

PACKET PROCESSING METHOD IN CLOUD COMPUTING SYSTEM, HOST, AND SYSTEM

TECHNICAL FIELD

[0001] The present invention relates to the field of IT technologies, and in particular, to a
5 packet processing method in a cloud computing system, a host, and a system.

BACKGROUND

[0002] A virtualization technology is one of key technologies in the cloud computing field. According to the virtualization technology, physical resources of a host can be abstracted into a shared resource pool to be used by virtual machines (Virtual Machine, VM)
10 running on the host. The virtual machines running on the host may share a network interface card of the host to communicate with an external network of the host. In the prior art, the network interface card may be allocated, by means of network interface card virtualization, to the virtual machines for use. Network interface card virtualization may be implemented by means of single-root input/output virtualization (Single-Root I/O Virtualization, SR-IOV) or
15 multi-root input/output virtualization (Multi Root Input/Output Virtualization, MR-IOV). The network interface card virtualization is also referred to as network interface card passthrough. SR-IOV passthrough is used as an example. When the network interface card supports SR-IOV, the virtual machines running on the host may share the network interface card on the host by using an SR-IOV technology.

[0003] When a network interface card supporting an SR-IOV capability is used on the
20 host, a network port of the network interface card is virtualized into at least one physical function (Physical Function, PF) and multiple virtual functions (Virtual Function, VF). A virtual machine on the host is connected to at least one VF. The network interface card is internally provided with switching equipment that functions as a switch. The switching

equipment forwards a data packet according to a media access control (MediaAccessControl, MAC) table, and forwards a data packet between a physical network port, the PF, and the VF. Because of limited processing and storage capabilities, the switching equipment cannot provide a user with abundant network functions such as a security group, quality of service (Quality of Service, QoS), layer 2 tunnel encapsulation, and distributed routing.

SUMMARY

[0004] This specification describes a packet processing method in a cloud computing system, a host, and a system, to resolve a problem that abundant network functions cannot be provided in a network interface card passthrough scenario.

[0005] According to a first aspect, an embodiment of the present invention provides a host. At least one network interface card is connected to the host. A virtual machine monitor VMM runs on the host, and a first virtual machine runs on the VMM. The VMM includes a virtual bridge and a virtual network function module, and the virtual network function module can provide a network function processing capability. The network interface card includes switching equipment and at least two network ports. A first network port supports a network interface card virtualization capability, the first network port is corresponding to at least one physical function PF and multiple virtual functions VFs, and at least one VF of the first network port is provided for the first virtual machine in a passthrough manner.

[0006] The first virtual machine sends a data packet from the VF that is connected to the first virtual machine. A destination MAC address of the data packet is a MAC address of an uplink port of the virtual bridge. The switching equipment of the first network port forwards the data packet according to the destination MAC address of the data packet, and sends the data packet to the virtual bridge on the VMM. The virtual bridge transfers the data packet to the virtual network function module. The virtual network function module provides abundant network functions for the data packet. In a network interface card passthrough scenario, after sending a data packet from a passthrough VF, a virtual machine may direct the data packet from a network interface card to a virtual network function module on a VMM in the foregoing manner, and provide abundant network functions for a user by using the software

module.

[0007] In a possible design, multiple virtual machines may run on the host, each virtual machine is connected to at least one VF of the first network port, and the multiple VFs of the first network port have different virtual local area network VLAN identifiers. A VLAN identifier different from a VLAN identifier of another VF is set for each VF. This ensures that a broadcast packet carrying a VLAN identifier is propagated only on a virtual local area network corresponding to the VLAN identifier and is not sent to another VF with a different VLAN identifier.

[0008] In a possible design, the first virtual machine obtains the destination MAC address of the to-be-sent data packet by using an APR process. Specifically:

[0009] The first virtual machine sends an ARP request packet by using the VF that is connected to the first virtual machine, where the ARP request packet carries a VLAN identifier of the VF that sends the ARP request packet;

after receiving the ARP request packet, the switching equipment of the first network port broadcasts the ARP request packet; after receiving the ARP request packet broadcast by the switching equipment, the virtual bridge on the VMM forwards the ARP request packet to an SDN controller, where the SDN controller records network topology information of a cloud computing system; and when receiving the ARP request packet, the SDN controller constructs a flow entry, and returns the flow entry to the virtual bridge, where the flow entry includes an ARP pickup setting, the ARP pickup setting is used to instruct the virtual bridge to construct an ARP response packet, and the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet; and

the virtual bridge returns the ARP response packet to the first virtual machine according to the flow entry, where the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet.

[0010] In this embodiment of the present invention, a new ARP process is defined. The destination MAC address of the to-be-sent data packet is set to the MAC address of the uplink port of the virtual bridge, so that the data packet sent by the first virtual machine is forwarded

to the virtual bridge by means of layer 2 forwarding, and the data packet is redirected back to the VMM from the network interface card.

[0011] In a possible design, the flow entry further includes a MAC address setting, and the MAC address setting is used to instruct the virtual bridge to change the destination MAC address of the data packet to a MAC address of a second virtual machine when the virtual bridge receives the data packet that is sent by the first virtual machine to the second virtual machine. The destination MAC address of the data packet sent by the first virtual machine is the MAC address of the uplink port of the virtual bridge. Therefore, when receiving the data packet, the virtual bridge needs to replace the destination MAC address of the data packet with the MAC address of the second virtual machine, so that the data packet sent by the virtual bridge can be routed to the second virtual machine according to a real address of the second virtual machine.

[0012] In a possible design, the uplink port of the virtual bridge is a PF of the first network port, and the virtual bridge is connected to the network interface card by using the PF; or the uplink port of the virtual bridge is a selected VF of the first network port, and the virtual bridge is connected to the network interface card by using the selected VF, where the selected VF that is connected to the virtual bridge is set to a promiscuous mode. It should be noted that, that the virtual bridge is connected to the network interface card by using the PF or the VF means that the virtual bridge is logically connected to the network interface card, that is, the data packet is transmitted between the virtual bridge and the network interface card by using the PF or the VF.

[0013] In a possible design, a source virtual machine (the first virtual machine) and a destination virtual machine (the second virtual machine) of the data packet are on a same host. In this case, after processing the data packet, the virtual network function module returns the processed data packet to the virtual bridge. The virtual bridge receives the processed data packet, and sends the processed data packet to the second virtual machine according to the destination MAC address of the data packet by using the switching equipment.

[0014] Alternatively, the second virtual machine and the first virtual machine may run on different hosts. In this case, the virtual network function module establishes a tunnel between the virtual network function module and another virtual network function module on a host on

which the second virtual machine is located, sends the processed data packet to a second network port of the network interface card, and then sends, by using the tunnel, the processed data packet to the host on which the second virtual machine is located, so that the another virtual network function module sends the processed data packet to the second virtual machine.

[0015] According to a second aspect, an embodiment of the present invention further provides a packet processing method in a cloud computing system. At least one host in the cloud computing system includes a virtual machine monitor VMM and at least one network interface card. A first virtual machine runs on the host. The VMM includes a virtual bridge and a virtual network function module. The network interface card includes switching equipment and at least two network ports. A first network port supports a network interface card virtualization capability. The first network port is corresponding to at least one PF and multiple VFs. The first virtual machine is connected to at least one VF of the first network port. The method includes:

determining, by the first virtual machine, a destination MAC address of a to-be-sent data packet, where the MAC address of the to-be-sent data packet is a MAC address of an uplink port of the virtual bridge;

sending, by the first virtual machine, the data packet to a second virtual machine by using the VF that is connected to the first virtual machine, where a destination IP address carried in the data packet is an IP address of the second virtual machine, and the destination MAC address carried in the data packet is the MAC address of the uplink port of the virtual bridge;

receiving, by the switching equipment, the data packet, and sending the data packet to the virtual bridge according to the destination MAC address of the data packet by using the uplink port of the virtual bridge;

receiving, by the virtual bridge, the data packet, changing the destination MAC address of the data packet to a MAC address of the second virtual machine, and sending a changed data packet to the virtual network function module; and

performing, by the virtual network function module, network function processing on the changed data packet, and sending the processed data packet, where a destination MAC

address of the processed data packet is the MAC address of the second virtual machine.

[0016] In a possible design, the multiple VFs of the first network port have different virtual local area network VLAN identifiers.

[0017] In another possible design, the first virtual machine obtains the destination MAC address of the to-be-sent data packet from an SDN controller by using an ARP process. The destination MAC address is the MAC address of the uplink port of the virtual bridge. A specific obtaining manner is the same as that in the description in the first aspect.

[0018] According to a third aspect, an embodiment of the present invention provides a cloud computing system. The cloud computing system includes the host described in the first aspect and an SDN controller. The SDN controller is configured to receive an ARP request packet from a first virtual machine that is forwarded by a virtual bridge, and return a flow entry to the virtual bridge. The flow entry includes an ARP pickup setting and a MAC address setting, the ARP pickup setting is used to instruct the virtual bridge to construct an ARP response packet based on the ARP request. The ARP response packet carries a MAC address that is of an uplink port of the virtual bridge and that is used as a destination MAC address of a to-be-sent data packet.

[0019] According to a fourth aspect, corresponding to the foregoing apparatus, method, and system, an embodiment of the present invention provides a host. The host has a function of implementing the host defined in the first aspect. The function may be implemented by hardware, or may be implemented by executing corresponding software by hardware. The hardware or software includes one or more modules corresponding to the foregoing function. Specifically, the host includes a first processor, a first memory, and at least one network interface card. The network interface card includes a second processor, a second memory, and at least two network ports. A first network port supports a network interface card virtualization capability. The first network port is corresponding to at least one PF and multiple VFs. The first virtual machine is connected to at least one VF of the first network port. The first memory and the second memory store instructions.

[0020] The first processor is configured to execute an instruction in the first memory to implement functions of the first virtual machine, the virtual bridge, and the virtual network function module in the first aspect.

[0021] The second processor is configured to execute an instruction in the second memory to implement a function of the switching equipment in the first aspect.

[0022] In a possible design, the host and an SDN controller are implemented by a general-purpose or dedicated server. A structure of the server includes a processor, a memory, a system bus, and an input/output interface. The processor is configured to support a corresponding function of the host in the system. The input/output interface is configured to communicate with another component in the cloud computing system. The processor executes the instructions stored in the memories.

[0023] According to a fifth aspect, an embodiment of the present invention provides a computer storage medium that is configured to store a computer software instruction used by the foregoing host. The computer software instruction includes a program designed for executing the foregoing aspects.

[0024] According to a sixth aspect, an embodiment of the present invention provides another computer storage medium that is configured to store a computer software instruction used by the switching equipment in the foregoing network interface card. The computer software instruction includes a program designed for executing the foregoing aspects.

[0025] According to a seventh aspect, an embodiment of the present invention provides a computer program. When a physical server runs the computer program, the physical server performs functions of a virtual machine, a virtual bridge, and a virtual network function module in a host.

[0026] According to an eighth aspect, an embodiment of the present invention provides a computer program. When a processor or a programmable logic circuit in a network interface card runs the computer program, the network interface card performs a function of the switching equipment in the foregoing aspects.

BRIEF DESCRIPTION OF DRAWINGS

[0027] To describe the technical solutions in the embodiments of the present invention or in the prior art more clearly, the following briefly describes the accompanying drawings required for describing the embodiments or the prior art. Apparently, the accompanying

drawings in the following description merely show some embodiments of the present invention, and a person of ordinary skill in the art can derive other implementations from these accompanying drawings without creative efforts. All these embodiments or implementations fall within the protection scope of the present invention.

5 **[0028]** FIG. 1A is a schematic diagram of a virtualized structure on a host according to an embodiment of the present invention;

[0029] FIG. 1B is a schematic diagram of a virtualized structure on a host in a network interface card passthrough scenario according to an embodiment of the present invention;

10 **[0030]** FIG. 2 is a schematic diagram of another virtualized architecture on a host according to an embodiment of the present invention;

[0031] FIG. 3 is a schematic diagram of an ARP process according to an embodiment of the present invention;

[0032] FIG. 4 is a schematic flowchart of sending a data packet to a VM2 by a VM1 according to an embodiment of the present invention;

15 **[0033]** FIG. 5 is a schematic structural diagram of hardware of a computer device according to an embodiment of the present invention; and

[0034] FIG. 6 is a schematic structural diagram of a cloud computing system according to an embodiment of the present invention.

DESCRIPTION OF EMBODIMENTS

20 **[0035]** A network architecture and a service scenario that are described in the embodiments of the present invention are used to describe the technical solutions in the embodiments of the present invention more clearly, and do not constitute any limitation on the technical solutions provided in the embodiments of the present invention. A person of ordinary skill in the art may learn that, with the evolution of network architectures and the
25 emergence of a new service scenario, the technical solutions provided in the embodiments of the present invention are also applicable for similar technical issues.

[0036] As shown in FIG. 1A, FIG. 1A is a schematic diagram of a virtualized structure on a host according to an embodiment of the present invention. The host is a physical server. A

bottom layer of the physical server is a hardware layer. The hardware layer mainly includes hardware resources such as a central processing unit (CPU, Central Processing Unit), a memory, a hard disk, and a network interface card. Server virtualization implements a virtualized running environment of multiple virtual machines (Virtual Machine, VM) on the physical server by using virtualization software (for example, VMWare ESX or Citrix XEN). A software layer that is installed on the server to implement a virtualized environment is referred to as a virtual machine monitor (Virtual Machine Monitor, VMM). The VMM running on the hardware layer is responsible for scheduling, allocating, and managing the hardware resources at the hardware layer. Multiple virtual machines VMs run on the VMM. The VMM provides each virtual machine with a hardware environment such as a virtualized CPU, memory, storage, IO device (such as a network interface card), and Ethernet switch, to ensure that the multiple virtual machines run in a manner of isolating from each other.

[0037] In a virtualized running environment, the VMM creates a virtual network interface card (Virtual Network Interface Card, vNIC) for each virtual machine, and a virtual switch vSwitch provides a capability of communication between the virtual machines and between the virtual machines and an external network. The virtual network interface card of each virtual machine is corresponding to a logical port of the vSwitch. A physical network interface card of the host is corresponding to a port connecting the vSwitch to an external physical switch. When a data packet sent or received by a virtual machine passes through the VMM, a virtual network function module on the VMM performs network function processing on the data that passes through the VMM. The virtual network function module is a software module and can be updated as required. Therefore, the virtual network function module on the VMM can provide abundant network functions for a user.

[0038] A network port of the physical network interface card supports a virtualization capability. This can be implemented by means of single-root input/output virtualization (Single Root Input/Output Virtualization, SR-IOV) or multi-root input/output virtualization (Multi Root Input/Output Virtualization, MR-IOV). In this embodiment of the present invention, an SR-IOV technology is used as an example for description. The SR-IOV technology is a hardware-based virtualization solution. A PCIe (Peripheral Component Interconnect Express, peripheral component interconnect express) device can be efficiently

shared between virtual machines. In addition, the SR-IOV technology is implemented in hardware, so that efficient I/O performance can be obtained.

[0039] The standardization organization PCI-SIG (Peripheral Component Interconnect Special Interest Group, Peripheral Component Interconnect Special Interest Group) defines an SR-IOV specification. For the SR-IOV specification, refer to <http://www.pcisig.com>.

[0040] By using the SR-IOV technology, a single I/O resource may be shared by the multiple virtual machines on the host. In this way, each virtual machine can access a same hardware resource. Therefore, a PCIe device with SR-IOV enabled (for example, the network port of the physical network interface card) may be displayed as multiple separate devices.

Each separate device that is displayed has independent PCIe configuration space. A physical network interface card supporting SR-IOV is used as an example. The physical network interface card includes network ports, and can enable/disable an SR-IOV capability for each network port. A network port for which the SR-IOV capability is enabled is corresponding to at least one physical function (Physical Function, PF) and multiple virtual functions (Virtual Function, VF). According to an existing SR-IOV specification, each PF may have a maximum of 64,000 VFs that are associated with the PF. After a VF is created, the VF may be directly assigned to a virtual machine on the host for use, so that the multiple virtual machines share the PCIe device by using at least one VF that is connected to the multiple virtual machines.

[0041] As shown in FIG. 1B, FIG. 1B is a schematic diagram of a virtualized structure on a host in a network interface card passthrough scenario according to an embodiment of the present invention. A difference between FIG. 1A and FIG. 1B lies in that: A physical network interface card supports single-root IO virtualization, a network port of the physical network interface card is corresponding to at least one PF and multiple VFs, and the VFs can share a physical resource of the physical network interface card (for example, a port of the network interface card).

[0042] When a virtual machine on the host sends a data packet by using a passthrough VF, the data packet sent by the virtual machine is directly transferred to the VF of the network interface card by using a VF driver installed on the virtual machine. As a result, the data packet cannot pass through a virtual network function module on a VMM. Switching equipment of the network interface card has a limited MAC table capacity and a limited

processing capability, and cannot perform unconstrained function extension. Therefore, the network interface card cannot provide abundant network functions. In a possible scenario, the switching equipment is a virtual Ethernet bridge & classifier.

[0043] It should be noted that this embodiment of the present invention is based on the network interface card passthrough scenario. The foregoing network interface card passthrough may be implemented by means of SR-IOV or MR-IOV, and this embodiment of the present invention imposes no limitation thereon.

[0044] As shown in FIG. 2, FIG. 2 is a schematic diagram of another virtualized architecture on a host according to an embodiment of the present invention. The host includes a virtual machine monitor VMM and at least one physical network interface card. At least one virtual machine runs on the host. The at least one physical network interface card includes at least two network ports: a first network port and a second network port. The first network port supports a network interface card virtualization capability, and the second network port is connected to an external physical switch of the host. The first network port is virtualized into at least one PF and at least two VFs (in FIG. 2, two VFs are used as an example for description). A VF driver is installed on a first virtual machine and a second virtual machine. The first virtual machine and the second virtual machine are separately connected to at least one VF. The PF is connected to a virtual bridge on the VMM. The virtual bridge is connected to a virtual network function module on the VMM, and the virtual network function module is connected to the external physical switch of the host by using the second network port.

[0045] In a specific scenario, the virtual bridge on the VMM may be an Open vSwitch bridge, and the PF of the first network port serves as an uplink port of the virtual bridge. It should be noted that a VF of the first network port may be alternatively selected as the uplink port of the virtual bridge, that is, the first network port is connected to the virtual bridge on the VMM by using the selected VF. In this case, a VF driver needs to be loaded on the VMM, and the VF that is connected to the virtual bridge needs to be set to a promiscuous mode. Similar to the PF, the VF that is set to the promiscuous mode can receive a data packet whose destination address is not a MAC address of the VF. For specific description about the promiscuous mode, refer to https://en.wikipedia.org/wiki/Promiscuous_mode. In the embodiments of the present invention, an example in which the PF of the first network port is

connected to the virtual bridge is used for description. A person skilled in the art may understand that, in the foregoing setting manner, the VF in the promiscuous mode (promiscuous mode) may also be used to connect to the virtual bridge.

[0046] In a possible implementation, the first network port and the second network port may be located on a same network interface card, or may be independently located on different network interface cards. This embodiment of the present invention imposes no limitation thereon.

[0047] In the embodiments of the present invention, network interface card virtualization may be specifically hardware virtualization of a network interface card.

[0048] The first virtual machine and the second virtual machine are separately connected to at least one VF of the first network port. Different virtual local area network (Virtual Local Area Network, VLAN) identifiers are set for VFs of the first network port. A VLAN is a broadcast domain. Therefore, a broadcast packet can be sent only to a virtual machine on a VLAN. Therefore, different VLAN identifiers are set for VFs, so that virtual machines are located in different broadcast domains and a broadcast packet sent by the virtual machine can be received only by the virtual machine itself and a PF. During setting of a VLAN identifier for a VF, it needs to be ensured that the VLAN identifier is unique within a host range. VLAN identifiers on different hosts may be the same.

[0049] It should be noted that, in the embodiment shown in FIG. 2, sending a data packet to the second virtual machine by the first virtual machine is communication between two virtual machines within a same host. A person skilled in the art may understand that the second virtual machine and the first virtual machine may be alternatively located on different hosts. In this case, the first virtual machine communicates with the second virtual machine in a cross-host manner. In FIG. 2, only two VFs and two VLAN subinterfaces are used as an example for description. In an actual product implementation, a network interface card supporting a network interface card virtualization function may be virtualized into at least one PF and multiple VFs. A quantity of VFs is not limited to two, and correspondingly, a quantity of VLAN subinterfaces is not limited to two, either.

[0050] In this embodiment of the present invention, an example in which a VM1 (the first virtual machine) sends a data packet to a VM2 (the second virtual machine) is used as an

example to describe a data packet processing process. Switching equipment in the network interface card needs to send, to the virtual network function module on the VMM, the data packet from the VM1 for processing. Before sending the data packet, the first virtual machine determines that a destination media access control MAC address of the to-be-sent data packet is a MAC address of the uplink port of the virtual bridge on the VMM. The first virtual machine sends the data packet to the second virtual machine by using the VF that is connected to the first virtual machine. A destination IP address carried in the data packet is an IP address of the second virtual machine, and the destination MAC address carried in the data packet is the MAC address of the uplink port of the virtual bridge. The switching equipment of the first network port receives the data packet that is sent by the first virtual machine by using the VF, and performs layer 2 forwarding on the data packet according to the destination MAC address of the data packet to send the data packet to the virtual bridge by using the uplink port of the virtual bridge. The virtual bridge receives the data packet, changes the destination MAC address of the data packet to a MAC address of the second virtual machine, and sends a changed data packet to the virtual network function module. The destination MAC address of the data packet is changed, so that the changed MAC address may be used in a subsequent transmission process of data. This ensures that the data packet can be sent to the second virtual machine. After receiving the changed data packet, the virtual network function module performs network function processing on the changed data packet to provide abundant network function services for a user, and then sends the processed data packet to the second virtual machine according to the MAC address of the second virtual machine.

[0051] In this embodiment of the present invention, a virtual machine sends a data packet from a VF that is connected to the virtual machine. Switching equipment of a first network port forwards the data packet, and sends the data packet to a virtual bridge on a VMM. The virtual bridge transfers the data packet to a virtual network function module. The virtual network function module provides abundant network functions for the data packet. In a network interface card passthrough scenario, after sending a data packet from a passthrough VF, a virtual machine may send the data packet to a virtual network function module on a VMM in the foregoing manner. The software module is used to provide abundant network functions for a user, so as to perform virtual network function processing on the data packet.

[0052] Further, an embodiment of the present invention provides an Address Resolution Protocol (Address Resolution Protocol, ARP) process, so that, before sending the data packet, the first virtual machine may obtain the MAC address of the uplink port of the virtual bridge, and set the destination MAC address of the data packet to the MAC address of the uplink port of the virtual bridge. In the ARP process provided in this embodiment of the present invention, a software-defined networking (Software Defined Network, SDN) controller performs pickup for an ARP request initiated by the VM1, adds the MAC address of the uplink port of the virtual bridge on the VMM to an ARP response, and directs, to the PF or VF that is connected to the virtual bridge and by allowing the MAC address to be corresponding to a destination IP address of the data packet, the data packet sent by the VM1. It should be noted that the SDN controller may obtain a network topology of a cloud computing network, that is, may implement the foregoing ARP process.

[0053] Specifically, as shown in FIG. 3, FIG. 3 is a schematic diagram of an ARP process according to an embodiment of the present invention.

[0054] Step 301: A VM1 sends an ARP request packet that carries an IP address of a VM2 and that is used to obtain a MAC address of the VM2. The ARP request packet is sent from a VF that is connected to the VM1. Switching equipment on a network interface card receives the ARP request packet. The ARP request packet carries a VLAN identifier of the VF that sends the ARP request packet.

[0055] Step 302: The switching equipment broadcasts the ARP request packet. Because VFs connected to each virtual machine on a host have different VLAN identifiers, the ARP request packet is sent to a virtual bridge by using a PF.

[0056] In a possible design, when the virtual bridge is connected to the network interface card by using a VF, the ARP request packet is sent to the virtual bridge by using the AF.

[0057] Step 303: The virtual bridge forwards the ARP request packet to an SDN controller.

[0058] Specifically, the virtual bridge may send the ARP request packet to the SDN controller by using a packet-in message.

[0059] Step 304: The SDN controller sets a flow entry for the virtual bridge, and delivers the flow entry to the virtual bridge.

[0060] The flow entry includes:

(1) an ARP pickup setting: the ARP pickup setting indicates that when receiving the ARP packet request that is sent by the VM1 and that is used to obtain the MAC address of the VM2, the virtual bridge constructs an ARP response packet, where the ARP response packet carries a MAC address that is of an uplink port of the virtual bridge and that is used as the MAC address of the VM2, and the MAC address of the uplink port of the virtual bridge is a MAC address of a PF or the VF that is connected to the virtual bridge;

(2) a VLAN identifier processing setting: for a unicast packet sent by the VM1, a VLAN identifier carried in the unicast packet is removed; and

(3) a MAC address setting: for a unicast packet sent to the VM2 by the VM1, a destination MAC address of the unicast packet is changed to the MAC address of the VM2, and a changed unicast packet is sent to a virtual network function module by using a port connected to the virtual network function module.

[0061] It should be noted that a setting principle of the foregoing flow entry may be based on the OpenFlow protocol.

[0062] Step 305: The virtual bridge configures the flow entry, returns an ARP response packet to the VM1 according to an ARP pickup setting, where the ARP response packet carries a MAC address of an uplink port of the virtual bridge; and sends the ARP response packet to the switching equipment by using the uplink port.

[0063] Step 306: The switching equipment forwards the ARP response packet to the VM1.

[0064] Step 307: The VM1 receives the ARP response packet, and sets the MAC address of the VM2 to the MAC address of the uplink port of the virtual bridge.

[0065] In this embodiment of the present invention, the foregoing ARP process is defined, so that layer 2 forwarding control of VM traffic is implemented without auto-learning of a MAC forwarding table. According to the foregoing ARP process, a destination MAC address of a data packet sent by the VM1 is the MAC address of the uplink port of the virtual bridge. This ensures that the data packet is sent to the virtual bridge by using the uplink port. Then the virtual bridge forwards the data packet to the virtual network function module for processing.

[0066] Referring to FIG. 4, FIG. 4 is a schematic flowchart of sending a data packet to a VM2 by a VM1 according to an embodiment of the present invention. In this embodiment of

the present invention, a VM1 on a host 1 sends a data packet to a VM2 on a host 2. A VMM of the host 1 includes a first virtual bridge and a first virtual network function module. A network interface card of the host 1 includes at least two network ports and first switching equipment. A VMM of the host 2 includes a second virtual bridge and a second virtual network function module. A network interface card of the host 2 includes at least two network ports and second switching equipment. Specifically, structures of the host 1 and the host 2 and functions of the components are shown in FIG. 2. Details are not described herein again. A process of sending the data packet to the VM2 by the VM1 includes:

[0067] Step 401: The VM1 sends, by using a VF that is connected to the VM1, the data packet whose destination is the VM2. A destination IP address of the data packet is an IP address of the VM2, and a destination MAC address of the data packet is a MAC address of an uplink port of the first virtual bridge on the host 1. The data packet carries a VLAN identifier of the VF.

[0068] Step 402: The first switching equipment of the network interface card 1 receives the data packet, and sends the data packet to the first virtual bridge according to the destination MAC address of the data packet by using the uplink port.

[0069] Step 403: The first virtual bridge processes the data packet according to a preset flow entry, removes the VLAN identifier in the data packet, and changes the destination MAC address of the data packet to a MAC address of the VM2.

[0070] Step 404: The first virtual bridge sends a changed data packet to the first virtual network function module.

[0071] Step 405: The first virtual network function module performs network function processing on the data packet, including a security group, QoS, layer 2 tunnel encapsulation, distributed routing, and the like.

[0072] It should be noted that the first virtual network function module may perform network function processing by using various implementations in the prior art. This embodiment of the present invention imposes no limitation thereon.

[0073] Step 406: The first virtual network function module sends the processed data packet to an external physical switch of the host 1 by using a second network port, and the physical switch routes, through a network, the data packet to the host 2 on which the VM2 is

located.

[0074] It should be noted that the data packet carries the IP address and the MAC address of the VM2 and the data packet may be routed, in an existing routing manner, to the host 2 on which the VM2 is located. This embodiment of the present invention does not limit a transmission process of the data packet on the network. For example, a tunnel may be established between the host 1 on which the VM1 is located and the host 2 on which the VM2 is located, and endpoints of the tunnel are respectively the virtual network function module of the host 1 and that of the host 2, so that the tunnel required for data transmission is established between the two hosts for the VM1 and the VM2.

[0075] Step 407: The second virtual network function module of the host 2 on which the VM2 is located performs network function processing on the received data packet, and sends the processed data packet to the second virtual bridge.

[0076] Step 408: The second virtual bridge performs layer 2 forwarding according to the MAC address of the VM2, and sends, by using the switching equipment, the data packet to a VF that is connected to the VM2, so as to send the data packet to the second virtual machine.

[0077] For a data packet that is sent from an exterior of a host to a destination virtual machine on the host, the data packet is first transferred from a second network port to a virtual network function module; and then after having undergone virtual network function processing, the data packet is transferred to the destination virtual machine of the data packet by using a first network port.

[0078] The embodiment shown in FIG. 4 describes cross-host communication between virtual machines. It may be understood that a source virtual machine and a destination virtual machine of a data packet may be located on a same host. In this case, a data packet transmission process is similar to the process in FIG. 4. A difference lies in that: when the source virtual machine and the destination virtual machine of the data packet are located on a same host, a virtual network function module of the host sends the processed data packet to a virtual bridge after completing network function processing, and the virtual bridge sends the processed data packet to the destination virtual machine according to a MAC address of the destination virtual machine.

[0079] The host and an SDN controller may use a general-purpose computer device. For

example, as shown in FIG. 5, FIG. 5 is a schematic structural diagram of hardware of a computer device according to an embodiment of the present invention. The computer device 500 includes at least one processor 501, a communications bus 502, a memory 503, and at least one communications interface 504.

5 **[0080]** The processor 501 may be a general-purpose central processing unit (CPU), a microprocessor, an application-specific integrated circuit (application-specific integrated circuit, ASIC), or one or more integrated circuits configured to control program execution in the solutions of the present invention.

10 **[0081]** The communications bus 502 may include a path for transferring information between the foregoing components. The communications interface 504 uses an apparatus such as a transceiver to communicate with another device or a communications network such as an Ethernet, a radio access network (RAN), or a wireless local area network (Wireless Local Area Networks, WLAN).

15 **[0082]** The memory 503 may be a read-only memory (read-only memory, ROM) or another type of static storage device that can store static information and instructions, a random access memory (random access memory, RAM) or another type of dynamic storage device that can store information and instructions, an electrically erasable programmable read-only memory (Electrically Erasable Programmable Read-Only Memory, EEPROM), a compact disc read-only memory (Compact Disc Read-Only Memory, CD-ROM) or another
20 compact disc storage, an optical disc storage (including a compact disc, a laser disc, an optical disc, a digital versatile disc, a Blu-ray disc, or the like), a magnetic disk storage medium or another magnetic storage device, or any other medium that can be used to carry or store expected program code in a form of an instruction or a data structure and that can be accessed by a computer. No limitation is imposed thereon. The memory may exist alone and is
25 connected to the processor by using the bus. Alternatively, the memory may be integrated with the processor.

30 **[0083]** The memory 503 is configured to store application program code for executing the solutions in the present invention, and the processor 501 controls the execution. The processor 501 is configured to execute the application program code stored in the memory 503.

30 **[0084]** In a specific implementation, in an embodiment, the processor 501 may include

one or more CPUs, for example, a CPU0 and a CPU1 in FIG. 5.

[0085] In a specific implementation, in an embodiment, the computer device 500 may include multiple processors, for example, the processor 501 and a processor 508 in FIG. 5. Each of the processors may be a single-core (single-CPU) processor, or may be a multi-core (multi-CPU) processor. The processor herein may be one or more devices, circuits, and/or processing cores that are configured to process data (for example, a computer program instruction).

[0086] In a specific implementation, in an embodiment, the computer device 500 may further include an output device 505 and an input device 506. The output device 505 communicates with the processor 501, and can display information in multiple manners. For example, the output device 505 may be a liquid crystal display (liquid crystal display, LCD), a light emitting diode (light emitting diode, LED) display device, a cathode ray tube (cathode ray tube, CRT) display device, a projector (projector), or the like. The input device 506 communicates with the processor 501, and may receive a user input in multiple manners. For example, the input device 506 may be a mouse, a keyboard, a touchscreen device, a sensing device, or the like.

[0087] The computer device 500 may be a general-purpose computer device or a dedicated computer device. In a specific implementation, the computer device 500 may be a desktop computer, a portable computer, a network server, a personal digital assistant (Personal Digital Assistant, PDA), a mobile phone, a tablet computer, a wireless terminal device, a communications device, an embedded device, or a device with a structure similar to the structure in FIG. 5. This embodiment of the present invention imposes no limitation on a type of the computer device 500.

[0088] The SDN controller and the host in FIG. 2 may be the device shown in FIG. 5. The memory stores one or more software modules for implementing functions of the host and the SDN controller. The host and the SDN controller may implement, by using the processor and the program code in the memory, the packet processing method disclosed in the embodiments of the present invention.

[0089] It should be noted that the computer device shown in FIG. 5 only provides possible hardware implementations of parts in a cloud computing system. According to differences or

variations of functions of the parts in the system, hardware components of the computer device may be added or deleted to match the functions of the parts in the system.

[0090] Further, similar to the hardware apparatus shown in FIG. 5, the network interface card in the foregoing embodiments includes a processor and a memory. The processor in the network interface card executes an instruction in the memory to implement a function of the foregoing switching equipment. Details are not described again in this embodiment of the present invention.

[0091] Still further, as shown in FIG. 6, FIG. 6 is a schematic structural diagram of a cloud computing system according to an embodiment of the present invention. The cloud computing system includes at least one host 601 and an SDN controller 602. The host includes at least one network interface card. A structure of the host is shown in FIG. 2.

[0092] The host 601 includes a virtual machine monitor VMM and at least one network interface card. A first virtual machine runs on the host 601. The VMM includes a virtual bridge and a virtual network function module. The network interface card includes switching equipment and at least two network ports. A first network port supports single-root input/output virtualization. The first network port is corresponding to at least one PF and multiple VFs. The first virtual machine is connected to at least one VF of the first network port.

[0093] The SDN controller 602 is configured to receive an ARP request packet from the first virtual machine that is forwarded by the virtual bridge, and return a flow entry to the virtual bridge. The flow entry includes an ARP pickup setting and a MAC address setting. The ARP pickup setting is used to instruct the virtual bridge to construct an ARP response packet for the first virtual machine. The ARP response packet carries a MAC address that is of an uplink port of the virtual bridge and that is used as a destination MAC address of a to-be-sent data packet.

[0094] The first virtual machine is configured to send the data packet to a second virtual machine by using the VF that is connected to the first virtual machine. A destination IP address carried in the data packet is an IP address of the second virtual machine, and the destination MAC address carried in the data packet is the MAC address of the uplink port of the virtual bridge.

[0095] The switching equipment is configured to receive the data packet, and send the data packet to the virtual bridge according to the destination MAC address of the data packet by using the uplink port of the virtual bridge.

[0096] The virtual bridge is configured to receive the data packet, change the destination MAC address of the data packet to a MAC address of the second virtual machine, and send a changed data packet to the virtual network function module.

[0097] The virtual network function module is configured to perform network function processing on the changed data packet, and send the processed data packet to the second virtual machine according to the MAC address of the second virtual machine.

[0098] The multiple VFs of the first network port have different virtual local area network VLAN identifiers.

[0099] The first virtual machine is further configured to send the ARP request packet by using the VF that is connected to the first virtual machine. The ARP request packet carries a VLAN identifier of the VF that sends the ARP request packet.

[0100] The switching equipment is further configured to broadcast the ARP request packet. The ARP request packet is sent to the virtual bridge by using the uplink port.

[0101] The virtual bridge is further configured to forward the ARP request packet to the SDN controller, receive the flow entry returned by the SDN controller, and return the ARP response packet to the first virtual machine. The ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet.

[0102] The flow entry further includes a MAC address setting. The MAC address setting is used to instruct the virtual bridge to change a destination MAC address of a unicast packet to the MAC address of the second virtual machine when the virtual bridge receives the unicast packet that is sent by the first virtual machine to the second virtual machine.

[0103] Further, the flow entry further includes an output setting. The output setting is used to instruct to send the changed data packet by using a port that is connected to the virtual network function module.

[0104] The uplink port of the virtual bridge is a PF of the first network port, and the virtual bridge is connected to the network interface card by using the PF; or the uplink port of

the virtual bridge is a selected VF of the first network port, and the virtual bridge is connected to the network interface card by using the selected VF, where the selected VF that is connected to the virtual bridge is set to a promiscuous mode.

[0105] When the second virtual machine and the first virtual machine run on a same host, the virtual bridge is further configured to receive the processed data packet, and send the processed data packet to the second virtual machine according to the destination MAC address of the data packet by using the switching equipment.

[0106] When the second virtual machine and the first virtual machine run on different hosts, a tunnel is established between a virtual network function module of a host on which the first virtual machine is located and a virtual network function module of a host on which the second virtual machine is located. The processed data packet is transmitted from a second network port to the virtual network function module of the host on which the second virtual machine is located. After completing network function processing on the data packet, the virtual network function module on which the second virtual machine is located sends the data packet to the second virtual machine by using a virtual bridge of the host on which the second virtual machine is located.

[0107] An embodiment of the present invention further provides a computer storage medium that is configured to store a computer software instruction used by the devices in FIG. 2 to FIG. 6. The computer software instruction includes a program designed for executing the foregoing method embodiments. A packet processing method in a cloud computing system may be implemented by executing a stored program.

[0108] In conclusion, according to the packet processing method in a cloud computing system, the host, and the system that are provided in the embodiments of the present invention, a packet sent by a virtual machine that uses a passthrough VF can be directed to a virtual bridge on a VMM, and further a virtual network function module on the VMM performs network function processing on the data packet, so as to provide abundant network functions for a user.

[0109] Although the present invention is described with reference to the embodiments, in a process of implementing the embodiments of the present invention that claims protection, a person skilled in the art may understand and implement another variation of the disclosed

embodiments by viewing the accompanying drawings, disclosed content, and the accompanying claims. In the claims, the word "comprising" (comprising) does not exclude other constituent parts or steps. A single processor or another unit may implement several functions enumerated in the claims. Some measures are recorded in dependent claims that are
5 different from each other, but this does not mean that these measures cannot be combined to produce a better effect.

[0110] A person skilled in the art should understand that the embodiments of the present invention may be provided as a method, an apparatus (device), or a computer program product. Therefore, the present invention may use a form of hardware only embodiments,
10 software only embodiments, or embodiments with a combination of software and hardware. Moreover, the embodiments of the present invention may use a form of a computer program product that is implemented on one or more computer-usable storage media (including but not limited to a disk memory, a CD-ROM, an optical memory, and the like) that include computer-usable program code. The computer program is stored/distributed in a proper
15 medium and is provided as or used as a part of the hardware together with another hardware, or may also use another allocation form, such as by using the Internet or another wired or wireless telecommunications system.

[0111] The computer program instructions in the embodiments of the present invention may be stored in a computer readable memory that can instruct a computer or another
20 programmable data processing device to work in a specific manner. The computer program instructions may be executed to implement functions of components in the foregoing embodiments.

[0112] These computer program instructions may also be loaded onto a computer or another programmable data processing device, so that a series of operations and steps are
25 performed on the computer or the another programmable device, thereby generating computer-implemented processing. Therefore, the instructions executed on the computer or the another programmable device provide steps for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0113] Although the present invention is described with reference to specific features and
30 the embodiments thereof, obviously, various modifications and combinations may be made to

them without departing from the spirit and scope of the present invention. Correspondingly, the specification and accompanying drawings are merely illustrative description of the present invention defined by the accompanying claims, and are considered as any of or all modifications, variations, combinations or equivalents that cover the scope of the present invention. Obviously, a person skilled in the art can make various modifications and variations to the present invention without departing from the spirit and scope of the present invention. The present invention is intended to cover these modifications and variations provided that they fall within the scope of protection defined by the following claims and their equivalent technologies.

+

CLAIMS

What is claimed is:

1. A host, wherein the host comprises a virtual machine monitor VMM and at least one network interface card, a first virtual machine runs on the host, the VMM comprises a virtual
5 bridge and a virtual network function module, and the at least one network interface card comprises switching equipment and at least two network ports, wherein a first network port supports a network interface card virtualization capability, the first network port is corresponding to at least one physical function PF and multiple virtual functions VFs, and the first virtual machine is connected to at least one VF of the first network port;

10 the first virtual machine is configured to determine a destination media access control MAC address of a to-be-sent data packet, wherein the MAC address of the to-be-sent data packet is a MAC address of an uplink port of the virtual bridge;

the first virtual machine is further configured to send the data packet to a second virtual machine by using the VF that is connected to the first virtual machine, wherein a destination
15 IP address carried in the data packet is an IP address of the second virtual machine, and the destination MAC address carried in the data packet is the MAC address of the uplink port of the virtual bridge;

the switching equipment is configured to receive the data packet, and send the data packet to the virtual bridge according to the destination MAC address of the data packet by
20 using the uplink port of the virtual bridge;

the virtual bridge is configured to receive the data packet, change the destination MAC address of the data packet to a MAC address of the second virtual machine, and send a changed data packet to the virtual network function module; and

the virtual network function module is configured to perform network function
25 processing on the changed data packet, and send the processed data packet, wherein a destination MAC address of the processed data packet is the MAC address of the second virtual machine.

2. The host according to claim 1, wherein the multiple VFs of the first network port have

different virtual local area network VLAN identifiers.

3. The host according to claim 2, wherein:

the first virtual machine is further configured to send an Address Resolution Protocol ARP request packet by using the VF that is connected to the first virtual machine, wherein the
5 ARP request packet carries a VLAN identifier of the VF that sends the ARP request packet;

the switching equipment is further configured to broadcast the ARP request packet, wherein the ARP request packet is sent to the virtual bridge by using the uplink port;

the virtual bridge is further configured to forward the ARP request packet to a software-defined networking SDN controller, and receive a flow entry returned by the SDN
10 controller, wherein the flow entry comprises an ARP pickup setting, the ARP pickup setting is used to instruct the virtual bridge to construct an ARP response packet based on the ARP request packet, and the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet; and

15 the virtual bridge is further configured to return the ARP response packet to the first virtual machine, wherein the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet.

4. The host according to claim 3, wherein the flow entry further comprises a MAC
20 address setting, and the MAC address setting is used to instruct the virtual bridge to change the destination MAC address of the data packet to the MAC address of the second virtual machine when the virtual bridge receives the data packet that is sent by the first virtual machine to the second virtual machine.

5. The host according to any one of claims 1 to 4, wherein:

25 the uplink port of the virtual bridge is a PF of the first network port, and the virtual bridge is connected to the network interface card by using the PF; or

the uplink port of the virtual bridge is a selected VF of the first network port, and the virtual bridge is connected to the network interface card by using the selected VF, wherein the selected VF that is connected to the virtual bridge is set to a promiscuous mode.

30 6. The host according to any one of claims 1 to 4, wherein the second virtual machine

and the first virtual machine run on a same host; and

the virtual bridge is further configured to receive the processed data packet, and send the processed data packet to the second virtual machine according to the destination MAC address of the data packet by using the switching equipment.

5 7. The host according to any one of claims 1 to 4, wherein the second virtual machine and the first virtual machine run on different hosts; and

the virtual network function module is specifically configured to establish a tunnel between the virtual network function module and another virtual network function module on a host on which the second virtual machine is located, send the processed data packet to a second network port of the network interface card, and send, by using the tunnel, the
10 processed data packet to the host on which the second virtual machine is located, so that the another virtual network function module sends the processed data packet to the second virtual machine.

8. A packet processing method in a cloud computing system, wherein at least one host in
15 the cloud computing system comprises a virtual machine monitor VMM and at least one network interface card, a first virtual machine runs on the host, the VMM comprises a virtual bridge and a virtual network function module, and the network interface card comprises switching equipment and at least two network ports, wherein a first network port supports a network interface card virtualization capability, the first network port is corresponding to at
20 least one PF and multiple VFs, and the first virtual machine is connected to at least one VF of the first network port; and the method comprises:

determining, by the first virtual machine, a destination MAC address of a to-be-sent data packet, wherein the MAC address of the to-be-sent data packet is a MAC address of an uplink port of the virtual bridge;

25 sending, by the first virtual machine, the data packet to a second virtual machine by using the VF that is connected to the first virtual machine, wherein a destination IP address carried in the data packet is an IP address of the second virtual machine, and the destination MAC address carried in the data packet is the MAC address of the uplink port of the virtual bridge;

30 receiving, by the switching equipment, the data packet, and sending the data packet to

the virtual bridge according to the destination MAC address of the data packet by using the uplink port of the virtual bridge;

receiving, by the virtual bridge, the data packet, changing the destination MAC address of the data packet to a MAC address of the second virtual machine, and sending a changed
5 data packet to the virtual network function module; and

performing, by the virtual network function module, network function processing on the changed data packet, and sending the processed data packet, wherein a destination MAC address of the processed data packet is the MAC address of the second virtual machine.

9. The method according to claim 8, wherein the multiple VFs of the first network port
10 have different virtual local area network VLAN identifiers.

10. The method according to claim 9, wherein the determining, by the first virtual machine, a destination MAC address of a to-be-sent data packet comprises:

sending, by the first virtual machine, an ARP request packet by using the VF that is connected to the first virtual machine, wherein the ARP request packet carries a VLAN
15 identifier of the VF that sends the ARP request packet;

broadcasting, by the switching equipment, the ARP request packet, wherein the ARP request packet is sent to the virtual bridge by using the uplink port;

forwarding, by the virtual bridge, the ARP request packet to an SDN controller, and receiving a flow entry returned by the SDN controller, wherein the flow entry comprises an
20 ARP pickup setting, the ARP pickup setting is used to instruct the virtual bridge to construct an ARP response packet based on the ARP request packet, and the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet; and

returning, by the virtual bridge, the ARP response packet to the first virtual machine,
25 wherein the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet.

11. The method according to claim 10, wherein the flow entry further comprises a MAC address setting, and the MAC address setting is used to instruct the virtual bridge to change the destination MAC address of the data packet to the MAC address of the second virtual
30 machine when the virtual bridge receives the data packet that is sent by the first virtual

machine to the second virtual machine.

12. The method according to any one of claims 8 to 11, wherein:

the uplink port of the virtual bridge is a PF of the first network port, and the virtual bridge is connected to the network interface card by using the PF; or

5 the uplink port of the virtual bridge is a selected VF of the first network port, and the virtual bridge is connected to the network interface card by using the selected VF, wherein the selected VF that is connected to the virtual bridge is set to a promiscuous mode.

13. The method according to any one of claims 8 to 11, wherein the second virtual machine and the first virtual machine run on a same host, and the method further comprises:

10 receiving, by the virtual bridge, the processed data packet, and sending the processed data packet to the second virtual machine according to the destination MAC address of the data packet by using the switching equipment.

14. The method according to any one of claims 8 to 11, wherein the second virtual machine and the first virtual machine run on different hosts, and the method further
15 comprises:

establishing, by the virtual network function module, a tunnel between the virtual network function module and another virtual network function module on a host on which the second virtual machine is located, sending the processed data packet to a second network port of the network interface card, and sending, by using the tunnel, the processed data packet to
20 the host on which the second virtual machine is located, so that the another virtual network function module sends the processed data packet to the second virtual machine.

15. A cloud computing system, comprising a software-defined networking SDN controller and at least one host, wherein the host comprises a virtual machine monitor VMM and at least one network interface card, a first virtual machine runs on the host, the VMM
25 comprises a virtual bridge and a virtual network function module, and the network interface card comprises switching equipment and at least two network ports, wherein a first network port supports a network interface card virtualization capability, the first network port is corresponding to at least one PF and multiple VFs, and the first virtual machine is connected to at least one VF of the first network port;

30 the SDN controller is configured to receive an ARP request packet from the first virtual

machine that is forwarded by the virtual bridge, and return a flow entry to the virtual bridge, wherein the flow entry comprises an ARP pickup setting and a MAC address setting, the ARP pickup setting is used to instruct the virtual bridge to construct an ARP response packet for the first virtual machine, and the ARP response packet carries a MAC address that is of an uplink
5 port of the virtual bridge and that is used as a destination MAC address of a to-be-sent data packet;

the first virtual machine is configured to send the data packet to a second virtual machine by using the VF that is connected to the first virtual machine, wherein a destination IP address carried in the data packet is an IP address of the second virtual machine, and the destination
10 MAC address carried in the data packet is the MAC address of the uplink port of the virtual bridge;

the switching equipment is configured to receive the data packet, and send the data packet to the virtual bridge according to the destination MAC address of the data packet by using the uplink port of the virtual bridge;

15 the virtual bridge is configured to receive the data packet, change the destination MAC address of the data packet to a MAC address of the second virtual machine, and send a changed data packet to the virtual network function module; and

the virtual network function module is configured to perform network function processing on the changed data packet, and send the processed data packet, wherein a
20 destination MAC address of the processed data packet is the MAC address of the second virtual machine.

16. The system according to claim 15, wherein the multiple VFs of the first network port have different virtual local area network VLAN identifiers.

17. The system according to claim 16, wherein:

25 the first virtual machine is further configured to send the ARP request packet by using the VF that is connected to the first virtual machine, wherein the ARP request packet carries a VLAN identifier of the VF that sends the ARP request packet;

the switching equipment is further configured to broadcast the ARP request packet, wherein the ARP request packet is sent to the virtual bridge by using the uplink port;

30 the virtual bridge is further configured to forward the ARP request packet to the SDN

controller, and receive the flow entry returned by the SDN controller, wherein the flow entry comprises the ARP pickup setting, the ARP pickup setting is used to instruct the virtual bridge to construct the ARP response packet based on the ARP request packet, and the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used
5 as the destination MAC address of the to-be-sent data packet; and

the virtual bridge is further configured to return the ARP response packet to the first virtual machine, wherein the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet.

10 18. The system according to claim 17, wherein the flow entry further comprises the MAC address setting, and the MAC address setting is used to instruct the virtual bridge to change the destination MAC address of the data packet to the MAC address of the second virtual machine when the virtual bridge receives the data packet that is sent by the first virtual machine to the second virtual machine.

15 19. The system according to any one of claims 15 to 18, wherein:

the uplink port of the virtual bridge is a PF of the first network port, and the virtual bridge is connected to the network interface card by using the PF; or

the uplink port of the virtual bridge is a selected VF of the first network port, and the virtual bridge is connected to the network interface card by using the selected VF, wherein the
20 selected VF that is connected to the virtual bridge is set to a promiscuous mode.

20 20. The system according to any one of claims 15 to 18, wherein the second virtual machine and the first virtual machine run on a same host; and

the virtual bridge is further configured to receive the processed data packet, and send the processed data packet to the second virtual machine according to the destination MAC
25 address of the data packet by using the switching equipment.

21. The system according to any one of claims 15 to 18, wherein the second virtual machine and the first virtual machine run on different hosts; and

the virtual network function module is specifically configured to establish a tunnel between the virtual network function module and another virtual network function module on
30 a host on which the second virtual machine is located, send the processed data packet to a

second network port of the network interface card, and send, by using the tunnel, the processed data packet to the host on which the second virtual machine is located, so that the another virtual network function module sends the processed data packet to the second virtual machine.

5 22. A host, comprising a first processor, a first memory, and at least one network interface card, wherein the network interface card comprises a second processor, a second memory, and at least two network ports, a first network port supports a network interface card virtualization capability, and the first network port is corresponding to at least one PF and multiple VFs; the first memory and the second memory store instructions; the first processor
10 executes a first instruction in the first memory to implement a function of a first virtual machine, and the first virtual machine is connected to at least one VF of the first network port; the first processor executes a second instruction in the first memory to implement a function of a virtual bridge; the first processor executes a third instruction in the first memory to implement a function of a virtual network function module; and the second processor is
15 configured to execute an instruction in the second memory to implement a function of switching equipment;

 the first processor is configured to execute the first instruction in the first memory to perform the following step: determining a destination MAC address of a to-be-sent data packet, wherein the MAC address of the to-be-sent data packet is a MAC address of an uplink
20 port of the virtual bridge; and sending the data packet to a second virtual machine by using the VF that is connected to the first virtual machine, wherein a destination IP address carried in the data packet is an IP address of the second virtual machine, and the destination MAC address carried in the data packet is the MAC address of the uplink port of the virtual bridge;

 the second processor is configured to execute the instruction in the second memory to
25 perform the following step: receiving the data packet, and sending the data packet to the virtual bridge according to the destination MAC address of the data packet by using the uplink port of the virtual bridge;

 the first processor is further configured to execute the second instruction in the first memory to perform the following step: receiving the data packet, and changing the destination
30 MAC address of the data packet to a MAC address of the second virtual machine; and

the first processor is further configured to execute the third instruction in the first memory to perform the following step: performing network function processing on a changed data packet, and sending the processed data packet, wherein a destination MAC address of the processed data packet is the MAC address of the second virtual machine.

5 23. The host according to claim 22, wherein the multiple VFs of the first network port have different virtual local area network VLAN identifiers;

the first processor is further configured to execute the first instruction in the first memory to perform the following step: sending an ARP request packet by using the VF that is connected to the first virtual machine, wherein the ARP request packet carries a VLAN
10 identifier of the VF that sends the ARP request packet;

the second processor is configured to execute the instruction in the second memory to perform the following step: broadcasting the ARP request packet, and sending the ARP request packet to the virtual bridge by using the uplink port; and

the first processor is further configured to execute the second instruction in the first
15 memory to perform the following step: forwarding the ARP request packet to an SDN controller, and receiving a flow entry returned by the SDN controller, wherein the flow entry comprises an ARP pickup setting, the ARP pickup setting is used to instruct the virtual bridge to construct an ARP response packet based on the ARP request packet, and the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used
20 as the destination MAC address of the to-be-sent data packet; and returning the ARP response packet to the first virtual machine, wherein the ARP response packet carries the MAC address that is of the uplink port of the virtual bridge and that is used as the destination MAC address of the to-be-sent data packet.

24. The host according to claim 22, wherein the flow entry further comprises a MAC
25 address setting, and the MAC address setting is used to instruct the virtual bridge to change the destination MAC address of the data packet to the MAC address of the second virtual machine when the virtual bridge receives the data packet that is sent by the first virtual machine to the second virtual machine.