Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2875807 A1 2013/12/27

(21) 2 875 807

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) **Date de dépôt PCT/PCT Filing Date:** 2013/06/12
(87) **Date publication PCT/PCT Publication Date:** 2013/12/27
(85) **Entrée phase nationale/National Entry:** 2014/12/04
(86) **N° demande PCT/PCT Application No.:** US 2013/045290
(87) **N° publication PCT/PCT Publication No.:** 2013/191972
(30) **Priorité/Priority:** 2012/06/21 (US13/529,747)

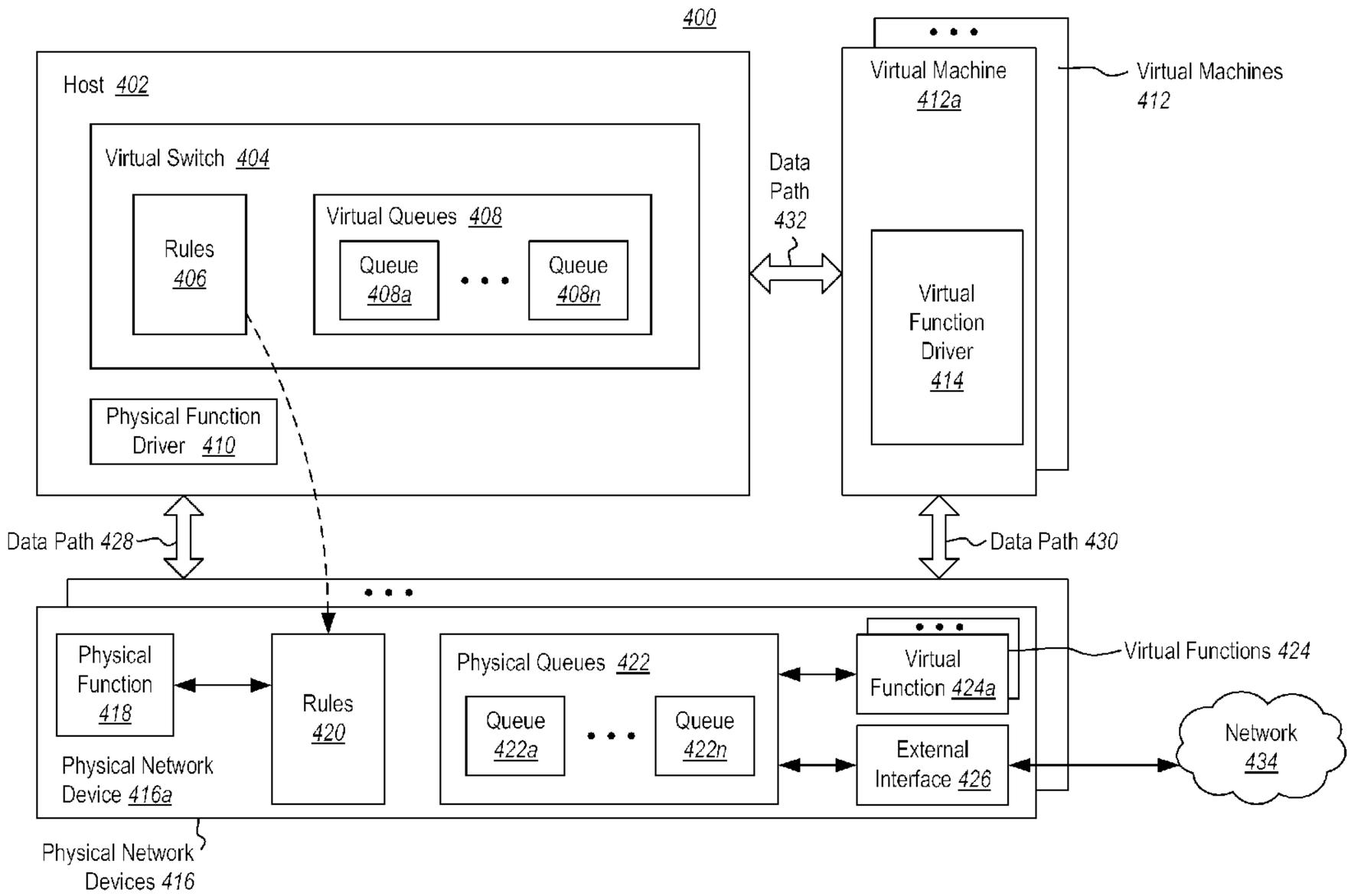
(51) **Cl.Int./Int.Cl.** ***G06F 9/50*** (2006.01)

(71) **Demandeur/Applicant:**
MICROSOFT CORPORATION, US

(72) **Inventeurs/Inventors:**
KANDULA, SRLKANTH, US;
KIM, CHANGHOON, US;
DABAGH, ALIREZA, US;
BANSAL, DEEPAK, US;
MALTZ, DAVID A., US

(74) **Agent:** SMART & BIGGAR

(54) **Titre : DECHARGEMENT DE FLUX DE MACHINE VIRTUELLE VERS DES FILES D'ATTENTE PHYSIQUES**
(54) **Title: OFFLOADING VIRTUAL MACHINE FLOWS TO PHYSICAL QUEUES**



(57) **Abrégé/Abstract:**
The present invention extends to methods, systems, and computer program products for offloading virtual machine flows to physical queues. A computer system executes one or more virtual machines, and programs a physical network device with one or more rules that manage network traffic for the virtual machines. The computer system also programs the network device to manage network traffic using the rules. In particular, the network device is programmed to determine availability of one or more physical queues at the network device that are usable for processing network flows for the virtual machines. The network device is also programmed to identify network flows for the virtual machines, including identifying characteristics of each network flow. The network device is also programmed to, based on the characteristics of the network flows and based on the rules, assign one or more of the network flows to at least one of the physical queues.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 December 2013 (27.12.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/191972 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2013/045290

(22) International Filing Date:
12 June 2013 (12.06.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/529,747 21 June 2012 (21.06.2012) US

(71) Applicant: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **KANDULA, Srlkanth**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **KIM, Changhoon**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **DABAGH, Alireza**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **BANSAL, Deepak**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **MALTZ, David A.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

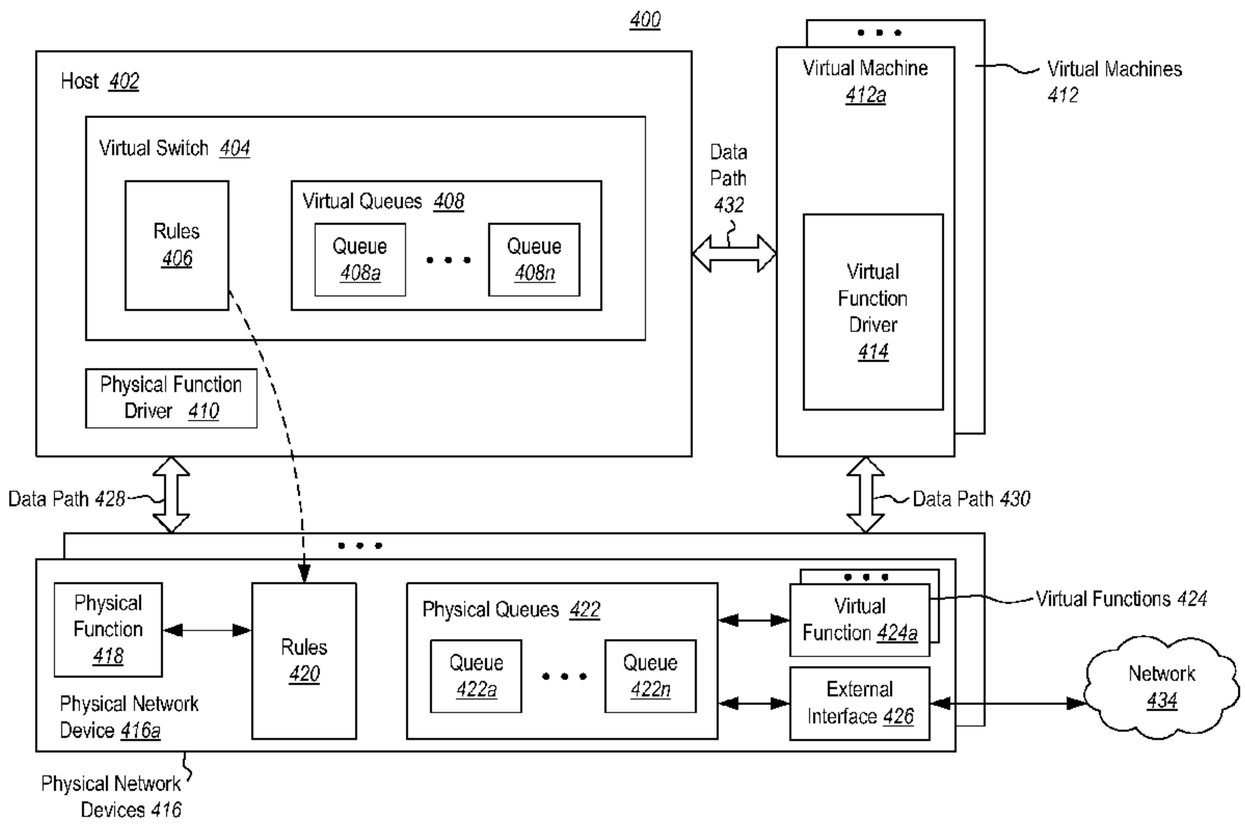
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

(54) Title: OFFLOADING VIRTUAL MACHINE FLOWS TO PHYSICAL QUEUES



(57) **Abstract**: The present invention extends to methods, systems, and computer program products for offloading virtual machine flows to physical queues. A computer system executes one or more virtual machines, and programs a physical network device with one or more rules that manage network traffic for the virtual machines. The computer system also programs the network device to manage network traffic using the rules. In particular, the network device is programmed to determine availability of one or more physical queues at the network device that are usable for processing network flows for the virtual machines. The network device is also programmed to identify network flows for the virtual machines, including identifying characteristics of each network flow. The network device is also programmed to, based on the characteristics of the network flows and based on the rules, assign one or more of the network flows to at least one of the physical queues.

WO 2013/191972 A1

OFFLOADING VIRTUAL MACHINE FLOWS TO PHYSICAL QUEUES

BACKGROUND

[0001] **1. Background and Relevant Art**

[0002] Computer systems and related technology affect many aspects of society. Indeed, the computer system's ability to process information has transformed the way we live and work. Computer systems now commonly perform a host of tasks (e.g., word processing, scheduling, accounting, etc.) that prior to the advent of the computer system were performed manually. More recently, computer systems have been coupled to one another and to other electronic devices to form both wired and wireless computer networks over which the computer systems and other electronic devices can transfer electronic data. Accordingly, the performance of many computing tasks is distributed across a number of different computer systems and/or a number of different computing environments.

[0003] Some computer systems are configured to provide virtualized environments for hosting one or more virtual machines. For example, para-virtualized execution environments include hypervisors. Hypervisors provide a parent partition (sometimes referred to as a host) and one or more child partitions. The parent partition communicates with and manages physical hardware, and is configured to run a host operating system and to manage a virtualization stack. Each child partition is configured as a "virtual machine" that runs a corresponding guest operating system.

[0004] Common scenarios in virtualization involve managing network packets among virtual machines that are executing at a virtualization host computer system, and to manage network packets flowing between the virtual machines and computers systems remote from the host computer system. As such, virtualization stacks at host operating systems may include networking virtualization stacks, including virtual switches. Virtual switches are configured to intercept, inspect, and manipulate network packets being communicated in connection with the virtual machines. Doing so, however, can be inefficient, as it can cause frequent and costly (e.g., in terms of CPU usage) context switches between the host operating system and guest operating systems and can introduce latency in network communications.

[0005] Recent developments in virtualization include Single-Root I/O Virtualization (SRIOV). SRIOV is an extension to the Peripheral Component Interconnect Express (PCIe) bus architecture that enables PCIe devices to communicate directly with child partitions. As such, SRIOV enables PCIe devices to expose themselves to child partitions / virtual machines through the hypervisor. For example, a SRIOV-compliant physical

Network Interface Card (NIC) or switch may present a physical function to the parent partition and present one or more virtual functions to corresponding child partitions. The host operating system can then include a physical function driver that communicates with the physical function, and each guest operating system can execute a virtual function driver that communicates with the corresponding virtual function. The physical NIC can then communicate network packets directly with guest operating systems (bypassing the host operating system), which can greatly improve network performance.

[0006] Despite the advances that SRIOV brings, there remain some inefficiencies in the area of network packet processing in virtualization environments.

BRIEF SUMMARY

[0007] The present invention extends to methods, systems, and computer program products for offloading virtual machine network flows to physical queues of network hardware. As such, embodiments of the present invention can enable virtual machine network traffic to pass directly between virtual machines and physical hardware, bypassing the parent partition and avoiding the inefficiencies associated with routing network traffic through the parent partition. In particular, embodiments of the present invention include configuring physical network hardware to assign network flows from virtual machines to physical queues at the physical network hardware, and potentially to assign more network flows to physical queues than the number of physical queues that exist at the physical network hardware.

[0008] In some embodiments, a method for managing network traffic includes a computer system executing one or more virtual machines. The method also includes the computer system programming a physical network device with one or more rules that are used by the physical network device to manage network traffic for the virtual machines. In particular, the physical network device is programmed to determine availability of one or more physical queues at the physical network device. The physical queues are usable for processing network flows for the virtual machines. The physical network device is also programmed to identify a plurality of network flows for the virtual machines, including identifying characteristics of each of the network flows. The physical network device is also programmed to assign one or more of the plurality of network flows to at least one of the physical queues based on the characteristics of the network flows and based on the rules.

[0009] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not

intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0010] Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0012] Figure 1 illustrates an exemplary computing system on which the principled described herein may be employed.

[0013] Figure 2 illustrates an environment in which the principles described herein may be employed.

[0014] Figure 3 illustrates a host on which the principles described herein may be employed.

[0015] Figure 4 illustrates an example computer architecture that facilitates offloading virtual machine flows to physical queues.

[0016] Figure 5 illustrates a flow chart of an example method for managing network traffic.

DETAILED DESCRIPTION

[0017] The present invention extends to methods, systems, and computer program products for offloading virtual machine network flows to physical queues of network hardware. As such, embodiments of the present invention can enable virtual machine network traffic to pass directly between virtual machines and physical hardware, bypassing the parent partition and avoiding the inefficiencies associated with routing

network traffic through the parent partition. In particular, embodiments of the present invention include configuring physical network hardware to assign network flows from virtual machines to physical queues at the physical network hardware, and potentially to assign more network flows to physical queues than the number of physical queues that exist at the physical network hardware.

[0018] First, some introductory discussion regarding general computing systems and computing environments in or on which the principles described herein may be employed will be described with respect to Figures 1-3. Then the basic principles for offloading virtual machine network flows to physical queues of network hardware will be described with respect to Figures 4 and 5.

[0019] Computing systems are now increasingly taking a wide variety of forms. Computing systems may, for example, be handheld devices, appliances, laptop computers, desktop computers, mainframes, distributed computing systems, or even devices that have not conventionally been considered a computing system. In this description and in the claims, the term "computing system" is defined broadly as including any device or system (or combination thereof) that includes at least one physical and tangible processor, and a physical and tangible memory capable of having stored thereon computer-executable instructions that may be executed by the processor(s). The memory may take any form and may depend on the nature and form of the computing system. A computing system may be distributed over a network environment and may include multiple constituent computing systems.

[0020] Embodiments described herein may comprise or utilize a special purpose or general-purpose computer including computer hardware, such as, for example, one or more processors and system memory. For example, Figure 1 illustrates an exemplary computing system 100. As illustrated in Figure 1, in its most basic configuration, computing system 100 typically includes at least one processing unit 102 and memory 104. The memory 104 may be physical system memory, which may be volatile, non-volatile, or some combination of the two. The term "memory" may also be used herein to refer to non-volatile mass storage such as physical storage media. If the computing system 100 is distributed, the processing, memory and/or storage capability may be distributed as well. As used herein, the term "module" or "component" can refer to software objects or routines that execute on the computing system 100. The different components, modules, engines, and services described herein may be implemented as objects or processes that execute on the computing system 100 (e.g., as separate threads).

[0021] In the description that follows, embodiments are described with reference to acts that are performed by one or more computing systems, such as the computing system 100. If such acts are implemented in software, one or more processors of the associated computing system that performs the acts direct the operation of the computing system in response to having executed computer-executable instructions. An example of such an operation involves the manipulation of data. Within the context of the computing system 100, computer-executable instructions (and the manipulated data) may be stored in the memory 104. Computing system 100 may also contain communication channels 108 that allow the computing system 100 to communicate with other message processors over, for example, network 110.

[0022] Embodiments described herein also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are physical storage media. Computer-readable media that carry computer-executable instructions are transmission media.Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: computer storage media and transmission media.

[0023] Computer storage media includes recordable-type storage media, such as RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0024] A “network” is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmissions media can include a network (e.g., the network 110) and/or data links which can be used to carry or desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media.

[0025] Further, upon reaching various computer system components, program code means in the form of computer-executable instructions or data structures can be transferred automatically from transmission media to computer storage media (or vice versa). For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a "NIC"), and then eventually transferred to computer system RAM and/or to less volatile computer storage media at a computer system. Thus, it should be understood that computer storage media can be included in computer system components that also (or even primarily) utilize transmission media.

[0026] Computer-executable instructions comprise, for example, instructions and data which, when executed at a processor, cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code.Although the subject matter is described herein using language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described herein. Rather, the features and acts described herein are disclosed as example forms of implementing the claims.

[0027] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, tablets, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

[0028] Figure 2 abstractly illustrates an environment 200 in which the principles described herein may be employed. The environment 200 includes multiple clients 210 interacting with a system 210 using an interface 202. The environment 200 is illustrated as having three clients 201A, 201B and 201C, although the ellipses 201D represents that

the principles described herein are not limited to the number of clients interfacing with the system 210 through the interface 202. The system 210 may provide services to the clients 201 on-demand, and thus the number of clients 201 receiving services from the system 210 may vary over time.

[0029] One or more of the clients 201 may, for example, be structured as described above in accordance with computing system 100 of Figure 1. Alternatively or in addition, one or more of the clients 201 may be an application or other software module that interfaces with the system 210 through the interface 202. The interface 202 may be an application program interface (API) that is defined in such a way that any computing system or software entity that is capable of using the API may communicate with the system 210.

[0030] The system 210 may be a distributed system, although this is not required. In one embodiment, the system 210 is a cloud computing environment. Cloud computing environments may be distributed, although not required, and may even be distributed internationally and/or have components possessed across multiple organizations.

[0031] In this description and the following claims, "cloud computing" is defined as a model for enabling on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). The definition of "cloud computing" is not limited to any of the other numerous advantages that can be obtained from such a model when properly deployed.

[0032] For instance, cloud computing is currently employed in the marketplace so as to offer ubiquitous and convenient on-demand access to the shared pool of configurable computing resources. Furthermore, the shared pool of configurable computing resources can be rapidly provisioned via virtualization and released with low management effort or service provider interaction, and then scaled accordingly.

[0033] A cloud computing model can be composed of various characteristics, such as on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service, and so forth. A cloud computing model may also come in the form of various service models such as, for example, Software as a Service ("SaaS"), Platform as a Service ("PaaS"), and Infrastructure as a Service ("IaaS"). The cloud computing model may also be deployed using different deployment models such as private cloud, community cloud, public cloud, hybrid cloud, and so forth. In this description and in the claims, a "cloud computing environment" is an environment in which cloud computing is employed.

[0034] As depicted, the system 210 includes multiple hosts 211, that are each capable of running virtual machines. Although the system 200 might include any number of hosts 211, there are three hosts 211A, 211B and 211C illustrated in Figure 2, with the ellipses 211D representing that the principles described herein are not limited to the exact number of hosts that are within the system 210. There may be as few as one, with no upper limit. Furthermore, the number of hosts may be static, or might dynamically change over time as new hosts are added to the system 210, or as hosts are dropped from the system 210. Each of the hosts 211 may be structured as described above for the computing system 100 of Figure 1.

[0035] Each host is capable of running one or more, and potentially many, virtual machines. For instance, Figure 3 abstractly illustrates a host 300 in further detail. As an example, the host 300 might represent any of the hosts 211 of Figure 2. In the case of Figure 3, the host 300 is illustrated as operating three virtual machines 310 including virtual machines 310A, 310B and 310C. However, the ellipses 310D once again represents that the principles described herein are not limited to the number of virtual machines running on the host 300. There may be as few as zero virtual machines running on the host with the only upper limit being defined by the physical capabilities of the host 300.

[0036] During operation, the virtual machines emulates a fully operational computing system including an at least an operating system, and perhaps one or more other applications as well. Each virtual machine is assigned to a particular client, and is responsible to support the desktop environment for that client.

[0037] The virtual machine generates a desktop image or other rendering instructions that represent a current state of the desktop, and then transmits the image or instructions to the client for rendering of the desktop. For instance, referring to Figures 2 and 3, suppose that the host 300 of Figure 3 represents the host 211A of Figure 2, and that the virtual machine 310A is assigned to client 201A (referred to herein as “the primary example”), the virtual machine 310A might generate the desktop image or instructions and dispatch such instructions to the corresponding client 201A from the host 211A via a service coordination system 213 and via the system interface 202.

[0038] As the user interacts with the desktop at the client, the user inputs are transmitted from the client to the virtual machine. For instance, in the primary example and referring to Figures 2 and 3, the user of the client 201A interacts with the desktop, and

the user inputs are transmitted from the client 201 to the virtual machine 310A via the interface 201, via the service coordination system 213 and via the host 211A.

[0039] The virtual machine processes the user inputs and, if appropriate, changes the desktop state. If such change in desktop state is to cause a change in the rendered desktop, then the virtual machine alters the image or rendering instructions, if appropriate, and transmits the altered image or rendered instructions to the client computing system for appropriate rendering. From the prospective of the user, it is as though the client computing system is itself performing the desktop processing.

[0040] The host 300 includes a hypervisor 320 that emulates virtual resources for the virtual machines 310 using physical resources 321 that are abstracted from view of the virtual machines 310. The hypervisor 321 also provides proper isolation between the virtual machines 310. Thus, from the perspective of any given virtual machine, the hypervisor 320 provides the illusion that the virtual machine is interfacing with a physical resource, even though the virtual machine only interfaces with the appearance (e.g., a virtual resource) of a physical resource, and not with a physical resource directly. In Figure 3, the physical resources 321 are abstractly represented as including resources 321A through 321F. Examples of physical resources 321 including processing capacity, memory, disk space, network bandwidth, media drives, and so forth.

[0041] The host 300 may operate a host agent 302 that monitors the performance of the host, and performs other operations that manage the host. Furthermore, the host 300 may include other components 303, such as a virtual switch as described later.

[0042] Referring back to Figure 2, the system 200 also includes services 212. In the illustrated example, the services 200 include five distinct services 212A, 212B, 212C, 212D and 212E, although the ellipses 212F represents that the principles described herein are not limited to the number of service in the system 210. A service coordination system 213 communicates with the hosts 211 and with the services 212 to thereby provide services requested by the clients 201, and other services (such as authentication, billing, and so forth) that may be prerequisites for the requested service.

[0043] Turning now to Figure 4, Figure 4 illustrates an example computer architecture 400 that facilitates offloading virtual machine flows to physical queues. As depicted, computer architecture 400 includes host 402, one or more virtual machines 412 (including virtual machine 412a), and one or more physical network devices 416 (including physical network device 416a).

[0044] Host 402 is configured to provide a virtualization environment. In some embodiments, host 402 may correspond to host 300 of Figure 300. For example, host 402 may include a parent partition (which executes a host operating system) and one or more child partitions. Each child partition can be viewed as providing a virtualized hardware environment for executing a corresponding virtual machine, such as virtual machine 412a. Host 402 may be used a part of a cloud computing environment that hosts virtual machines on behalf of tenants.

[0045] Each of virtual machines 412 (including virtual machine 412a) executes one or more virtualized applications, such as an operating system, application software, etc. Each of virtual machines 412 is capable of sending and receiving network packets. For example, each of virtual machines 412 includes a network stack (e.g., a TCP/IP stack) and is capable of sending and/or receiving network packets and other information through host 402 over data path 432 and/or through physical network devices 416 over data path 430. As such, virtual machines 412 can create network flows.

[0046] Each physical network device 416 is connected to other computer systems and/or networks using one or more external interfaces. Figure 4 depicts that physical network device 416a is connected to network 434 using external interface 426. Physical network devices 416 can include any appropriate type of physical networking hardware, such as NICs, switches, etc.

[0047] In addition, each physical network device 416 comprises physical hardware that is compatible with a virtualized environment. For example, Figure 4 depicts that physical network device 416a presents virtual functions 424 to virtual machines 412. In particular, physical network device 416a may present one or more virtual functions to each of virtual machines 412. For example, Figure 4 depicts that physical network device 416a presents virtual function 424a to virtual machine 412a. Each of virtual machine 412, in turn, includes a corresponding virtual function driver. For example, Figure 4 depicts that virtual machine 412a includes virtual function driver 414. As such, each of virtual machines 412 can access its corresponding virtual function 424 over data path 430, and can use data path 430 to communicate network packets with physical network device 416a without routing the network packets through host 402. Doing so can reduce processor usage and network latency when compared to routing network packets through host 402.

[0048] In addition, Figure 4 also depicts that physical network device 416 a presents physical function 418 to host 402. Figure 4 also depicts that host 402 includes a corresponding physical function driver 410, and that data path 428 connects physical

function 418 at physical network device 416a and physical function driver 410 at host 402. As such, physical function 418 and physical function driver 410 can operate for exchange of network packets between physical network device 416a and host 402.

[0049] As indicated previously, physical NIC 110 may, in some embodiments, comprise PCIe hardware that is SRIOV-compliant. In such embodiments, one or more of virtual functions 424 or physical function 418 may comprise PCIe functions. However, it will be appreciated that the principles described herein may be applicable to a variety of hardware devices, and are not limited to SRIOV-compliant devices or to PCIe devices.

[0050] Each of physical network devices 416 can include one or more physical queues, which can be used by physical network devices 416 when processing network flows that are associated with virtual machines 412. For example, Figure 4 depicts that physical network device 416a includes physical queues 422, including queue 422a and any additional number (i.e., zero or more) of additional physical queues, as represented by the horizontal ellipses and queue 422n. According to one or more embodiments, host 402 configures one or more of physical network devices 416 to manage use of its physical queues when processing network flows for virtual machines 412. As depicted, for example, virtual switch 404 at host 402 can include rules 406. Using rules 406, virtual switch 404 can program physical network device 416a with rules 420, and can program physical network device 416a to manage network flow assignments to physical queues 422 based on those rules. Rules 420 may be identical to rules 406, may be altered in some manner, and/or may include a subset of rules 406. As such, physical network device 416a can be configured to efficiently handle network flows from virtual machines 412, including making assignments of network flows to physical queues 422, without involving host 402 for every network flow.

[0051] Rules 420 can include rules that enable physical network device 416a to assign a number network flows to physical queues 422 that is greater in number than a number of queues present at physical queues 422. In a simple example, network traffic from virtual machines 412 may involve eight active network flows, but physical network device 416a may use rules 420 to assign these eight flows to only four available queues in physical queues 422. Physical network device 416a can be configured to make network flow to queue assignments based on characteristics of the flows, and/or based on classifications of the flows. In some embodiments, physical network device 416a places network flows into different classifications based on characteristics of the flows and based on rules 420. In some additional or alternative embodiments, physical network device 416a places network

flows into different classifications based on suggestions made by virtual machines 412. For example, virtual machine 412a may attach some attribute to a flow, or may communicate a suggested classification to physical function 418 separate from the flow.

[0052] Rules 420 can enable various types of queue assignment algorithms. For example, rules 420 may specify that a plurality of network flows having a relatively low traffic level maybe assigned together on a single physical queue, while flows having a relatively high traffic level are to each be assigned exclusively to corresponding physical queue. In another example, rules 420 may specify that a plurality of flows having similar or compatible requirements are be combined on the same queue. For example, if network packets of a plurality of flows are to be paced (rate limited) at a similar rate, those flows may be assigned together on a single physical queue. Other similar or compatible requirements may include priority (e.g., grouping flows of low priority together on a single queue), quality of service (QoS) (e.g., grouping flows with low QoS requirements together on a single queue), etc. Rules 420 may also specify that flows from the same virtual machine are to be grouped onto a single physical queue or group of physical queues. As such, the embodiments herein can facilitate the partitioning of hardware resources among virtual machines 412.

[0053] In some embodiments, physical network devices 416 and virtual switch 404 can work together to balance execution of network flows there between. For example, Figure 4 depicts that virtual switch 404 can include software-based virtual queues 408 (including queue 408a and any additional number (i.e., zero or more) of additional queues, as represented by the horizontal ellipses and queue 408n). As such, some network flows may be assigned to physical queues 422, and some flows may be assigned to virtual queues 408. One will appreciate that physical queues 422 may provide faster, more granular, and/or more reliable performance than virtual queues 408. As such, network flows may be classified into flows that should be assigned to physical queues 422 to take advantage of the faster, more granular, and/or more reliable performance at physical network device 416a, and flows that may be assigned to virtual queues 408 at host 402 because fast, granular, and/or reliable performance may not be as important for these flows. Such an assignment may be suggested by virtual machines 412, and/or may be made by physical network devices 416 and/or virtual switch 404.

[0054] In some embodiments, a flow may pass through a plurality of physical network devices 416 (e.g., a NIC and a switch), and host 402 can program each physical network device to handle the flow independently. For example, one physical network device may

be programmed to assign the flow to a single physical queue at the device, while another physical network device may be programmed to assign combine the flow with other flows at a single physical queue at the device.

[0055] Figure 5 illustrates a flowchart of a method 500 for managing network traffic. Method 500 will be described with respect to the components and data of computer architecture 400.

[0056] Method 500 includes an act of executing one or more virtual machines (act 502). For example, host 402 can execute virtual machines 412, which can include virtual machine 412a. In some embodiments, act 502 can include executing the virtual machine(s) in a para-virtualized manner, including using one or more SRIOV-compliant physical network devices. As such, at least one physical network device (e.g., physical network device 418) may present a virtual function (e.g., virtual function 424a) to virtual machine 412a, and virtual machine 412a may include a corresponding virtual function driver (e.g., virtual function driver 414) for communicating network packets directly with the physical network device.

[0057] Method 500 also includes an act of programming a physical network device with one or more rules, the one or more rules being configured to manage network traffic for the one or more virtual machines (act 504). For example, virtual switch 404 can program physical network device 416a with rules 420. Rules 420 can be a copy of, or be based on, rules 406 at virtual switch 404. Rules 420 can be configured to enable physical network device 416a to make assignments between network flows associated with virtual machines 412 and physical queues 422 at physical network device 416a.

[0058] Method 500 also includes an act of programming the physical network device to manage network traffic (act 506). For example, virtual switch 404 can configure physical network device 416a to make flow assignments based on rules 420. In some embodiments, programming physical network device 416a to manage network traffic occurs as a consequence of programming physical network device 416a with rules 420. In other embodiments, programming physical network device 416a to manage network traffic includes expressly programming physical network device 416a with additional computer-executable instructions and/or additional configuration settings.

[0059] Act 506 includes programming the physical network device to determine availability of one or more physical queues at the physical network device, the one or more physical queues being usable for processing network flows for the one or more

virtual machines (act 508). For example, physical network device 416a can be configured to identify physical queues 422, including a present availability of physical queues 422.

[0060] Act 506 includes programming the physical network device to identify a plurality of network flows for the one or more virtual machines, including identifying one or more characteristics of each of the plurality of network flows (act 510). For example, physical network device 416a can be configured to identify network flows that are associated with virtual machines 412. Physical network device 416a can also be configured to analyze characteristics of the flows, categorization suggestions from virtual machines 412, or any other appropriate information, to classify or otherwise categorize the flows.

[0061] Act 506 includes programming the physical network device to, based on the one or more characteristics of each of the plurality of network flows and based on the one or more rules, assign one or more of the plurality of network flows to at least one of the one or more physical queues (act 512). For example, based on rules 420, and based on characteristics and categorizations identified in act 510, physical network device 416a can assign the flows to physical queues 422. In doing so, physical network device 416a may assign a number of flows to physical queues 422 that exceeds the number of physical queues. For example, physical network device 416a may assign flows having similar characteristics, compatible priorities or traffic loads, etc. to the same physical queue. Additionally or alternatively, physical network device 416a may work with virtual switch 404 to assign a first subset of flows to virtual queues 408 at virtual switch 404 and a second subset of flows to physical queues 422 at physical network device 416a.

[0062] Accordingly the embodiments described herein can improve network performance and utilization of physical hardware by enabling a physical network device to make assignments between flows and physical queues. When making such assignments, the embodiments described herein can enable the physical hardware to process a greater number of flows with physical queues than the number of physical queues are available. Additionally or alternatively, when making such assignments, the embodiments described herein can enable the physical hardware to balance processing of flows between physical queues and virtual queues.

[0063] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing

description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

What is claimed:

1. A virtualization host computer system (402) that is configured to manage network traffic for one or more virtual machines (412) that are executing at the virtualization host computer system, the virtualization host computer system comprising:

one or more processors;

one or more physical network devices (416); and

one or more computer-readable storage media having stored thereon computer-executable instructions that, when executed by the one or more processors, cause the virtualization host computer system to execute a virtual switch (404), the virtual switch being configured to program each of the one or more physical network devices with one or more corresponding rules (420) and to perform the following:

determine availability of one or more physical queues (422) at the physical network device, the one or more physical queues being usable for processing network flows for the one or more virtual machines;

identify a plurality of network flows for the one or more virtual machines, including identifying one or more characteristics of each of the plurality of network flows; and

based on the one or more characteristics of each of the plurality of network flows and based on the one or more rules, assigning one or more of the plurality of network flows to at least one of the one or more physical queues.

2. The virtualization host computer system as recited in claim 1, wherein assigning one or more of the plurality of network flows to at least one of the one or more physical queues comprises:

assigning at least two of the plurality of network flows to a single physical queue.

3. The virtualization host computer system as recited in claim 1, wherein the plurality of network flows are greater in number than the one or more physical queues, and wherein assigning one or more of the plurality of network flows to at least one of the one or more physical queues comprises:

assigning all of the plurality of network flows to the one or more physical queues, such that at least one of the one or more physical queues is assigned more than one of the network flows.

4. The virtualization host computer system as recited in claim 1, wherein the plurality of network flows are greater in number than the one or more physical queues, and wherein assigning one or more of the plurality of network flows to at least one of the one or more physical queues comprises:

assigning a first subset of the plurality of network flows to the one or more physical queues; and

assigning a second subset of the plurality of network flows to one or more software-based queues at the virtual switch.

5. The virtualization host computer system as recited in claim 1, wherein identifying one or more characteristics of each of the plurality of network flows includes:

identifying that at least two of the plurality of network flows have similar characteristics; and

based on the at least two of the plurality of network flows having similar characteristics, identifying that the at least two of the plurality of network flows can be assigned to a single physical queue.

6. The virtualization host computer system as recited in claim 5, wherein the at least two of the plurality of network flows having similar characteristics comprises the at least two of the plurality of network flows having similar rate limiting characteristics.

7. A method, implemented at a computer system (400) that includes one or more processors and one or more physical network devices (416), for managing network traffic, the method comprising:

executing one or more virtual machines (412);

programming a physical network device (416a) with one or more rules (420), the one or more rules being configured to manage network traffic for the one or more virtual machines; and

programming the physical network device to manage network traffic, including the following:

determining availability of one or more physical queues (422) at the physical network device, the one or more physical queues being usable for processing network flows for the one or more virtual machines;

identifying a plurality of network flows for the one or more virtual machines, including identifying one or more characteristics of each of the plurality of network flows; and

based on the one or more characteristics of each of the plurality of network flows and based on the one or more rules, assigning one or more of the plurality of network flows to at least one of the one or more physical queues.

8. The method as recited in claim 7, wherein the plurality of network flows are greater in number than the one or more physical queues, and wherein assigning one or more of the plurality of network flows to at least one of the one or more physical queues comprises:

assigning all of the plurality of network flows to the one or more physical queues, such that at least one of the one or more physical queues is assigned more than one of the network flows.

9. The method as recited in claim 7, wherein the plurality of network flows are greater in number than the one or more physical queues, and wherein assigning one or more of the plurality of network flows to at least one of the one or more physical queues comprises:

assigning a first subset of the plurality of network flows to the one or more physical queues; and

assigning a second subset of the plurality of network flows to one or more software-based queues at the virtual switch.

10. A virtualization host computer system (402) that is configured to manage network traffic for one or more virtual machines (412) that are executing at the virtualization host computer system, the virtualization host computer system comprising:

one or more processors;

one or more physical network devices (416); and

one or more computer-readable storage media having stored thereon computer-executable instructions that, when executed by the one or more processors, cause the virtualization host computer system to execute a virtual switch (404), the virtual switch being configured to program each of the one or

more physical network devices with one or more corresponding rules (420) and to perform the following:

determine availability of one or more physical queues (422) at the physical network device, the one or more physical queues being usable for processing network flows for the one or more virtual machines;

identify a plurality of network flows for the one or more virtual machines, including identifying one or more characteristics of each of the plurality of network flows, the plurality of network flows being greater in number than the one or more physical queues; and

based on the one or more characteristics of each of the plurality of network flows and based on the one or more rules, assigning the plurality of network flows to at least one of the one or more physical queues, such that at least one of the one or more physical queues is assigned more than one of the plurality of network flows.

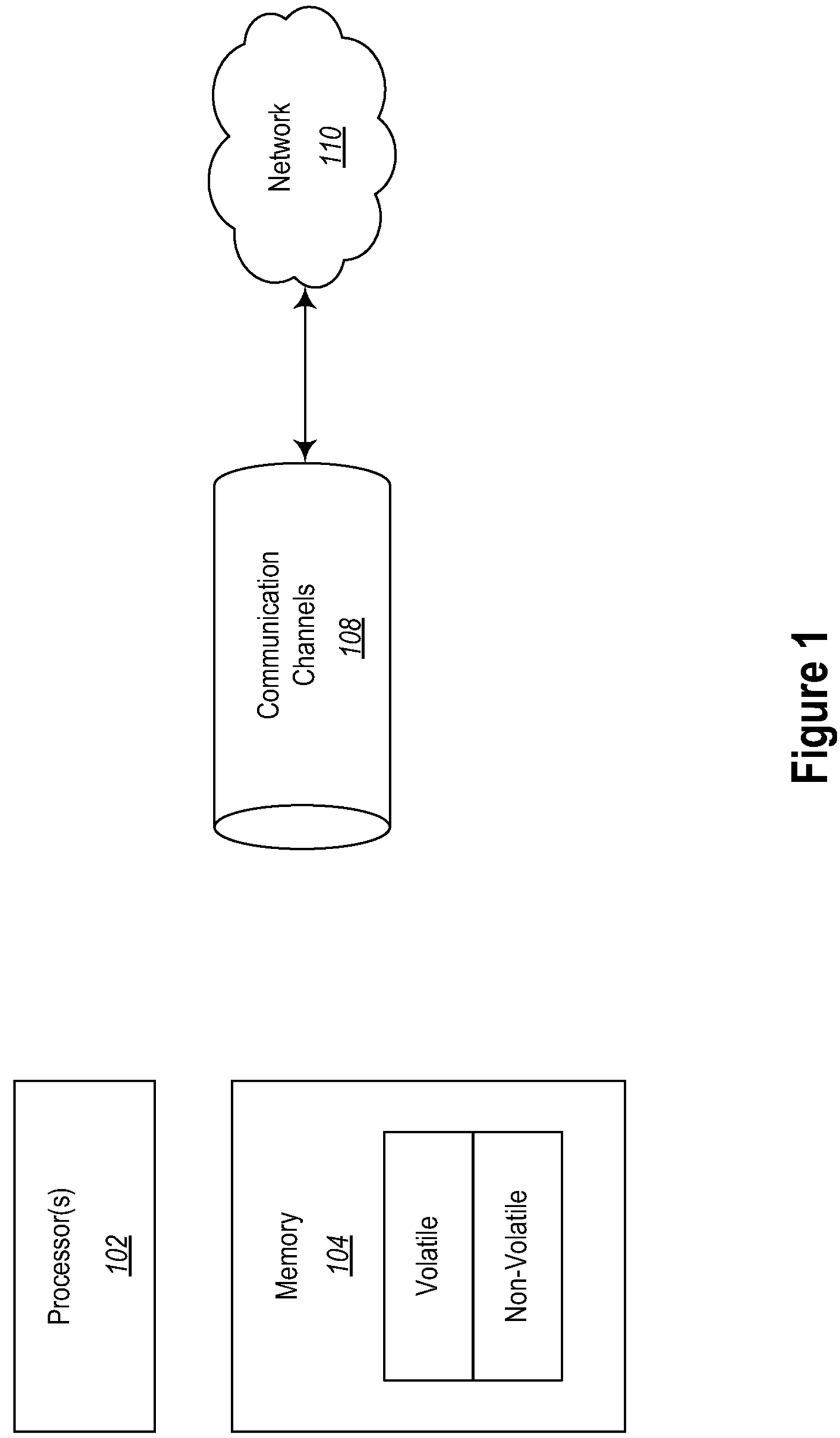

Computing System
100
Network
110
Communication Channels
108
Processor(s)
102
Memory
104
Volatile
Non-Volatile

Figure 1

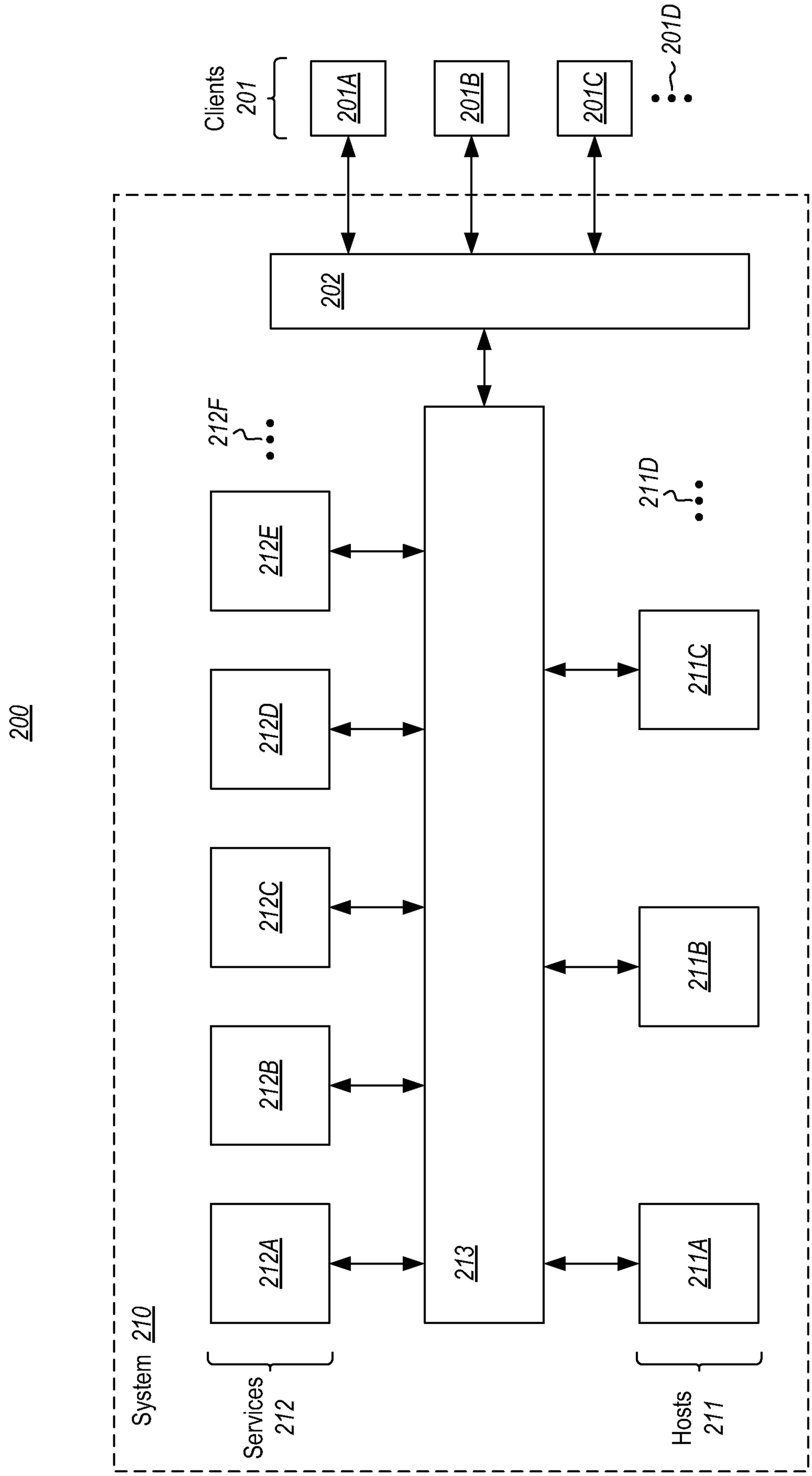

200
System 210
Clients 201
201A
201B
201C
201D
202
Services 212
212A
212B
212C
212D
212E
212F
213
Hosts 211
211A
211B
211C
211D

Figure 2

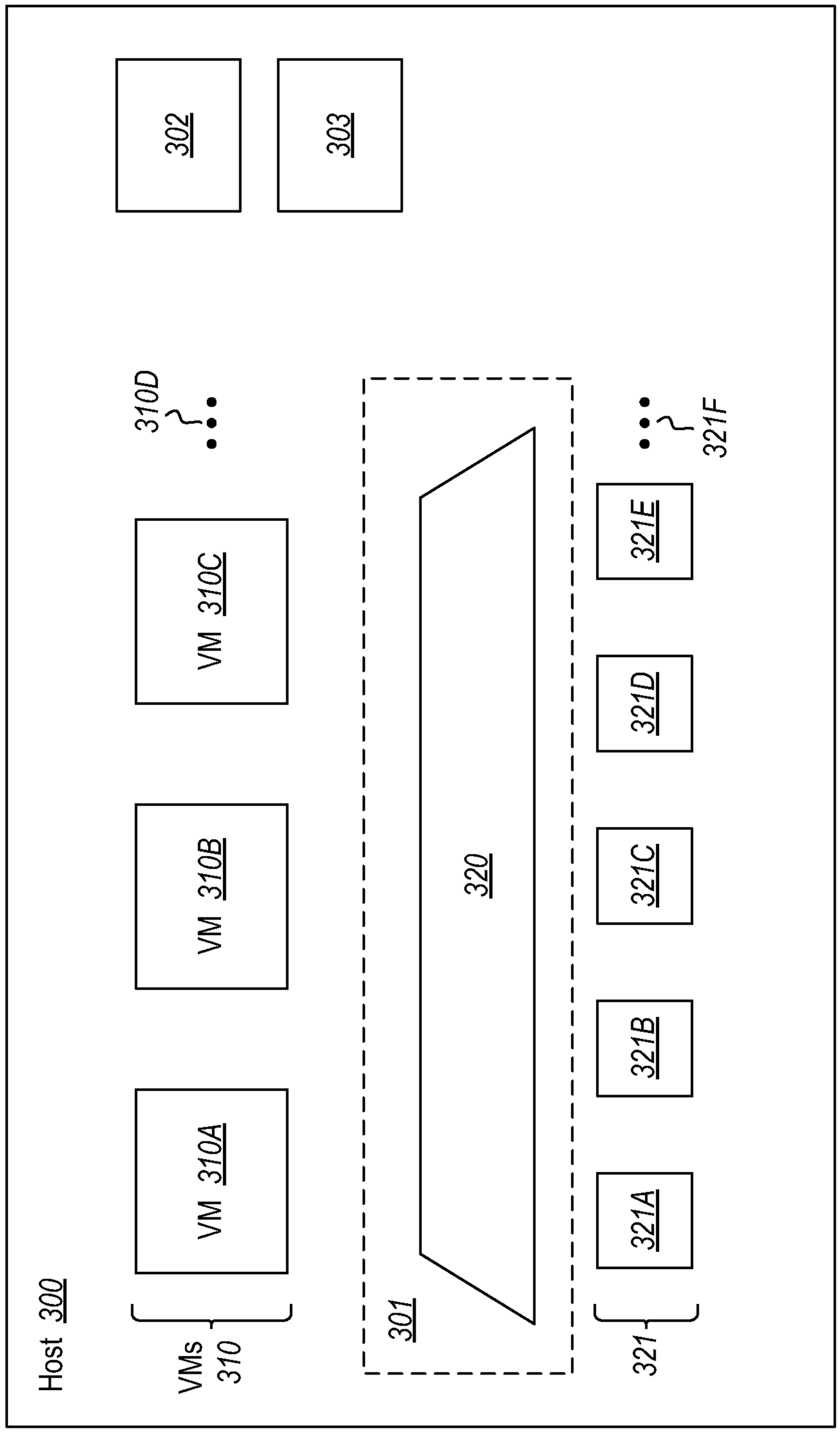

Host 300
VMs 310
VM 310A
VM 310B
VM 310C
310D
301
320
321
321A
321B
321C
321D
321E
321F
302
303

Figure 3

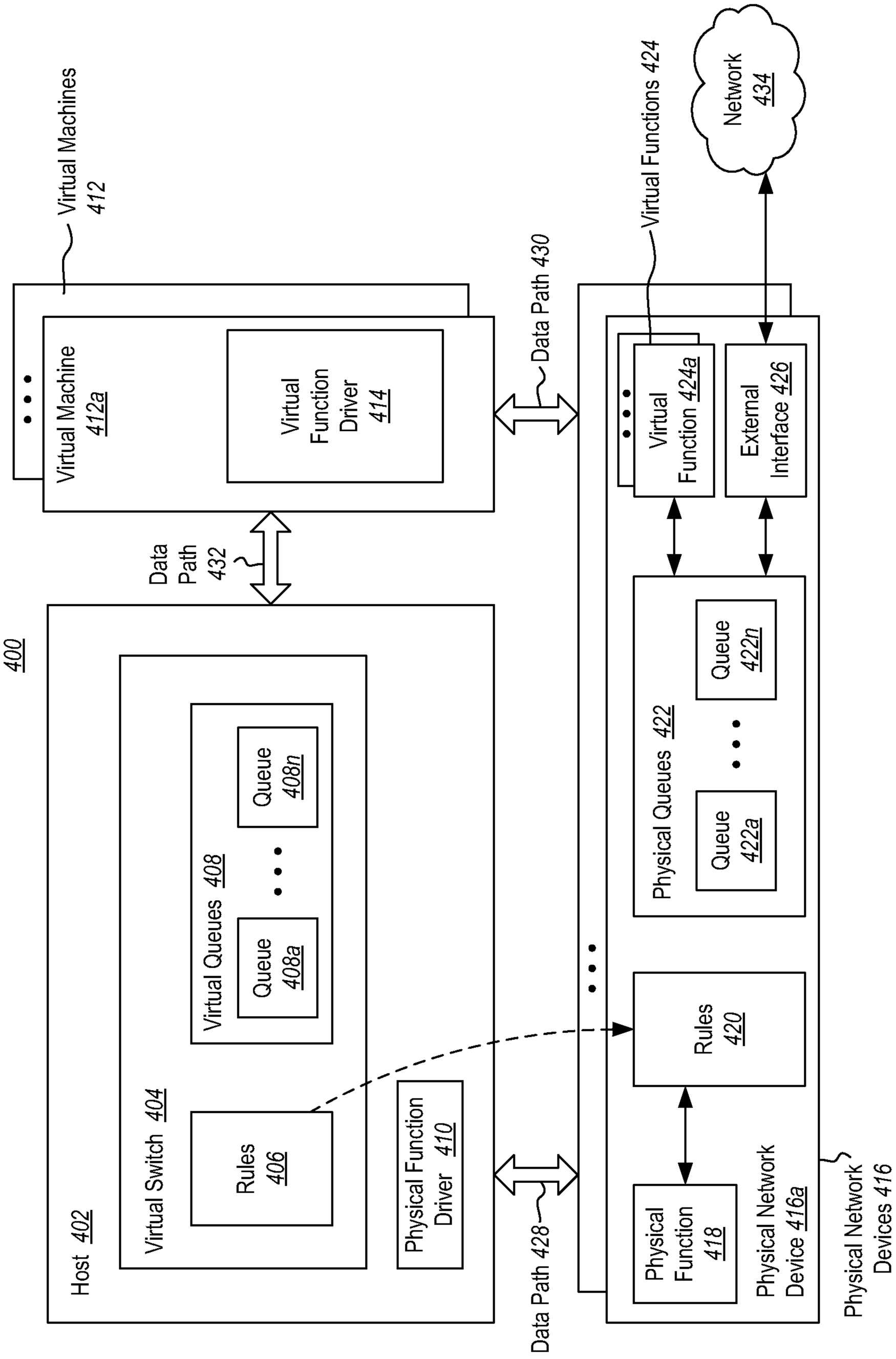

Host 402
400
Virtual Switch 404
Rules 406
Virtual Queues 408
Queue 408a
Queue 408n
Physical Function Driver 410
Data Path 432
Virtual Machines 412
Virtual Machine 412a
Virtual Function Driver 414
Data Path 428
Data Path 430
Physical Network Devices 416
Physical Network Device 416a
Physical Function 418
Rules 420
Physical Queues 422
Queue 422a
Queue 422n
Virtual Functions 424
Virtual Function 424a
External Interface 426
Network 434

Figure 4

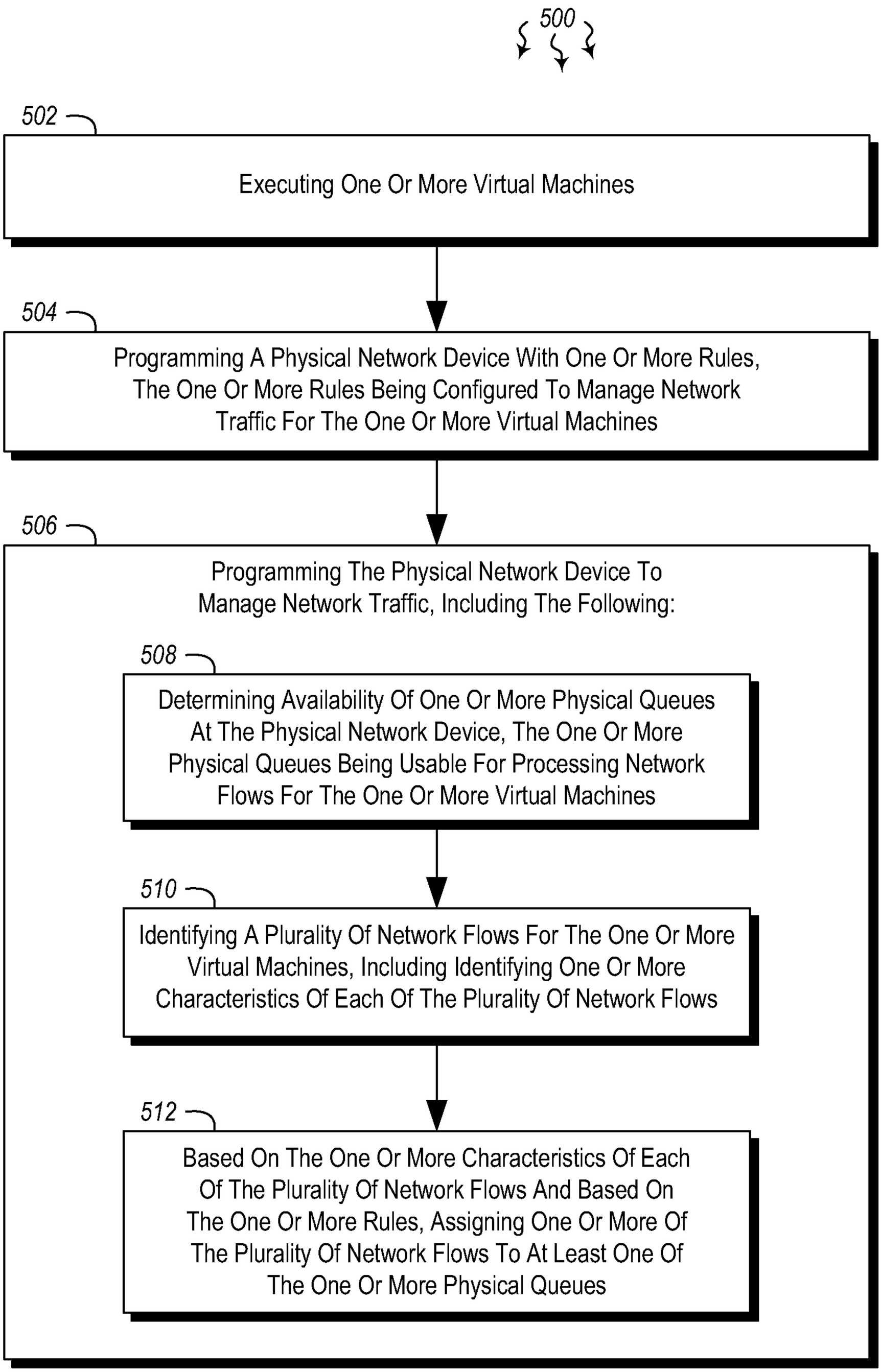
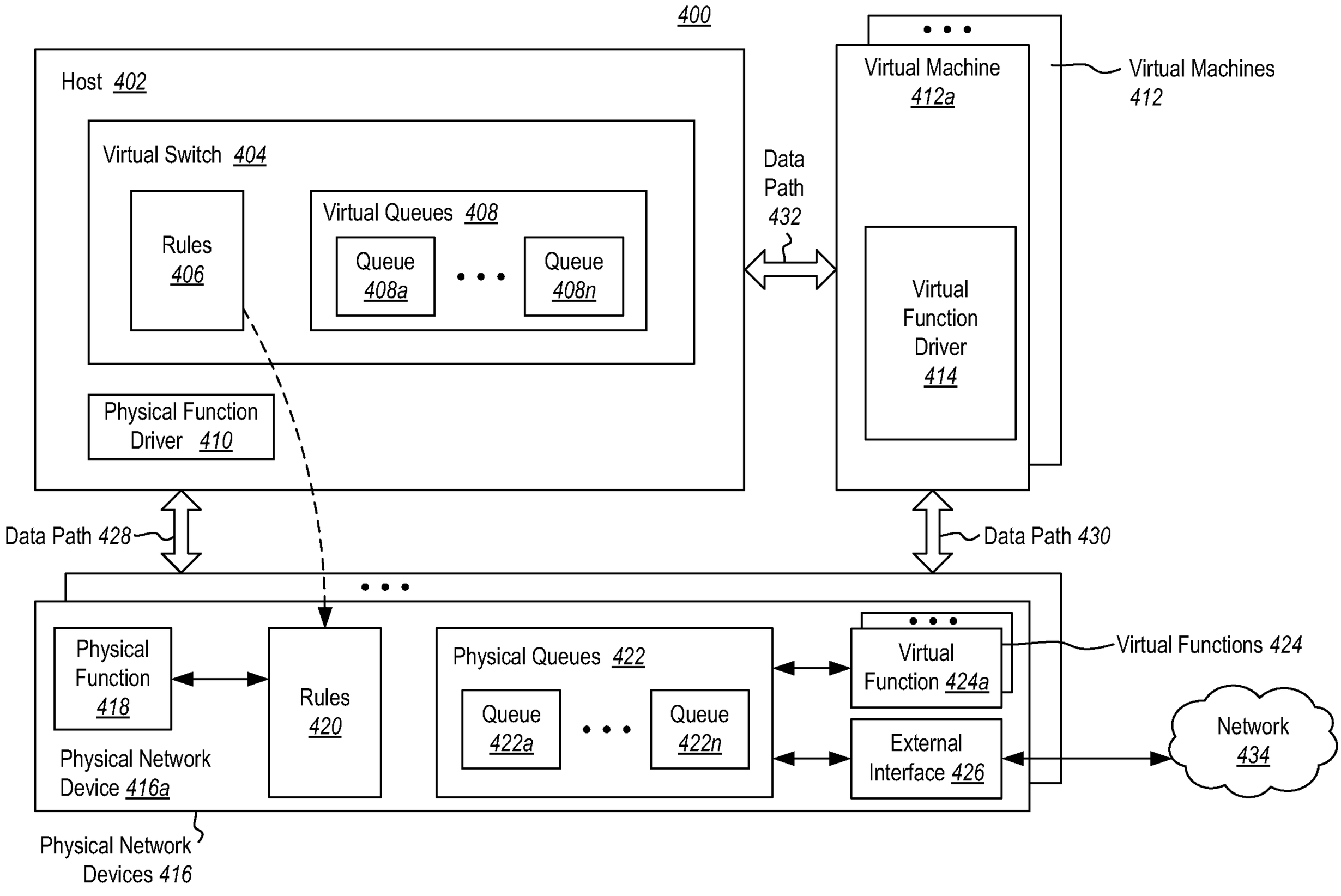

500
502
Executing One Or More Virtual Machines
504
Programming A Physical Network Device With One Or More Rules, The One Or More Rules Being Configured To Manage Network Traffic For The One Or More Virtual Machines
506
Programming The Physical Network Device To Manage Network Traffic, Including The Following:
508
Determining Availability Of One Or More Physical Queues At The Physical Network Device, The One Or More Physical Queues Being Usable For Processing Network Flows For The One Or More Virtual Machines
510
Identifying A Plurality Of Network Flows For The One Or More Virtual Machines, Including Identifying One Or More Characteristics Of Each Of The Plurality Of Network Flows
512
Based On The One Or More Characteristics Of Each Of The Plurality Of Network Flows And Based On The One Or More Rules, Assigning One Or More Of The Plurality Of Network Flows To At Least One Of The One Or More Physical Queues

Figure 5


400
Host 402
Virtual Switch 404
Rules 406
Virtual Queues 408
Queue 408a
Queue 408n
Physical Function Driver 410
Data Path 432
Virtual Machine 412a
Virtual Machines 412
Virtual Function Driver 414
Data Path 428
Data Path 430
Physical Function 418
Rules 420
Physical Queues 422
Queue 422a
Queue 422n
Physical Network Device 416a
Virtual Function 424a
Virtual Functions 424
External Interface 426
Network 434
Physical Network Devices 416