not include an identifier of the first session, and the connection direction of the first session is a sending direction of the first packet; determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet; modify the first session table, where a modified first session table includes the identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate; and send the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes. The at least one engine node other than the first engine node in the plurality of engine nodes may further receive the modified first session table.

[0042] In a possible implementation, the first engine node may further receive a first packet, where the first packet belongs to a first session; when the first session is not a new session, determine a connection direction of the first session based on the first session table; and determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet, where the first session table includes an identifier of the first session, fourth information, and the connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; or the first session table includes an identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate, where the connection direction of the first session includes a forward direction and a reverse direction of the first packet. The

forward direction and the reverse direction of the first packet may include a sending direction of the first packet and a direction opposite (or reverse) to the sending direction of the first packet.

[0043] In a possible implementation, the first engine node may further receive a second packet, where the second packet belongs to a first data stream; perform stream reassembly processing on the second packet when the second packet is a 1st packet in the first data stream; and send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, where the fifth information indicates that reassembly of the first data stream has been completed; or send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, where the sixth information indicates that reassembly of the first data stream has not been completed and a main processing node of the first data stream is the first engine node. The at least one engine node other than the first engine node in the plurality of engine nodes may further receive the fifth information or the sixth information.

[0044] In a possible implementation, the first information does not include information about a first data stream, and the first engine node may further receive a second packet, where the second packet belongs to the first data stream; when the second packet is not a 1st packet in the first data stream, determine, based on the first information, that reassembly of the first data stream has not been completed and a main processing node of the first data stream does not exist; and buffer the second packet.

[0045] In a possible implementation, the plurality of engine nodes further include a third engine node, the third engine node may send seventh information to at least one engine node other than the third engine node in the plurality of engine nodes, and the seventh information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is the third engine node. The first engine node receives the seventh information and sends the second packet to the third engine node. The third engine node receives the second packet and performs stream reassembly on the second packet.

[0046] In another possible implementation, the first engine node may further receive a third packet, where the third packet belongs to the first data stream; perform stream reassembly processing on the second packet and the third packet when the third packet is the 1st packet in the first data stream; and send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, where the fifth information indicates that reassembly of the first data stream has been completed; or send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been

completed, where the sixth information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is the first engine node. The at least one engine node other than the first engine node in the plurality of engine nodes may further receive the fifth information or the sixth information.

5 **[0047]** In a possible implementation, the plurality of engine nodes further include a third engine node, and the first information indicates that reassembly of a first data stream has not been completed and a main processing node of the first data stream is the third engine node. The first engine node may further receive a second packet, where the second packet belongs to the first data stream; and send the second packet to the third engine node based on the first information. The
10 third engine node may receive the second packet and perform stream reassembly on the second packet.

[0048] According to a third aspect, this application provides a data processing apparatus, where the data processing apparatus is used in a first engine node in a distributed firewall system, the distributed firewall system includes a plurality of engine nodes, and the first engine node
15 belongs to the plurality of engine nodes. For example, the data processing apparatus may perform the method according to the first aspect and any one of the possible implementations of the first aspect. The data processing apparatus may include a processing module, a sending module, and a receiving module. The processing module may also be referred to as a processing unit, a logic unit, or the like. The sending module may also be referred to as a sending unit. The receiving module
20 may also be referred to as a receiving unit. The sending module and the receiving module may also be collectively referred to as a transceiver module (or a transceiver unit), a communication module (or a communication unit), an interface circuit, or the like.

[0049] The processing module is configured to create a first bitmap file for a second engine node, where the first bitmap file indicates a reception status of each message in M messages of the
25 second engine node by the first engine node, the M messages include one or more of the following: a first session table or stream reassembly-related first information, the second engine node belongs to the plurality of engine nodes, and M is a positive integer.

[0050] The sending module is configured to send a request message to the second engine node when the first bitmap file indicates that N messages in the M messages have not been received,
30 where the request message is used to request the N messages, and N is a positive integer less than or equal to M.

[0051] The receiving module is configured to receive the N messages sent by the second engine node.

[0052] The processing module is further configured to modify the first bitmap file, where a
35 modified first bitmap file indicates that the M messages are already received.

[0053] In a possible implementation, the processing module is further configured to store H messages to be sent, where the H messages include one or more of the following: a second session table or stream reassembly-related second information, and H is a positive integer; and the sending module is further configured to send the H messages to at least one engine node other than the first engine node in the plurality of engine nodes.

[0054] In a possible implementation, the processing module is configured to write the H messages into a buffer when the H messages are sent to the at least one engine node other than the first engine node in the plurality of engine nodes; and the sending module is configured to send the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes when a first condition is met, where the first condition includes one or more of the following: A data volume of the H messages is greater than or equal to a first threshold, or a buffering duration of the H messages is greater than or equal to a second threshold.

[0055] In a possible implementation, the receiving module is further configured to receive a first packet, where the first packet belongs to a first session; and the processing module is further configured to: when the first session is a new session, determine, based on a connection direction of the first session and an access control list, whether to allow access to the first packet.

[0056] In a possible implementation, the processing module is further configured to modify the first session table when the first session table does not include an identifier of the first session, or the first session table includes an identifier of the first session and third information, with the third information indicating that the connection direction of the first session is indeterminate, where a modified first session table includes the identifier of the first session, fourth information, and the connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; and the sending module is further configured to send the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes.

[0057] In a possible implementation, the receiving module is further configured to receive a first packet, where the first packet belongs to a first session; the processing module is further configured to: when the first session is not a new session, determine a connection direction of the first session based on the first session table, where the first session table does not include an identifier of the first session, and the connection direction of the first session is a sending direction of the first packet; determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet; and modify the first session table, where a modified first session table includes the identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate; and the sending module is further configured to send the modified first session table to the at least

one engine node other than the first engine node in the plurality of engine nodes.

[0058] In a possible implementation, the receiving module is further configured to receive a first packet, where the first packet belongs to a first session; and the processing module is further configured to: when the first session is not a new session, determine a connection direction of the first session based on the first session table, and determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet, where the first session table includes an identifier of the first session, fourth information, and the connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; or the first session table includes an identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate, where the connection direction of the first session includes a forward direction and a reverse direction of the first packet. The forward direction and the reverse direction of the first packet may include a sending direction of the first packet and a direction opposite (or reverse) to the sending direction of the first packet.

[0059] In a possible implementation, the receiving module is further configured to receive a second packet, where the second packet belongs to a first data stream; the processing module is further configured to perform stream reassembly processing on the second packet when the second packet is a 1st packet in the first data stream; and the sending module is further configured to send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, where the fifth information indicates that reassembly of the first data stream has been completed; or the sending module is further configured to send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, where the sixth information indicates that reassembly of the first data stream has not been completed and a main processing node of the first data stream is the first engine node.

[0060] In a possible implementation, the data processing apparatus may further include a storage module, where the first information does not include information about a first data stream; the receiving module is further configured to receive a second packet, where the second packet belongs to the first data stream; the processing module is further configured to: when the second packet is not a 1st packet in the first data stream, determine, based on the first information, that reassembly of the first data stream has not been completed and a main processing node of the first data stream does not exist; and the storage module is configured to buffer the second packet.

[0061] In a possible implementation, the receiving module is configured to receive seventh information, where the seventh information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is a third engine node,

and the third engine node belongs to the plurality of engine nodes; and the sending module is further configured to send the second packet to the third engine node.

[0062] In another possible implementation, the receiving module is further configured to receive a third packet, where the third packet belongs to the first data stream; the processing module is further configured to perform stream reassembly processing on the second packet and the third packet when the third packet is the 1st packet in the first data stream; and the sending module is further configured to send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, where the fifth information indicates that reassembly of the first data stream has been completed; or the sending module is further configured to send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, where the sixth information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is the first engine node.

[0063] In a possible implementation, the first information indicates that reassembly of a first data stream has not been completed and a main processing node of the first data stream is a third engine node; the receiving module is further configured to receive a second packet, where the second packet belongs to the first data stream; and the sending module is further configured to send the second packet to the third engine node based on the first information.

[0064] According to a fourth aspect, this application provides a distributed firewall system, where the distributed firewall system includes a plurality of engine nodes, and the plurality of engine nodes include a first engine node and a second engine node.

[0065] The second engine node is configured to: store M messages to be sent, and send the M messages to at least one engine node other than the second engine node in the plurality of engine nodes, where the M messages include one or more of the following: a first session table or stream reassembly-related first information, and M is a positive integer.

[0066] The first engine node is configured to: create a first bitmap file for the second engine node, and send a request message to the second engine node when the first bitmap file indicates that N messages in the M messages have not been received, where the first bitmap file indicates a reception status of each message in the M messages by the first engine node, the request message is used to request the N messages, and N is a positive integer less than or equal to M.

[0067] The second engine node is further configured to: receive the request message from the first engine node, and send the N messages to the first engine node.

[0068] The first engine node is further configured to: receive the N messages from the second engine node, and modify the first bitmap file, where a modified first bitmap file indicates that the

M messages are already received.

[0069] In a possible implementation, the second engine node is configured to: write the M messages into a buffer when sending the M messages to the at least one engine node other than the second engine node in the plurality of engine nodes; and send the M messages to the at least one engine node other than the second engine node in the plurality of engine nodes when a second condition is met, where the second condition includes one or more of the following: A data volume of the M messages is greater than or equal to a third threshold, or a buffering duration of the M messages is greater than or equal to a fourth threshold.

[0070] In a possible implementation, the first engine node is further configured to: store H messages to be sent, where the H messages include one or more of the following: a second session table or stream reassembly-related second information, and H is a positive integer; and send the H messages to at least one engine node other than the first engine node in the plurality of engine nodes. The at least one engine node other than the first engine node in the plurality of engine nodes is further configured to receive the H messages.

[0071] In a possible implementation, the first engine node is further configured to: store H messages to be sent, where the H messages include one or more of the following: a second session table or stream reassembly-related second information, and H is a positive integer; and send the H messages to at least one engine node other than the first engine node in the plurality of engine nodes.

[0072] In a possible implementation, the first engine node is configured to: write the H messages into a buffer when sending the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes, and send the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes when a first condition is met, where the first condition includes one or more of the following: A data volume of the H messages is greater than or equal to a first threshold, or a buffering duration of the H messages is greater than or equal to a second threshold.

[0073] In a possible implementation, the first engine node is further configured to: receive a first packet, where the first packet belongs to a first session; and when the first session is a new session, determine, based on a connection direction of the first session and an access control list, whether to allow access to the first packet.

[0074] In a possible implementation, the first engine node is further configured to: modify the first session table when the first session table does not include an identifier of the first session, or the first session table includes an identifier of the first session and third information, with the third information indicating that the connection direction of the first session is indeterminate, where a modified first session table includes the identifier of the first session, fourth information, and the

connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; and send the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes. The at least one engine node other than the first engine node in the plurality of engine nodes is further configured to receive the modified first session table.

[0075] In a possible implementation, the first engine node is further configured to: receive a first packet, where the first packet belongs to a first session; when the first session is not a new session, determine a connection direction of the first session based on the first session table, where the first session table does not include an identifier of the first session, and the connection direction of the first session is a sending direction of the first packet; determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet; modify the first session table, where a modified first session table includes the identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate; and send the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes. The at least one engine node other than the first engine node in the plurality of engine nodes is further configured to receive the modified first session table.

[0076] In a possible implementation, the first engine node is further configured to: receive a first packet, where the first packet belongs to a first session; when the first session is not a new session, determine a connection direction of the first session based on the first session table; and determine, based on the connection direction of the first session and an access control list, whether to allow access to the first packet, where the first session table includes an identifier of the first session, fourth information, and the connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; or the first session table includes an identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate, where the connection direction of the first session includes a forward direction and a reverse direction of the first packet. The forward direction and the reverse direction of the first packet may include a sending direction of the first packet and a direction opposite (or reverse) to the sending direction of the first packet.

[0077] In a possible implementation, the first engine node is further configured to: receive a second packet, where the second packet belongs to a first data stream; perform stream reassembly processing on the second packet when the second packet is a 1st packet in the first data stream; and send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, where the fifth

information indicates that reassembly of the first data stream has been completed; or send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, where the sixth information indicates that reassembly of the first data stream has not been completed and a main processing node of the first data stream is the first engine node. The at least one engine node other than the first engine node in the plurality of engine nodes is further configured to receive the fifth information or the sixth information.

[0078] In a possible implementation, the first information does not include information about a first data stream, and the first engine node is further configured to: receive a second packet, where the second packet belongs to the first data stream; when the second packet is not a 1st packet in the first data stream, determine, based on the first information, that reassembly of the first data stream has not been completed and a main processing node of the first data stream does not exist; and buffer the second packet.

[0079] In a possible implementation, the plurality of engine nodes further include a third engine node, the third engine node is configured to send seventh information to at least one engine node other than the third engine node in the plurality of engine nodes, and the seventh information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is the third engine node. The first engine node is further configured to: receive the seventh information and send the second packet to the third engine node. The third engine node is further configured to: receive the second packet and perform stream reassembly on the second packet.

[0080] In another possible implementation, the first engine node is further configured to: receive a third packet, where the third packet belongs to the first data stream; perform stream reassembly processing on the second packet and the third packet when the third packet is the 1st packet in the first data stream; and send fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, where the fifth information indicates that reassembly of the first data stream has been completed; or send sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, where the sixth information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is the first engine node. The at least one engine node other than the first engine node in the plurality of engine nodes is further configured to receive the fifth information or the sixth information.

[0081] In a possible implementation, the plurality of engine nodes further include a third engine node, and the first information indicates that reassembly of a first data stream has not been

completed and a main processing node of the first data stream is the third engine node. The first engine node is further configured to: receive a second packet, where the second packet belongs to the first data stream; and send the second packet to the third engine node based on the first information. The third engine node is further configured to: receive the second packet and perform stream reassembly on the second packet.

[0082] According to a fifth aspect, this application provides a data processing apparatus, where the data processing apparatus includes a processor, and the processor is configured to: read and execute a computer program or instructions stored in a memory, to enable the compute device to perform the method according to the first aspect and any one of the possible implementations of the first aspect. Optionally, the data processing apparatus may further include the memory, and the memory stores the computer program or instructions.

[0083] According to a sixth aspect, this application provides a compute device, where the compute device includes at least one processor and a communication interface, and the at least one processor is configured to: read and execute instructions through the communication interface, to enable the compute device to perform the method according to the first aspect and any one of the possible implementations of the first aspect.

[0084] According to a seventh aspect, this application further provides a compute device cluster, including at least one compute device, where each compute device includes a processor and a memory, a processor of the at least one compute device is configured to execute instructions stored in a memory of the at least one compute device, to enable the compute device cluster to perform the method according to the second aspect and any one of the possible implementations of the second aspect.

[0085] According to an eighth aspect, this application further provides a chip system, where the chip system includes at least one processor, configured to execute one or more computer programs or instructions in a memory, to implement the method according to the first aspect or the second aspect and any one of the possible implementations of the first aspect or the second aspect. Optionally, the at least one processor is coupled to the memory.

[0086] According to a ninth aspect, this application further provides a computer program product, including computer-executable instructions, where the computer-executable instructions are used to enable a compute device to perform the method according to the first aspect or the second aspect and any one of the possible implementations of the first aspect or the second aspect.

[0087] According to a tenth aspect, this application further provides a computer-readable storage medium, where the computer-readable storage medium stores computer-executable instructions, and the computer-executable instructions are used to enable a compute device to perform the method according to the first aspect or the second aspect and any one of the possible

implementations of the first aspect or the second aspect.

[0088] For technical effects that can be achieved in the second aspect to the tenth aspect and any one of the possible implementations of the second aspect to the tenth aspect, refer to descriptions of beneficial effects in the first aspect and any one of the possible implementations of the first aspect. Details are not described herein again.

BRIEF DESCRIPTION OF DRAWINGS

[0089] FIG. 1a is a diagram of a structure of a distributed firewall system according to an embodiment of this application;

[0090] FIG. 1b is a diagram of a structure of another distributed firewall system according to an embodiment of this application;

[0091] FIG. 2 is a schematic flowchart of a first data processing method according to an embodiment of this application;

[0092] FIG. 3 is a diagram of data sharing according to an embodiment of this application;

[0093] FIG. 4A and FIG. 4B are a schematic flowchart of a second data processing method according to an embodiment of this application;

[0094] FIG. 5A and FIG. 5B are a schematic flowchart of a third data processing method according to an embodiment of this application;

[0095] FIG. 6 is a diagram of a structure of a data processing apparatus according to an embodiment of this application;

[0096] FIG. 7 is a diagram of a structure of a compute device according to an embodiment of this application;

[0097] FIG. 8 is a diagram of a structure of a compute device cluster according to an embodiment of this application; and

[0098] FIG. 9 is a diagram of a structure of another compute device cluster according to an embodiment of this application.

DESCRIPTION OF EMBODIMENTS

[0099] Embodiments of this application are presented with respect to a system including a plurality of devices, components, modules, or the like. It should be understood that the system may include other devices, components, modules, or the like that are not mentioned, or may include only a part of devices, components, modules, or the like that are mentioned in embodiments.

[00100] In embodiments of this application, "a plurality of" may mean two or more. In view of this, in embodiments of this application, "a plurality of" may also be understood as "at least two".

"At least one" may be understood as one or more, for example, one, two, or more. For example, "including at least one" means including one, two, or more. For example, including at least one of A, B, and C means that A, B, C, A and B, A and C, B and C, or A, B, and C may be included. "And/or" describes an association relationship between associated objects and three relationships may specifically exist. For example, A and/or B may represent the following three cases: Only A exists, both A and B exist, and only B exists. In addition, the character "/" generally indicates an "or" relationship between the associated objects.

[00101] Ordinal numerals such as "first" and "second" in embodiments of this application are usually for distinguishing between different objects, and are not for limiting a sequence, a time sequence, priorities, importance degrees, or the like of a plurality of objects. For example, a first engine node and a second engine node in embodiments of this application are used to distinguish between different engine nodes, and are not for limiting a sequence, a time sequence, priorities, importance degrees, or the like of the two engine nodes.

[00102] Embodiments of this application may be applied to a firewall system. The firewall system may include a load balancing node and a plurality of engine nodes. The load balancing node is mainly responsible for balancing traffic among the plurality of engine nodes, and the engine nodes mainly control access to a packet according to a security policy. Currently, the engine nodes included in the firewall system mainly control access to a session based on session-related information (for example, a session list). When the firewall system is scaled out or in, or an engine node is faulty, some sessions may be redistributed to another engine node. Because the another engine node does not have information about these sessions, the sessions may be interrupted, affecting service transmission.

[00103] In view of this, embodiments of this application provide a data processing method and apparatus, and a distributed firewall system, to reduce impact on services caused by scale-out or scale-in of the firewall system or a fault of an engine node. The method and the apparatus in this application are based on the same technical concept. Because problem-resolving principles of the method and the apparatus are similar, mutual reference may be made to implementation of the apparatus and the method, and repeated parts are not described.

[00104] Before the data processing method provided in embodiments of this application is described, the following first describes a distributed firewall system provided in embodiments of this application.

[00105] FIG. 1a and FIG. 1b each are a diagram of an architecture of a distributed firewall system (which may also be referred to as a firewall system or a firewall for short) according to an embodiment of this application. As shown in FIG. 1a or FIG. 1b, the distributed firewall system is deployed at a network border, and serves as a bridge between an internal network and an external

network (for example, the Internet (Internet)). The distributed firewall system can separate the internal network from the external network and control access to data at the network border to prevent unauthorized access and malicious intrusion, thereby ensuring data security of the internal network. As shown in FIG. 1a or FIG. 1b, the distributed firewall system may include a plurality of engine nodes. The plurality of engine nodes may communicate with each other. A quantity of engine nodes included in the distributed firewall system is not limited in embodiments of this application. FIG. 1a or FIG. 1b illustrates a first engine node, a second engine node, and a third engine node as examples.

[00106] The engine node may also be referred to as an end node, a node, or the like. A name of the engine node is not limited in embodiments of this application. The engine node mainly controls access to a packet according to a security policy (for example, a session list or an access control list (access control list, ACL)). For example, the engine node may determine, according to the security policy, whether to allow access to the packet; and if, according to the security policy, the engine node determines to allow access to the packet, the engine node forwards the packet; or if, according to the security policy, the engine node determines not to allow access to the packet, the engine node discards the packet. Functions of the engine node may be implemented by a computer device, a server, a cloud server, a physical machine, a virtual machine, or the like. Functions and implementation forms of the engine node are not limited in embodiments of this application.

[00107] Optionally, the distributed firewall system may further include a load balancing node, which is represented by dashed lines in FIG. 1a and FIG. 1b. The load balancing node may communicate with the plurality of engine nodes.

[00108] The load balancing node may also be referred to as a load balancing device. A name of the load balancing node is not limited in embodiments of this application. The load balancing node is mainly responsible for distributing service data and balancing traffic among the plurality of engine nodes. Functions of the load balancing node may be implemented by a computer device, a server, a cloud server, a physical machine, a virtual machine, or the like. Functions and implementation forms of the load balancing node are not limited in embodiments of this application.

[00109] Optionally, the distributed firewall system may further include a control node, which is not shown in FIG. 1a and FIG. 1b. The control node may communicate with the plurality of engine nodes. The control node may further communicate with the load balancing node. The control node may also be referred to as a control center or the like. A name of the control node is not limited in embodiments of this application. The control node is mainly responsible for delivering configuration information to the plurality of engine nodes and/or the load balancing node, for example, configuring the access control list (access control list, ACL) and a load

balancing policy. Functions of the control node may be implemented by a computer device, a server, a cloud server, a physical machine, a virtual machine, or the like. Functions and implementation forms of the control node are not limited in embodiments of this application.

[00110] The packet is a unit for data switching and transmission in a network, namely, a data block that a station sends at a time. The packet in embodiments of this application may be a packet based on the transmission control protocol/internet protocol (transmission control protocol/internet protocol, TCP/IP). For example, the packet may be a TCP packet, an IP packet, or an internet control message protocol (internet control message protocol, ICMP) packet. This is not limited. When data that needs to be transmitted is large, a packet segmentation technology may be used to divide the data into a plurality of smaller data segments for transmission, to improve transmission efficiency. Data division behavior occurring at a transport layer using TCP may be referred to as TCP segmentation. Data division behavior occurring at a network IP layer using IP may be referred to as IP fragmentation. Both TCP segmentation and IP fragmentation can obtain a complete data stream through stream reassembly. Stream reassembly means combining packets of the same TCP or IP stream in sequence to obtain an application-layer data stream through parsing.

[00111] A session is a time period from the initiation to the termination of a network connection, and includes the entire process from the initiation of a connection request by a session initiator to the response to that request by a session responder. The session includes a plurality of packets formed by a bidirectional flow. A TCP packet is used as an example. A plurality of packets having the same quintuple after the source and destination in the quintuple are swapped belong to the same session. The quintuple includes a source IP, a destination IP, a source port, a destination port, and a protocol number. A direction of the session may also be referred to as a connection direction of the session, a connection direction of a packet, or the like. This is not limited. The connection direction of the session may be a direction from the session initiator to the session responder.

[00112] The distributed firewall system shown in FIG. 1b differs from the distributed firewall system shown in FIG. 1a in that each engine node in the distributed firewall system may include a message broadcast module, a session processing module, and a stream reassembly module.

[00113] The message broadcast module may be configured to: implement a sending action and/or a receiving action of an engine node, assist the session processing module in implementing access control of a packet by the engine node, and assist the stream reassembly module in implementing stream reassembly processing on a packet segment or a packet fragment by the engine node. For example, the message broadcast module may send a session table to another engine node in the distributed firewall system, which means that the plurality of engine nodes in the distributed firewall system may share the session table. In this way, a session for which the engine node hosting the message broadcast module is responsible may also be served by the

another node. For example, the message broadcast module may send stream reassembly-related information to another engine node in the distributed firewall system, which means that the plurality of engine nodes in the distributed firewall system may share the stream reassembly-related information. In this way, a data stream for which the engine node hosting the message broadcast module is responsible may be collaboratively processed by the plurality of engine nodes.

[00114] The session processing module may also be referred to as a session table module, and may be configured to implement access control of a packet by the engine node. For example, the session processing module may perform access control on the received packet based on a session table (the session table may be maintained by the session processing module, or may be from another engine node via the message broadcast module, which is not limited).

[00115] The stream reassembly module may be configured to enable an engine node to perform stream reassembly processing on packet segments or packet fragments. For example, the stream reassembly module may perform stream reassembly processing on the received packet segments (or packet fragments) based on the stream reassembly-related information (the stream reassembly-related information may be maintained by the stream reassembly module, or may be from another engine node via the message broadcast module, which is not limited), or send the received packet segments (or packet fragments) to the another engine node via the message broadcast module for stream reassembly processing.

[00116] It should be noted that the plurality of functional modules may be implemented by software, hardware, or a combination of software and hardware, and the plurality of functional modules may be randomly combined or divided based on a specific implementation. In addition, the distributed firewall system provided in this embodiment of this application may include another node. An implementation form of the another node is not limited in embodiments of this application.

[00117] The plurality of engine nodes in the distributed firewall system provided in this embodiment of this application may share the session table, the stream reassembly-related information, and/or the like, to reduce impact on service transmission caused by scale-out or scale-in of the distributed firewall system or a fault of an engine node. In the following, with reference to FIG. 2 to FIG. 5B, the data processing method provided in embodiments of this application is described from a perspective of exchange between a first engine node and a second engine node and from a perspective of the first engine node.

[00118] FIG. 2 is a schematic flowchart of a first data processing method according to an embodiment of this application. The method may be applied to the distributed firewall system shown in FIG. 1a or FIG. 1b. As shown in FIG. 2, the method may include the following steps.

[00119] S201: The second engine node stores at least one message to be sent.

[00120] The second engine node may write the at least one to-be-sent message into a file (or a data file). Writing the message into a file may be understood as storing, or writing the message into a hard disk drive (or a disk, or a memory, or the like). For ease of understanding, the following uses an example for description in which the at least one message to be sent by the second engine node is denoted as M messages, where M is a positive integer. FIG. 2 shows an example in which the second engine node stores the to-be-sent M messages. The M messages may be understood as messages shared by the second engine node to another engine node in the distributed firewall system. The messages may be understood as one or more of the following: information, a table, a text, data, or the like.

[00121] In this embodiment of this application, the M messages may include a first session table, or include stream reassembly-related first information, or include the first session table and the stream reassembly-related first information. The first session table may include information about at least one session. The information about the at least one session may include an identifier of the at least one session and indication information indicating whether a connection direction of each session in the at least one session is determinate. Further, if the indication information indicates that the connection direction of the session is determinate, the information about the at least one session may further include the connection direction of the session, as shown in Table 1. Table 1 shows an example in which the first session table includes three sessions. It should be understood that the information about the at least one session may further include other content. This is not limited in embodiments of this application.

Table 1

Identifiers of the sessions	Whether connection directions of the sessions are determinate	Connection directions of the sessions
Session 1	Indeterminate	
Session 2	Determinate	From A to B
Session 3	Indeterminate	

[00122] The stream reassembly-related first information may include an identifier of at least one data stream and indication information indicating whether processing of each data stream in the at least one data stream has been completed. Optionally, the first information may further include an identifier of a main processing node of the data stream, as shown in Table 2. Three data streams are used as an example in Table 2. It should be understood that the first information may further include other content. This is not limited in embodiments of this application.

Table 2

Identifiers of the data streams	Processing completed or not	Main processing node
Data stream 1	Completed	
Data stream 2	Uncompleted	Engine node 1
Data stream 3	Uncompleted	

[00123] It should be noted that the M messages may alternatively include some content of the first session table and/or include some content of the first information. For ease of understanding, an example in which the M messages include the first session table and/or the first information is used for description in this embodiment of this application.

[00124] S202: The second engine node sends the M messages. For example, the second engine node may send the M messages to at least one engine node other than the second engine node in the distributed firewall system. For example, a message broadcast module in the second engine node sends the M messages.

[00125] For example, the second engine node may write the M messages into a file, and send one copy of the M messages to the at least one engine node other than the second engine node in the distributed firewall system. In an implementation, the second engine node may buffer (or buffer) the M messages; or the second engine node may write the M messages into a buffer (or a buffer). That is, the second engine node writes the M messages into the file and the buffer. Further, when a second condition is met, the second engine node may send the M messages to the at least one engine node other than the second engine node in the distributed firewall system. The second condition may include that a data volume of the M messages is greater than or equal to a third threshold, or a buffering duration of the M messages is greater than or equal to a fourth threshold, or the data volume of the M messages is greater than or equal to the third threshold and the buffering duration of the M messages is greater than or equal to the fourth threshold. Both the third threshold and the fourth threshold may be preset. This is not limited. For example, the third threshold may be 1.3 KB. This is not limited. For example, the fourth threshold may be 100 microseconds. This is not limited.

[00126] In an implementation, the second engine node may determine sequence numbers of the M messages. For example, the second engine node may determine the sequence numbers of the M messages based on time (or timestamps) at which the M messages are written into the file and offsets at which the M messages are written into the file. The time at which the M messages are written into the file may be denoted as first time or a first timestamp.

[00127] In an implementation, the distributed firewall system (or the second engine node) may have an expiration mechanism. For example, when storage duration of any one of the M messages

exceeds a specified threshold, the second engine node may delete at least one message whose storage duration exceeds the specified threshold. For example, the first engine node may delete the at least one message from the file and/or the buffer, to reduce waste of storage resources.

[00128] Optionally, the second engine node may send the M messages to the at least one engine node by using the user datagram protocol (user datagram protocol, UDP). This can ensure real-time data transmission. When the distributed firewall system includes three or more engine nodes, the second engine node may send the M messages to a plurality of engine nodes other than the second engine node in the distributed firewall system in a UDP multicast manner, to reduce network transmission resources, or the second engine node may send the M messages to a plurality of engine nodes other than the second engine node in the distributed firewall system one by one in a UDP unicast manner, to adapt to a network environment that does not support UDP multicast.

[00129] Through S201 and S202, data sharing (or data synchronization) can be implemented between the plurality of engine nodes in the distributed firewall system. In an implementation, the plurality of engine nodes in the distributed firewall system may share a same key value (key value, KV) structure and have a local memory-level read/write speed, so that data written by any engine node can be read globally. The KV structure may be, for example, a hash (hash) table, but is not limited thereto. For example, an engine node updates (or modifies) a hash table, and the engine node may send updated content or an updated hash table to another engine node. For example, as shown in FIG. 3, data 1 of an engine node 1 is first written into a file (not shown in FIG. 3) and a local buffer 1, and then sent to an engine node 2 and an engine node 3; data 2 of the engine node 2 is first written into a file (not shown in FIG. 3) and a local buffer 1, and then sent to the engine node 1 and the engine node 3; and data 3 of the engine node 3 is first written into a file (not shown in FIG. 3) and a local buffer 1, and then sent to the engine node 1 and the engine node 2. In this way, the engine node 1, the engine node 2, and the engine node 3 can all use the data 1, the data 2, and the data 3. It may be understood that the local buffer of the engine node may be deployed outside the engine node, or may be deployed inside the engine node. This is not limited. FIG. 3 shows an example in which the engine node 1, the engine node 2, and the engine node 3 communicate with each other through a bus (represented by a black bold line in FIG. 3).

[00130] It should be noted that the second engine node may send the M messages to the at least one engine node other than the second engine node in the distributed firewall system, where the at least one engine node may include the first engine node, or may not include the first engine node. For example, the first engine node may be an engine node added to the distributed firewall system after S202 and before S203.

[00131] S203: The first engine node creates (or constructs, determines, or establishes) a first bitmap file for the second engine node.

[00132] The first engine node may create at least one bitmap file for each engine node in at least one engine node other than the first engine node in the distributed firewall system. The at least one engine node other than the first engine node in the distributed firewall system includes the second engine node. The at least one bitmap file created by the first engine node for the second engine node includes the first bitmap file. The at least one bitmap file created by the first engine node for the second engine node may be used to deduplicate a plurality of messages sent by the second engine node to the first engine node, and quickly determine which messages have not been received. For ease of understanding, the following uses the first bitmap file as an example for description. For other bitmap files, refer to descriptions of the first bitmap file.

[00133] For example, the first engine node may create the first bitmap file in response to receiving a 1st message from the second engine node, where the 1st message belongs to the M messages. Alternatively, the first engine node may create the first bitmap file based on other information. If the first engine node is a new engine node, the first location file may be understood as an empty file, or the first location file may be understood as a full-range file. Alternatively, the first engine node may create the first bitmap file based on configuration information of a control node. Alternatively, after the first engine node is added to the distributed firewall system, the first engine node may send a request for obtaining shared information to the at least one engine node (including the second engine node) other than the first engine node in the distributed firewall system, and create the at least one bitmap file for the at least one engine node after receiving a corresponding message. A trigger condition for creating the first bitmap file by the first engine node is not limited in embodiments of this application.

[00134] The first bitmap file may indicate a reception status of each message in the M messages of the second engine node by the first engine node. Optionally, the first bitmap file may be named based on the first time (or the first timestamp), to indicate a reception status of, by the first engine node in the first time, a message indicated by each sequence number. The first time (or the first timestamp) is time at which the second engine node writes the M messages into the file. For example, the first bitmap file may include sequence numbers of the M messages and indication information indicating whether each message in the M messages is received, as shown in Table 3. Three messages are used as an example in Table 3.

Table 3

Sequence numbers of the messages	Received or not
Sequence number 1	Not received
Sequence number 2	Received

Sequence numbers of the messages	Received or not
Sequence number 3	Not received

[00135] In S202, the second engine node sends the M messages. Each time the first engine node receives a message, the first engine node may determine, based on the first bitmap file, whether the message has already been received. A message 1 in the M messages is used as an example. If the first bitmap file indicates that the message 1 has already been received, the first engine node discards the message 1. If the first bitmap file indicates that the message 1 is not received, the first engine node updates the first bitmap file, where an updated first bitmap file indicates that the message 1 has already been received, and transfers the message 1 to an upper-layer application for processing (for example, the message 1 is written into a session table and/or for stream reassembly). The first engine node processes each received message in a processing manner of the message 1 until the last message received by the first engine node is processed. Further, the first engine node may periodically or aperiodically detect the first bitmap file, to determine whether the M messages are already received. If the first bitmap file indicates that the M messages are already received, the process ends. If the first bitmap file indicates that the N messages in the M messages have not been received, the first engine node performs S204. N is a positive integer less than or equal to M. For example, if the first engine node is a new node, N may be equal to M. In this embodiment of this application, an example in which the N messages in the M messages have not been received is used for description. In S202 in FIG. 2, "X" indicates that the N messages in the M messages have not been received.

[00136] S204: The first engine node sends a request message to the second engine node when the first bitmap file indicates that the N messages in the M messages have not been received. Correspondingly, the second engine node receives the request message from the first engine node.

[00137] The request message may be used to request the N messages, or the request message may be used to request retransmission of the N messages, or the request message may be used to obtain the N messages. Optionally, the request message includes sequence numbers of the N messages. For example, the first engine node may send the request message to the second engine node by using a TCP protocol. This is not limited.

[00138] S205: The second engine node sends the N messages to the first engine node.

[00139] For example, after receiving the request message, the second engine node may send the N messages to the first engine node based on the request message. The first engine node may process each received message in the processing manner of the message 1. If some or all of the N messages have not been received, the first engine node performs S203 until all the N messages have been received. FIG. 2 shows an example in which the first engine node receives the N

messages.

[00140] S206: The first engine node modifies the first bitmap file.

[00141] The first engine node modifies (or updates) the first bitmap file. For example, the first engine node modifies (or updates) the first bitmap file based on receiving statuses of the N messages. Refer to the processing manner of the message 1. A modified first bitmap file indicates that the M messages are already received.

[00142] FIG. 2 is described by using an example in which the first engine node serves as a message receiver. Optionally, the first engine node may alternatively serve as a message sender. In a possible implementation, the first engine node may store the at least one message to be sent (or write the message into the file), and send the at least one message to the at least one engine node other than the first engine node in the distributed firewall system. For brevity, the following uses an example for description in which the at least one message to be sent by the first engine node is denoted as H messages, where H is a positive integer. The H messages may include a second session table, or include stream reassembly-related second information, or include the second session table and the stream reassembly-related second information. For the H messages, refer to descriptions of the M messages. For the second session table, refer to descriptions of the first session table. For the second information, refer to descriptions of the first information. Details are not described again.

[00143] For example, the first engine node sending the H messages to the at least one engine node other than the first engine node in the distributed firewall system may be: The first engine node writes the H messages into a buffer, and sends the H messages to the at least one engine node other than the first engine node in the distributed firewall system when a first condition is met. The first condition may include that a data volume of the H messages is greater than or equal to a first threshold, or a buffering duration of the H messages is greater than or equal to a second threshold, or the data volume of the H messages is greater than or equal to the first threshold and the buffering duration of the H messages is greater than or equal to the second threshold, where H is a positive integer. The first threshold and the second threshold may be predefined. This is not limited.

[00144] Optionally, the first engine node may further determine sequence numbers of the H messages. For an implementation process of determining the sequence numbers of the H messages, refer to descriptions of determining the sequence numbers of the M messages by the second engine node. Details are not described again.

[00145] It should be noted that an execution sequence of the steps in FIG. 2 is an example, and is not limited thereto. For example, the first engine node may synchronously receive the N messages and modify the first bitmap file.

[00146] In the foregoing first data processing method, the plurality of engine nodes in the

distributed firewall system can share the session table (and/or the stream reassembly-related information). In this way, when the distributed firewall system is scaled in (that is, a quantity of engine nodes is reduced) or an engine node is faulty, because another engine node shares a session table (and/or stream reassembly-related information) of the engine node, the another engine node can continue to serve a session (and/or a data stream) for which the engine node is responsible, thereby reducing impact on a service caused by scale-in of the distributed firewall system or a fault of the engine node. When the distributed firewall system is scaled out (that is, a new engine node is added to the distributed firewall system), according to the foregoing first data processing method, the engine node can obtain historical data (that is, a session table and/or stream reassembly-related information) of another engine node in the distributed firewall system. In this way, even if a service task of the another engine node is distributed to the new engine node, the new engine node can continue to serve the service based on the historical data, thereby reducing impact on the service during scale-out of the distributed firewall system.

[00147] A single session is processed by an engine node. When traffic of the session is heavy, the engine node may be overwhelmed by an elephant flow, leading to a risk that a single engine node crashes and becomes unable to function. In the foregoing embodiment of this application, the plurality of engine nodes in the distributed firewall system can share the session table and/or the stream reassembly-related information, which means that one session can be distributed to the plurality of engine nodes for processing. This can reduce a risk that a single node crashes due to a large single session flow, so that the distributed firewall system can support a larger single session flow, thereby improving stability of the distributed firewall system.

[00148] As mentioned above, the plurality of engine nodes in the distributed firewall system can share the session table, and the session table is mainly used to control access to a packet. This is described below with reference to FIG. 4A and FIG. 4B.

[00149] FIG. 4A and FIG. 4B are a schematic flowchart of a second data processing method according to an embodiment of this application. The method may be applied to the distributed firewall system shown in FIG. 1a or FIG. 1b. As shown in FIG. 4A and FIG. 4B, the method may include the following steps.

[00150] S401: The first engine node receives a first packet.

[00151] The first packet belongs to a first session. The first packet may be from an internal network, or may be from an external network. This is not limited. For example, when joining the distributed firewall system, the first engine node may load a configuration, construct an ACL, and the like, and obtain historical data of the distributed firewall system by using the foregoing first data processing method, to obtain a first session table. Optionally, the first engine node may further obtain first information. Then, the first engine node starts a packet receiving and sending process

to receive the first packet.

[00152] S402: The first engine node determines whether the first session is a new session. For example, a session processing module in the first engine node may determine whether the first session is a new session.

5 **[00153]** For example, the first engine node may determine, by parsing a first packet header, whether the first session is a new session. For example, if the first packet is a TCP syn packet, the first engine node determines that the first session is a new session, and determines that a connection direction of the first session is a sending direction of the first packet. For another example, if the first packet is a TCP syn-ack packet, the first engine node determines that the first session is a new session, and determines that a connection direction of the first session is the reverse of a sending direction of the first packet. For another example, if the first packet is an ICMP ping request packet, the first engine node determines that the first session is a new session, and determines that a connection direction of the first session is a sending direction of the first packet. For another example, if the first packet is an ICMP ping response packet, the first engine node determines that
10 the first session is a new session, and determines that a connection direction of the first session is the reverse of a sending direction of the first packet. An implementation process in which the first engine node determines whether the first session is a new session is not limited in embodiments of this application.

[00154] If the first engine node determines that the first session is a new session, S403 and S404 are performed; or if the first engine node determines that the first session is not a new session, S406 is performed.

[00155] S403: The first engine node determines, based on the connection direction of the first session and the ACL, whether to allow access to the first packet. For example, the session processing module in the first engine node may determine, based on the connection direction of the first session and the ACL, whether to allow access to the first packet.
25

[00156] If the first session is a new session, the first engine node may directly obtain the connection direction of the first session, and determine, based on the connection direction of the first session and the ACL, whether to allow access to the first packet. For example, the first engine node queries the ACL based on the connection direction of the first session. Further, if a query result indicates that the first packet is allowed to pass through, the first engine node forwards the first packet (for example, sends the first packet to a destination of the first packet); or if a query result indicates that the first packet is not allowed to pass through, the first engine node discards the first packet.
30

[00157] S404: The first engine node modifies the first session table. For example, the session processing module in the first engine node may modify the first session table.
35

[00158] S404 and S405 are optional steps, and are represented by dashed lines in FIG. 4A. The first engine node may determine, based on the first session table, whether to modify the first session table. For example, if the first session table includes an identifier of the first session, fourth information, and the connection direction of the first session, the first engine node determines not to modify the first session table; or if the first session table does not include an identifier of the first session, or the first session table includes an identifier of the first session and third information, with the third information indicating that the connection direction of the first session is indeterminate, the first engine node may modify the first session table. A modified first session table includes the identifier of the first session, the fourth information, and the connection direction of the first session.

[00159] S405: The first engine node sends the modified first session table to at least one engine node. For example, a message broadcast module in the first engine node may send the modified first session table to the at least one engine node.

[00160] The at least one engine node is an engine node other than the first engine node in the distributed firewall system. For an implementation process of S405, refer to the implementation processes of S201 and S202. Details are not described again. For example, the first engine node writes the modified first session table into a file and a buffer, and sends the modified first session table in the buffer.

[00161] It should be noted that the first engine node may send the modified first session table to the at least one engine node, or may send modified content of the first session table to the at least one engine node. This is not limited in embodiments of this application.

[00162] S406: The first engine node determines whether the first session table includes an identifier of the first session. For example, the session processing module in the first engine node may determine whether the first session table includes the identifier of the first session.

[00163] When the first session is not a new session, the first engine node may determine the connection direction of the first session based on the first session table. For example, the first engine node may determine whether the first session table includes the identifier of the first session; or the first engine node may determine whether the first session hits the first session table; or the first engine node may determine whether the first session matches the first session table; or the first engine node may determine whether the first session table includes information about the first session.

[00164] If the first engine node determines that the first session table does not include the identifier of the first session, S407 to S409 are performed; or if the first engine node determines that the first session table includes the identifier of the first session, S410 is performed.

[00165] S407: The first engine node determines, based on the connection direction of the first

session and the ACL, whether to allow access to the first packet. For example, the session processing module in the first engine node may determine, based on the connection direction of the first session and the ACL, whether to allow access to the first packet.

[00166] The connection direction of the first session is the sending direction of the first packet.

5 To be specific, the first engine node uses the sending direction of the first packet as the connection direction of the first session, and determines, based on the connection direction of the first session and the ACL, whether to allow access to the first packet. For example, the first engine node queries the ACL based on the sending direction of the first packet. Further, if a query result indicates that the first packet is allowed to pass through, the first engine node forwards the first packet (for
10 example, sends the first packet to a destination of the first packet); or if a query result indicates that the first packet is not allowed to pass through, the first engine node discards the first packet.

[00167] S408: The first engine node modifies the first session table. For example, the session processing module in the first engine node modifies the first session table.

[00168] A modified first session table includes the identifier of the first session and third
15 information. The third information indicates that the connection direction of the first session is indeterminate.

[00169] S409: The first engine node sends the modified first session table to at least one engine node. For example, a message broadcast module in the first engine node may send the modified first session table to the at least one engine node.

20 **[00170]** The at least one engine node is an engine node other than the first engine node in the distributed firewall system. For an implementation process of S409, refer to the implementation process of S405. Details are not described again.

[00171] S410: The first engine node determines whether the connection direction of the first session is determinate. For example, the session processing module in the first engine node may
25 determine whether the connection direction of the first session is determinate.

[00172] The first engine node may determine, based on the first session table, whether the connection direction of the first session is determinate. For example, if the first session table includes the identifier of the first session, fourth information, and the connection direction of the first session, the first engine node determines that the connection direction of the first session is
30 determinate, and content of S411 is performed; or if the first session table includes the identifier of the first session and third information, the first engine node determines that the connection direction of the first session is indeterminate, and S412 is performed.

[00173] S411: The first engine node determines, based on the connection direction of the first session and the ACL, whether to allow access to the first packet. For example, the session
35 processing module in the first engine node may determine, based on the connection direction of

the first session and the ACL, whether to allow access to the first packet.

[00174] The first engine node obtains the connection direction of the first session from the first session table, and determines, based on the connection direction of the first session and the ACL, whether to allow access to the first packet. Further, if a query result indicates that the first packet is allowed to pass through in either a forward direction or a reverse direction, the first engine node forwards the first packet (for example, sends the first packet to a destination of the first packet); or if a query result indicates that the first packet is allowed to pass through in neither a forward direction nor a reverse direction, the first engine node discards the first packet.

[00175] S412: The first engine node determines, based on the connection direction of the first session and the ACL, whether to allow access to the first packet. For example, the session processing module in the first engine node may determine, based on the connection direction of the first session and the ACL, whether to allow access to the first packet.

[00176] The connection direction of the first session includes a forward direction and a reverse direction of the first packet. To be specific, the first engine node uses the forward direction and the reverse direction of the first packet as the connection direction of the first session, and determines, based on the connection direction of the first session and the ACL, whether to allow access to the first packet. For example, the first engine node queries the ACL based on the forward direction and the reverse direction of the first packet. Further, if a query result indicates that the first packet is allowed to pass through in either the forward direction or the reverse direction, the first engine node forwards the first packet (for example, sends the first packet to a destination of the first packet); or if a query result indicates that the first packet is allowed to pass through in neither the forward direction nor the reverse direction, the first engine node discards the first packet. The forward direction and the reverse direction of the first packet are the sending direction of the first packet and the reverse of the sending direction of the first packet.

[00177] In the foregoing second data processing method, the first engine node can implement access control of the first packet based on a session table shared by the distributed firewall system. A plurality of engine nodes in the distributed firewall system share the session table to ensure that a session for which any engine node is responsible can also be served by another engine node. This allows the firewall to adapt to elastic scale-out and scale-in application scenarios.

[00178] As mentioned above, the plurality of engine nodes in the distributed firewall system can share the stream reassembly-related information, where the stream reassembly-related information is mainly used to perform stream reassembly processing on packet segments or packet fragments. This is described below with reference to FIG. 5A and FIG. 5B.

[00179] FIG. 5A and FIG. 5B are a schematic flowchart of a third processing method according to an embodiment of this application. The method may be applied to the distributed firewall system

shown in FIG. 1a or FIG. 1b. As shown in FIG. 5A and FIG. 5B, the method may include the following steps.

[00180] S501: The first engine node receives a second packet.

[00181] The second packet belongs to a first data stream. The second packet may be from an internal network, or may be from an external network. This is not limited. For example, when joining the distributed firewall system, the first engine node may load a configuration, construct an ACL, and the like, and obtain historical data of the distributed firewall system by using the foregoing first data processing method, to obtain stream reassembly-related first information. Optionally, the first engine node may further obtain a first session table. Then, the first engine node starts a packet receiving and sending process to receive the second packet.

[00182] S502: The first engine node determines whether the second packet is a 1st packet in the first data stream. For example, a stream reassembly module in the first engine node may determine whether the second packet is the 1st packet in the first data stream.

[00183] For example, the first engine node parses a packet header of the second packet, and may determine whether the second packet is the 1st packet in the first data stream. If the first engine node determines that the second packet is the 1st packet in the first data stream, S503 and S504 are performed; or if the first engine node determines that the second packet is not the 1st packet in the first data stream, S507 is performed.

[00184] S503: The first engine node performs stream reassembly processing on the second packet. For example, the stream reassembly module in the first engine node may perform stream reassembly processing on the second packet.

[00185] Optionally, if a buffer of the first engine node includes a packet (for example, denoted as a fourth packet) of the first data stream, the first engine node may perform stream reassembly processing on the second packet and the fourth packet.

[00186] S504: The first engine node determines whether reassembly of the first data stream has been completed. For example, the stream reassembly module in the first engine node may determine whether reassembly of the first data stream has been completed.

[00187] If the first engine node determines that reassembly of the first data stream has been completed, S505 is performed; or if the first engine node determines that reassembly of the first data stream has not been completed, S506 is performed.

[00188] S505: The first engine node sends fifth information to at least one engine node. For example, a message broadcast module in the first engine node may send the fifth information to the at least one engine node.

[00189] The at least one engine node is an engine node other than the first engine node in the distributed firewall system. The fifth information may indicate that reassembly of the first data

stream has been completed, so that another engine node deletes a packet of the first data stream in a buffer, thereby reducing storage resources. For an implementation process of S505, refer to the implementation processes of S201 and S202. Details are not described again.

[00190] S506: The first engine node sends sixth information to at least one engine node. For example, a message broadcast module in the first engine node may send the sixth information to the at least one engine node.

[00191] The at least one engine node is an engine node other than the first engine node in the distributed firewall system. The sixth information may indicate that reassembly of the first data stream has not been completed and a main processing node of the first data stream is the first engine node, so that another engine node send a packet of the first data stream buffered in a buffer to the first engine node. For an implementation process of S506, refer to the implementation processes of S201 and S202. Details are not described again.

[00192] S507: The first engine node determines, based on the first information, whether reassembly of the first data stream has been completed. For example, the stream reassembly module in the first engine node may determine, based on the first information, whether reassembly of the first data stream has been completed.

[00193] If the first information is fifth information, the first engine node determines that reassembly of the first data stream has been completed, and then discards the second packet. The procedure ends. Alternatively, if the first information does not include information about the first data stream, S508 and S509 are performed. Alternatively, if the first information indicates that reassembly of the first data stream has not been completed and a main processing node is a third engine node, S510 is performed.

[00194] S508: The first engine node buffers the second packet.

[00195] For example, the first engine node writes the second packet into the buffer.

[00196] S509: The first engine node determines whether the main processing node is the first engine node. For example, the stream reassembly module in the first engine node may determine whether the main processing node is the first engine node.

[00197] If the first engine node determines that the second packet is not the 1st packet in the first data stream in S502, and then receives the 1st packet (for example, denoted as a third packet) in the first data stream (for example, receives the third packet after S502 and before S507), the first engine node determines that the first engine node itself is the main processing node, and S503 is performed. For example, the first engine node receives the third packet (not shown in FIG. 5A and FIG. 5B), and performs stream reassembly processing on the second packet and the third packet when the third packet is the 1st packet in the first data stream.

[00198] Alternatively, if the first engine node determines, based on the first information or

seventh information, that the main processing node is not the first engine node, S510 is performed. For example, if the first information indicates that reassembly of the first data stream has not been completed and the main processing node is the third engine node, the first engine node determines, based on the first information, that the main processing node is not the first engine node, and S510 is performed. For another example, the first engine node receives the seventh information (for example, receives the seventh information from the third engine node after S507 and in S509), where the seventh information indicates that reassembly of the first data stream has not been completed and the main processing node is the third engine node, the first engine node determines, based on the seventh information, that the main processing node is not the first engine node, and S510 is performed.

[00199] S510: The first engine node sends the second packet to the third engine node.

[00200] Correspondingly, the third engine node receives the second packet and performs stream reassembly on the second packet. For example, if the first information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is the third engine node, the third engine node sends the second packet to the third engine node based on the first information. For another example, if the first information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream does not exist, the first engine node may receive the seventh information, and send the second packet to the third engine node based on the seventh information.

[00201] In the foregoing third data processing method, the first engine node can implement reassembly processing on the packet segments or the packet fragments based on the stream reassembly-related information shared by the distributed firewall system. The plurality of engine nodes in the distributed firewall system share the stream reassembly-related information, so that packet segments or packet fragments of a same data stream can be distributed to any engine node. In this way, the main processing node can implement reassembly on the packet segments or the packet fragments by using the foregoing third data processing method, to obtain a complete data stream.

[00202] In embodiments provided in this application, the method provided in embodiments of this application is described from a perspective of the first engine node. The steps performed by the first engine node may be implemented by different functional entities that form the first engine node. The first engine node may include a hardware structure and/or a software module, and implement the functions in a form of the hardware structure, the software module, or a combination of the hardware structure and the software module. Whether a function in the foregoing functions is performed by using the hardware structure, the software module, or the combination of the hardware structure and the software module depends on particular applications and design

constraints of the technical solutions.

[00203] With reference to the accompanying drawings, the following describes a data processing apparatus configured to implement the foregoing methods in embodiments of this application. Therefore, all the foregoing content may be used in the following embodiments.

5 Repeated content is not described again.

[00204] FIG. 6 is a diagram of a structure of a data processing apparatus according to an embodiment of this application. The data processing apparatus 600 may implement functions of the first engine node in the foregoing method embodiments. As shown in FIG. 6, the data processing apparatus 600 includes a processing module 601, a sending module 602, and a receiving module 603. The processing module 601 may also be referred to as a processing unit, a logic unit, or the like. The sending module 602 may also be referred to as a sending unit. The receiving module 603 may also be referred to as a receiving unit. The sending module 602 and the receiving module 603 may also be collectively referred to as a transceiver module (or a transceiver unit), a communication module (or a communication unit), an interface circuit, or the like.

15 **[00205]** Optionally, the data processing apparatus 600 may further include a storage module 604, which is represented by a dashed line in FIG. 6. The storage module 604 may also be referred to as a storage unit, a memory, or the like.

[00206] The processing module 601, the sending module 602, the receiving module 603, and the storage module 604 may all be implemented by software, or may be implemented by hardware, or may be implemented by a combination of software and hardware. For example, the following uses the processing module 601 as an example to describe an implementation of the processing module 601. Similarly, for implementations of the sending module 602, the receiving module 603, and the storage module 604, refer to the implementation of the processing module 601.

[00207] The module is used as an example of a software functional unit, and the processing module 601 may include code run on a computing instance. The computing instance may include at least one of a physical host (a compute device), a virtual machine, and a container. Further, there may be one or more computing instances. For example, the processing module 601 may include code run on a plurality of hosts/virtual machines/containers. It should be noted that the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same region (region), or may be distributed in different regions. Further, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same availability zone (availability zone, AZ), or may be distributed in different AZs. Each AZ includes one data center or a plurality of data centers that are geographically close to each other. Usually, one region may include a plurality of AZs.

35 **[00208]** Similarly, the plurality of hosts/virtual machines/containers configured to run the code

may be distributed in a same virtual private cloud (virtual private cloud, VPC), or may be distributed in a plurality of VPCs. Usually, one VPC is arranged in one region. For cross-region communication between two VPCs in a same region and between VPCs in different regions, a communication gateway needs to be arranged in each of the VPCs, and interconnection between the VPCs is implemented through the communication gateway.

[00209] The module is used as an example of a hardware functional unit, and the processing module 601 may include at least one compute device like a server. Alternatively, the processing module 601 may be a device implemented by using an application-specific integrated circuit (application-specific integrated circuit, ASIC) or a programmable logic device (programmable logic device, PLD), or the like. The PLD may be implemented by using a complex programmable logic device (complex programmable logic device, CPLD), a field-programmable gate array (field-programmable gate array, FPGA), generic array logic (generic array logic, GAL), or any combination thereof.

[00210] A plurality of compute devices included in the processing module 601 may be distributed in a same region, or may be distributed in different regions. The plurality of compute devices included in the processing module 601 may be distributed in a same AZ, or may be distributed in different AZs. Similarly, the plurality of compute devices included in the processing module 601 may be distributed in a same VPC, or may be distributed in a plurality of VPCs. The plurality of compute devices may be any combination of compute devices such as a server, an ASIC, a PLD, a CPLD, an FPGA, and GAL.

[00211] For example, the processing module 601 may be configured to perform S203 and S206 in the method embodiment shown in FIG. 2, perform S402 to S404, S406 and S407, and S410 and S411 in the method embodiment shown in FIG. 4A and FIG. 4B, and perform S502 to S504, S507, and S509 in the method embodiment shown in FIG. 5A and FIG. 5B. The sending module 602 may be configured to perform S204 in the method embodiment shown in FIG. 2, perform S405 and S409 in the method embodiment shown in FIG. 4A and FIG. 4B, perform S505, S506, and S510 in the method embodiment shown in FIG. 5A and FIG. 5B, and the like. The receiving module 603 may be configured to perform S201 and S205 in the method embodiment shown in FIG. 2, perform S401 in the method embodiment shown in FIG. 4A, perform S501 in the method embodiment shown in FIG. 5A, and the like. The storage module 604 may be configured to perform S508 in the method embodiment shown in FIG. 5B, and the like. For a specific implementation process, refer to the foregoing method embodiments. Details are not described herein again.

[00212] This application further provides a compute device 700. As shown in FIG. 7, the compute device 700 includes a bus 701, a processor 702, a memory 703, and a communication

interface 704. The processor 702, the memory 703, and the communication interface 704 communicate with each other through the bus 702. The compute device 700 may be a server, and is configured to implement functions of a first engine node. It should be understood that a quantity of processors and a quantity of memories in the compute device 700 are not limited in this application.

[00213] The bus 701 may be a peripheral component interconnect (peripheral component interconnect, PCI) bus, an extended industry standard architecture (extended industry standard architecture, EISA) bus, or the like. Buses may be classified into an address bus, a data bus, a control bus, and the like. For ease of representation, only one line is used to represent the bus in FIG. 7, but this does not mean that there is only one bus or only one type of bus. The bus 701 may include a path for transmitting information between components (for example, the memory 703, the processor 702, and the communication interface 704) of the compute device 700.

[00214] The processor 702 may include any one or more of processors such as a central processing unit (central processing unit, CPU), a graphics processing unit (graphics processing unit, GPU), a micro processor (micro processor, MP), or a digital signal processor (digital signal processor, DSP).

[00215] The memory 703 may include a volatile memory (volatile memory), for example, a random access memory (random access memory, RAM). The processor 702 may further include a non-volatile memory (non-volatile memory), for example, a read-only memory (read-only memory, ROM), a flash memory, a hard disk drive (hard disk drive, HDD), or a solid-state drive (solid-state drive, SSD).

[00216] The memory 703 stores executable program code, and the processor 702 executes the executable program code to separately implement functions of the processing module 601, the sending module 602, the receiving module 603, and the storage module 604, so as to implement the foregoing method embodiments. That is, the memory 703 stores instructions used to perform the foregoing method embodiments.

[00217] Alternatively, the memory 703 stores executable code, and the processor 702 executes the executable code to implement functions of the processing module 601, the sending module 602, the receiving module 603, and the storage module 604, so as to implement the foregoing method embodiments. That is, the memory 703 stores instructions used to perform the foregoing method embodiments.

[00218] The communication interface 704 implements communication between the compute device 700 and another device or a communication network by using a transceiver module, for example, but not limited to a network interface card or a transceiver. For example, the communication interface 704 may implement functions of the sending module 602 and the

receiving module 603.

[00219] An embodiment of this application further provides a compute device cluster. The compute device cluster includes at least one compute device. The compute device may be a server, for example, a central server, an edge server, or a local server in a local data center. In some
5 embodiments, the compute device may alternatively be a terminal device, for example, a desktop computer, a notebook computer, or a smartphone.

[00220] As shown in FIG. 8, the compute device cluster includes at least one compute device 700. A memory 703 in one or more compute devices 700 in the compute device cluster may store a same instruction used to perform the foregoing method embodiments.

10 **[00221]** In some possible implementations, a memory 703 in one or more compute devices 700 in the compute device cluster may alternatively store some instructions used to perform the foregoing method embodiments. In other words, one compute device or a combination of more compute devices 700 may jointly execute instructions used to perform the foregoing method embodiments.

15 **[00222]** It should be noted that memories 703 in different compute devices 700 in the compute device cluster may store different instructions respectively used to perform some functions of a data processing apparatus 600. To be specific, the instructions stored in the memories 703 in the different compute devices 700 may implement functions of one or more of the following modules: a processing module 601, a sending module 602, a receiving module 603, and a storage module
20 604.

[00223] In some possible implementations, the one or more compute devices in the compute device cluster may be connected through a network. The network may be a wide area network, a local area network, or the like. FIG. 9 shows a possible implementation. As shown in FIG. 9, two
25 compute devices 700A and 700B are connected through a network. Specifically, each compute device is connected to the network through a communication interface in the compute device. In this possible implementation, the memory 703 in the compute device 700A stores an instruction for performing functions of the processing module 601 and the storage module 604. In addition, the memory 703 in the compute device 700B stores an instruction for performing functions of the sending module 602 and the receiving module 603.

30 **[00224]** A connection manner for the compute device cluster shown in FIG. 9 may be that a receiving action and a sending action need to be performed in the foregoing method embodiment provided in this application. Therefore, it is considered that functions implemented by the sending module 602 and the receiving module 603 are performed by the compute device 700B.

[00225] It should be understood that functions of the compute device 700A shown in FIG. 9
35 may alternatively be completed by the plurality of compute devices 700. Similarly, functions of

the compute device 700B may alternatively be completed by the plurality of compute devices 700.

[00226] An embodiment of this application further provides another compute device cluster. For a connection relationship between compute devices in the compute device cluster, refer to the connection manners for the compute device cluster in FIG. 8 and FIG. 9 similarly. The compute device cluster in this embodiment differs from the foregoing compute device cluster in that a memory 703 in one or more compute devices 700 in the compute device cluster may store a same instruction used to perform the foregoing method embodiments.

[00227] In some possible implementations, the memory 703 in the one or more compute devices 700 in the compute device cluster may alternatively store some instructions used to perform the foregoing method embodiments. In other words, one compute device or a combination of more compute devices 700 may jointly execute instructions used to perform the foregoing method embodiments.

[00228] It should be noted that memories 703 in different compute devices 700 in the compute device cluster may store different instructions used to perform some functions of a distributed firewall system. To be specific, the instructions stored in the memories 703 in the different compute devices 700 may implement functions of one or more of the following nodes: a first engine node, a second engine node, and a third engine node.

[00229] An embodiment of this application further provides a computer program product including instructions. The computer program product may be software or a program product that includes the instructions and that can run on a compute device or can be stored in any usable medium. When the computer program product runs on at least one compute device, the at least one compute device is enabled to perform the foregoing method embodiments.

[00230] An embodiment of this application further provides a computer-readable storage medium. The computer-readable storage medium may be any usable medium that can be stored by a compute device, or a data storage device, such as a data center, including one or more usable media. The usable medium may be a magnetic medium (for example, a floppy disk, a hard disk drive, or a magnetic tape), an optical medium (for example, a DVD), a semiconductor medium (for example, a solid-state drive), or the like. The computer-readable storage medium includes instructions, and the instructions instruct a compute device to perform the foregoing method embodiments, or instruct a compute device to perform the foregoing method embodiments.

[00231] Finally, it should be noted that the foregoing embodiments are merely intended for describing the technical solutions of the present invention rather than limiting the present invention. Although the present invention is described in detail with reference to the foregoing embodiments, a person of ordinary skill in the art should understand that the technical solutions described in the foregoing embodiments may still be modified or some technical features thereof may be

equivalently replaced, without departing from the protection scope of the technical solutions of embodiments of the prevention.

CLAIMS

1. A data processing method, wherein a distributed firewall system comprises a plurality of engine nodes, and the method comprises:

creating, by a first engine node, a first bitmap file for a second engine node, wherein the first
5 bitmap file indicates a reception status of each message in at least one message of the second engine node by the first engine node, the at least one message comprises one or more of the following: a first session table or stream reassembly-related first information, and the first engine node and the second engine node belong to the plurality of engine nodes;

sending, by the first engine node, a request message to the second engine node when the first
10 bitmap file indicates that N messages in the at least one message have not been received, wherein the request message is used to request the N messages, and N is a positive integer;

receiving, by the first engine node, the N messages sent by the second engine node; and

modifying, by the first engine node, the first bitmap file, wherein a modified first bitmap file indicates that the at least one message has already been received.

15 2. The method according to claim 1, wherein the method further comprises:

storing, by the first engine node, H messages to be sent, wherein the H messages comprise one or more of the following: a second session table or stream reassembly-related second information, and H is a positive integer; and

sending, by the first engine node, the H messages to at least one engine node other than the
20 first engine node in the plurality of engine nodes.

3. The method according to claim 2, wherein sending the H messages to the at least one engine node other than the first engine node in the plurality of engine nodes comprises:

writing, by the first engine node, the H messages into a buffer; and

sending, by the first engine node, the H messages to the at least one engine node other than
25 the first engine node in the plurality of engine nodes when a first condition is met, wherein the first condition comprises one or more of the following: a data volume of the H messages is greater than or equal to a first threshold, or a buffering duration of the H messages is greater than or equal to a second threshold.

4. The method according to any one of claims 1 to 3, wherein the method further comprises:

30 receiving, by the first engine node, a first packet, wherein the first packet belongs to a first session; and

when the first session is a new session, determining, by the first engine node based on a connection direction of the first session and an access control list, whether to allow access to the first packet.

5. The method according to claim 4, wherein the at least one message comprises the first session table, and the method further comprises:

modifying, by the first engine node, the first session table when the first session table does not comprise an identifier of the first session, or the first session table comprises an identifier of the first session and third information, with the third information indicating that the connection direction of the first session is indeterminate, wherein a modified first session table comprises the identifier of the first session, fourth information, and the connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; and

sending, by the first engine node, the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes.

6. The method according to any one of claims 1 to 3, wherein the at least one message comprises the first session table, and the method further comprises:

receiving, by the first engine node, a first packet, wherein the first packet belongs to a first session; and

when the first session is not a new session, determining, by the first engine node, a connection direction of the first session based on the first session table, wherein the first session table does not comprise an identifier of the first session, and the connection direction of the first session is a sending direction of the first packet;

determining, by the first engine node based on the connection direction of the first session and an access control list, whether to allow access to the first packet;

modifying, by the first engine node, the first session table, wherein a modified first session table comprises the identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate; and

sending, by the first engine node, the modified first session table to the at least one engine node other than the first engine node in the plurality of engine nodes.

7. The method according to any one of claims 1 to 3, wherein the at least one message comprises the first session table, and the method further comprises:

receiving, by the first engine node, a first packet, wherein the first packet belongs to a first session; and

when the first session is not a new session, determining, by the first engine node, a connection direction of the first session based on the first session table; and

determining, by the first engine node based on the connection direction of the first session and an access control list, whether to allow access to the first packet, wherein

the first session table comprises an identifier of the first session, fourth information, and the

connection direction of the first session, and the fourth information indicates that the connection direction of the first session is determinate; or the first session table comprises an identifier of the first session and third information, and the third information indicates that the connection direction of the first session is indeterminate, wherein the connection direction of the first session comprises a forward direction and a reverse direction of the first packet.

8. The method according to any one of claims 1 to 7, wherein the method further comprises: receiving, by the first engine node, a second packet, wherein the second packet belongs to a first data stream;

performing, by the first engine node, stream reassembly processing on the second packet when the second packet is a 1st packet in the first data stream; and

sending, by the first engine node, fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, wherein the fifth information indicates that reassembly of the first data stream has been completed; or sending, by the first engine node, sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, wherein the sixth information indicates that reassembly of the first data stream has not been completed and a main processing node of the first data stream is the first engine node.

9. The method according to any one of claims 1 to 7, wherein the at least one message comprises the first information, the first information does not comprise information about a first data stream, and the method further comprises:

receiving, by the first engine node, a second packet, wherein the second packet belongs to the first data stream;

when the second packet is not a 1st packet in the first data stream, determining, by the first engine node based on the first information, that reassembly of the first data stream has not been completed and a main processing node of the first data stream does not exist; and

buffering, by the first engine node, the second packet.

10. The method according to claim 9, wherein the method further comprises:

receiving, by the first engine node, seventh information, wherein the seventh information indicates that reassembly of the first data stream has not been completed and the main processing node of the first data stream is a third engine node, and the third engine node belongs to the plurality of engine nodes; and

sending, by the first engine node, the second packet to the third engine node.

11. The method according to claim 9, wherein the method further comprises:

receiving, by the first engine node, a third packet, wherein the third packet belongs to the first

data stream;

performing, by the first engine node, stream reassembly processing on the second packet and the third packet when the third packet is the 1st packet in the first data stream; and

5 sending, by the first engine node, fifth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has been completed, wherein the fifth information indicates that reassembly of the first data stream has been completed; or sending, by the first engine node, sixth information to the at least one engine node other than the first engine node in the plurality of engine nodes when reassembly of the first data stream has not been completed, wherein the sixth information indicates that
10 reassembly of the first data stream has not been completed and the main processing node of the first data stream is the first engine node.

12. The method according to any one of claims 1 to 7, wherein the at least one message comprises the first information, the first information indicates that reassembly of the first data stream has not been completed and a main processing node of the first data stream is a third engine
15 node, and the method further comprises:

receiving, by the first engine node, a second packet, wherein the second packet belongs to the first data stream; and

sending, by the first engine node, the second packet to the third engine node based on the first information.

20 13. A data processing method, applied to a distributed firewall system, wherein the distributed firewall system comprises a plurality of engine nodes, the plurality of engine nodes comprise a first engine node and a second engine node, and the method comprises:

storing, by the second engine node, at least one message to be sent, and sending the at least one message to at least one engine node other than the second engine node in the plurality of
25 engine nodes, wherein the at least one message comprises one or more of the following: a first session table or stream reassembly-related first information;

creating, by the first engine node, a first bitmap file for the second engine node, and sending a request message to the second engine node when the first bitmap file indicates that N messages in the at least one message have not been received, wherein the first bitmap file indicates a
30 reception status of each message in the at least one message by the first engine node, the request message is used to request the N messages, and N is a positive integer;

receiving, by the second engine node, the request message from the first engine node, and sending the N messages to the first engine node; and

35 receiving, by the first engine node, the N messages sent by the second engine node, and modifying the first bitmap file, wherein a modified first bitmap file indicates that the at least one

message has already been received.

14. A data processing apparatus, wherein the apparatus is used in a first engine node in a distributed firewall system, the distributed firewall system comprises a plurality of engine nodes, the first engine node belongs to the plurality of engine nodes, and the apparatus comprises a processing module, a sending module, and a receiving module;

the processing module is configured to create a first bitmap file for a second engine node, wherein the first bitmap file indicates a reception status of each message in at least one message of the second engine node by the first engine node, the at least one message comprises one or more of the following: a first session table or stream reassembly-related first information, and the second engine node belongs to the plurality of engine nodes;

the sending module is configured to send a request message to the second engine node when the first bitmap file indicates that N messages in the at least one message have not been received, wherein the request message is used to request the N messages, and N is a positive integer;

the receiving module is configured to receive the N messages sent by the second engine node; and

the processing module is further configured to modify the first bitmap file, wherein a modified first bitmap file indicates that the at least one message has already been received.

15. A distributed firewall system, wherein the distributed firewall system comprises a plurality of engine nodes, and the plurality of engine nodes comprise a first engine node and a second engine node;

the second engine node is configured to: store at least one message to be sent, and send the at least one message to at least one engine node other than the second engine node in the plurality of engine nodes, wherein the at least one message comprises one or more of the following: a first session table or stream reassembly-related first information;

the first engine node is configured to: create a first bitmap file for the second engine node, and send a request message to the second engine node when the first bitmap file indicates that N messages in the at least one message have not been received, wherein the first bitmap file indicates a reception status of each message in the at least one message by the first engine node, the request message is used to request the N messages, and N is a positive integer;

the second engine node is further configured to: receive the request message from the first engine node, and send the N messages to the first engine node; and

the first engine node is further configured to: receive the N messages sent by the second engine node, and modify the first bitmap file, wherein a modified first bitmap file indicates that the at least one message has already been received.

16. A compute device cluster, comprising at least one compute device, wherein each compute

device comprises a processor and a memory; and

a processor of the at least one compute device is configured to execute instructions stored in a memory of the at least one compute device, to enable the compute device cluster to perform the method according to any one of claims 1 to 12 or claim 13.

5 17. A computer-readable storage medium, comprising computer program instructions, wherein when the computer program instructions are executed by a compute device cluster, the compute device cluster performs the method according to any one of claims 1 to 12 or claim 13.

10 18. A computer program product comprising instructions, wherein when the instructions are executed by a compute device cluster, the compute device cluster is enabled to perform the method according to any one of claims 1 to 12 or claim 13.