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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 April 2014 (03.04.2014)

(10) International Publication Number
WO 2014/052099 A2

(51) International Patent Classification: Not classified

(21) International Application Number: PCT/US20 13/060207

(22) International Filing Date: 17 September 2013 (17.09.201 3)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/705,618 25 September 2012 (25.09.2012) US

(71) Applicant: **A10 NETWORKS, INC.** [US/US]; 3 West Plumeria Drive, San Jose, California 95134 (US).

(72) Inventors: **SANKAR, Swaminathan;** 3 West Plumeria Drive, San Jose, California 95134 (US). **KARAMPURWALA, Hasnain;** 3 West Plumeria Drive, San Jose, California 95134 (US). **GUPTA, Rahul;** 3 West Plumeria Drive, San Jose, California 95134 (US). **KAMAT, Gurudeep;** 3 West Plumeria Drive, San Jose, California 95134 (US). **SAMPAT, Rishi;** 3 West Plumeria Drive, San Jose, California 95134 (US). **JALAN, Rajkumar;** 3 West Plumeria Drive, San Jose, California 95134 (US).

(74) Agents: **KLINE, Keith** et al; Carr & Ferrell LLP, 120 Constitution Drive, Menlo Park, California 94025 (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, SA, 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).

[Continued on next page]

(54) Title: LOAD DISTRIBUTION IN DATA NETWORKS

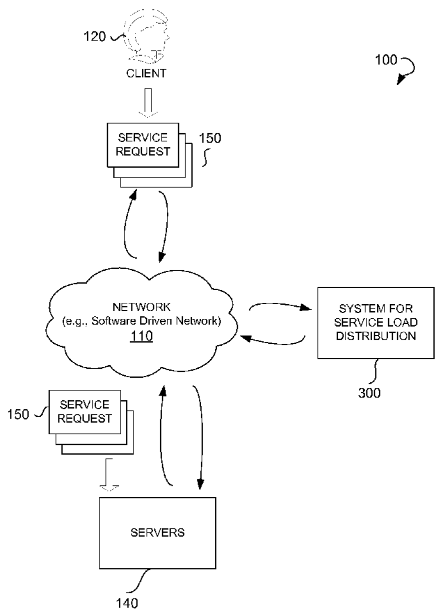
120
CLIENT
100
SERVICE REQUEST
150
NETWORK (e.g., Software Driven Network) 110
SYSTEM FOR SERVICE LOAD DISTRIBUTION
300
150
SERVICE REQUEST
SERVERS
140

FIG. 1

(57) Abstract: Provided are methods and systems for load distribution in a data network. A method for load distribution in the data network may comprise retrieving network data associated with the data network and service node data associated with one or more service nodes. The method may further comprise analyzing the retrieved network data and service node data. Based on the analysis, a service policy may be generated. Upon receiving one or more service requests, the one or more service requests may be distributed among the service nodes according to the service policy.

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

LOAD DISTRIBUTION IN DATA NETWORKS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This patent application claims the priority benefit of U.S. provisional patent application No. 61/705,618, filed Sep. 25, 2012, the disclosure of which is incorporated herein by reference.

TECHNICAL FIELD

[0002] This disclosure relates generally to data processing, and, more specifically, to load distribution in software driven networks (SDN).

BACKGROUND

[0003] The approaches described in this section could be pursued but are not necessarily approaches that have previously been conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section.

[0004] In a typical load balancing scenario, a service hosted by a group of servers is front-ended by a load balancer (LB) (also referred to herein as a LB device) which represents this service to clients as a virtual service. Clients needing the service can address their packets to the virtual service using a virtual Internet Protocol (IP) address and a virtual port. For example, www.example.com:80 is a service that is being load balanced and there is a group of servers that host this service. An LB can be configured with a virtual IP (VIP) e.g. 100.100.100.1 and virtual port (VPort) e.g. Port 80, which, in turn, are mapped to the IP addresses and port numbers of the servers handling this service. The Domain Name Service (DNS) server handling this domain can be configured to send packets to the VIP and VPort associated with this LB.

[0005] The LB will inspect incoming packets and based on the policies/algorithms will choose a particular server from the group of servers, modify the packet if necessary and forward the packet towards the server. On the way back from the server (optional), the LB will get the packet, modify the packet if necessary and forward the packet back towards the client.

[0006] There is often a need to scale up or scale down the LB service. For example, the LB service may need to be scaled up or down based on time of the day e.g. days vs. nights, weekdays vs. weekends. For example, fixed-interval software updates may result in predictable network congestions and, therefore, the LB service may need to be scaled up to handle the flash crowd phenomenon and scaled down subsequently. The popularity of the service may necessitate the need to scale up the service. These situations can be handled within the LB when the performance characteristics of the LB device can handle the scaling adjustments needed.

[0007] However, in many cases the performance needs to be increased to beyond what a single load balancing device can handle. Typical approaches for this include physical chassis-based solutions, where cards can be inserted and removed to handle the service requirements. These approaches have many disadvantages which include the need to pre-provision space, power, and price for a chassis for future needs. Additionally, a single chassis can only scale up to the maximum capacity of its cards. To cure this deficiency, one can attempt to stack LB devices and send traffic between the devices as needed. However, this approach may also have disadvantages such as the link between the devices becoming the bottleneck, increased latencies as packets have to traverse multiple LBs to reach the entity that will eventually handle the requests.

[0008] Another existing solution is to add multiple LB devices, create individual VIPs on each device for the same servers in the backend and use the DNS to distribute the load among them. When another LB needs to be added, another entry is added to the DNS database. When an LB needs to be removed, the corresponding entry is removed from the DNS database. However, this approach has the following issues. DNS records are cached and hence addition/removal of LBs may take time before they are effective. This is especially problematic when an LB is removed as data directed to the LB can be lost. The distribution across the LBs is very coarse and not traffic aware e.g. one LB may be overwhelmed while other LBs may be idle, some clients may be heavier users and end up sending requests to the same LB, and so forth. The distribution between LBs may not be LB capacity aware e.g. LBl may be a much more powerful device than LB2. Thus, the existing solutions to solve this problem all have their disadvantages.

SUMMARY

[0009] This summary is provided to introduce a selection of concepts in a simplified form that are further described in the Detailed Description below. 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] The present disclosure is related to approaches for load distribution in a data network. Specifically, a method for load distribution in a data network may comprise retrieving network data associated with the data network and service node data associated with one or more service nodes. The method may further comprise analyzing the retrieved network data and service node data. Based on the analysis, a service policy may be generated. The generated service policy may be provided to devices associated with the data network. Upon receiving one or more service requests, the one or more service requests may be distributed among the one or more service nodes according to the service policy.

[0011] According to another approach of the present disclosure, there is provided a system for load distribution in a data network. The system may comprise a cluster master. The cluster master may be configured to retrieve and analyze network data associated with the data network and service node data associated with one or more service nodes. Based on the analysis, the cluster master may generate a service policy and provide the generated service policy to devices associated with the data network. The system may further comprise a traffic classification engine. The traffic classification engine may be configured to receive the service policy from the cluster master. Upon receiving one or more service requests, the traffic classification engine may distribute the service requests among one or more service nodes according to the service policy.

Furthermore, the system may comprise the one or more service nodes. The service nodes may be configured to receive the service policy from the cluster master and receive the one or more service requests from the traffic classification engine. The service nodes may process the one or more service requests according to the service policy.

[0012] In another approach of the present disclosure, the cluster master may reside within the traffic classification engine layer or the service node layer. Additionally the traffic classification engine may, in turn, reside within the service node layer.

[0013] In further example embodiments of the present disclosure, the method steps are stored on a machine-readable medium comprising instructions, which when implemented by one or more processors perform the recited steps. In yet further example embodiments, hardware systems, or devices can be adapted to perform the recited steps. Other features, examples, and embodiments are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Embodiments are illustrated by way of example, and not by limitation, in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

[0015] FIG. 1 shows an environment within which a method and a system for service load distribution in a data network can be implemented, according to an example embodiment.

[0016] FIG. 2 is a process flow diagram showing a method for service load distribution in a data network, according to an example embodiment.

[0017] FIG. 3 is a block diagram showing various modules of a system for service load distribution in a data network, according to an example embodiment.

[0018] FIG. 4 is a scheme for service load distribution of a data network, according to an example embodiment.

[0019] FIG. 5 shows a diagrammatic representation of a computing device for a machine in the example electronic form of a computer system, within which a set of instructions for causing the machine to perform any one or more of the methodologies discussed herein can be executed.

DETAILED DESCRIPTION

[0020] The following detailed description includes references to the accompanying drawings, which form a part of the detailed description. The drawings show illustrations in accordance with example embodiments. These example embodiments, which are also referred to herein as "examples," are described in enough detail to enable those skilled in the art to practice the present subject matter. The embodiments can be combined, other embodiments can be utilized, or structural, logical, and electrical changes can be made without departing from the scope of what is claimed. The following detailed description is therefore not to be taken in a limiting sense, and the scope is defined by the appended claims and their equivalents. In this document, the terms "a" and "an" are used, as is common in patent documents, to include one or more than one. In this document, the term "or" is used to refer to a nonexclusive "or," such that "A or B" includes "A but not B," "B but not A," and "A and B," unless otherwise indicated.

[0021] The present disclosure relates to efficient ways of implementing load balancing by having an LB service and a data network, such as an SDN, work together to deliver packets to multiple LBs. Because the SDN is aware of the requirements of the LB service, it can efficiently distribute traffic to the LBs. This approach allows the same virtual service to be hosted on multiple LBs, without needing any DNS changes. There are minimal to no latency impacts since the packets are delivered directly to the LB that handles them. Fine-grained distribution of flows to the LBs can be achieved based on the LBs capabilities, network capabilities and current loads. This approach also supports scaling up/down of services as needed as well as facilitating management and operation of the load balancing by administrators.

[0022] In some example embodiments, a protocol can be running between the LBs and SDN elements that make the SDN and the LB exchange information on how to distribute traffic, dynamically inserting forwarding rules to influence packet path selection on devices that are capable of performing such forwarding. Algorithms controlled by the LBs can be implemented on routers, switches and other devices to influence traffic steering.

[0023] To ensure distribution of data flow in a network of heterogeneous switches from multiple vendors, additional technologies can be used. These technologies may utilize a controller to compute paths between sources and destination and program the flows on the network devices between the sources and destination. This property can be leveraged to program flows intelligently to scale out/in the load balancing implementation in the network based on demand, availability of resources, and so forth.

[0024] As LBs activate and deactivate based on the requirements, such as for example, load increases or configurations changes, the LBs can update the controller and have the controller make changes to the flows in the network. In an example embodiment, in case there is no appropriate external controller, the LB may itself act as the controller and may directly make changes to the flows in the network. Similarly, the controller can work with the LB to inform the LB of network loads and other inputs, health of devices in the network, and so forth to assist the LBs with making decisions.

[0025] The techniques of the embodiments disclosed herein may be implemented using a variety of technologies. For example, the methods described herein may be implemented in software executing on a computer system or in hardware utilizing either a combination of microprocessors or other specially designed application-specific integrated circuits (ASICs), programmable logic devices, or various combinations thereof. In particular, the methods described herein may be implemented by a series of computer-executable instructions residing on a storage medium such as a disk drive, or computer-readable medium. It should be noted that methods disclosed herein can be implemented by a computer (e.g., a desktop computer, a tablet computer, a laptop computer, and a server), game console, handheld gaming device, cellular phone, smart phone, smart television system, and so forth.

[0026] As outlined in the summary, the embodiments of the present disclosure refer to load distribution in an SDN. As referred herein, an SDN is a network that allows managing network services through abstraction of lower level functionality by decoupling a control plane that makes decisions as to where a service request, e.g. traffic from a client to a server, is to be sent from a data plane responsible for forwarding the service request to the selected destination based on the decision of the control plane. The data plane may reside on the network hardware or software devices and the control plane may be executed through the software. Such separation of the planes of the SDN may enable network virtualization, since commands or control rules may be executed by the software. The SDN may be configured to deliver client service requests or host service requests to virtual machines and physical devices, e.g. servers.

[0027] The control plane may be configured to ascertain the health and other data associated with the SDN and virtual machines, for example, by real time data network applets. The control plane may leverage the real time data network applets and other means to gauge service responsiveness on the virtual machines, monitor the total connections, central processing unit utilization, and memory as well as network connectivity on the virtual machines and use that information to influence the load distribution decisions and forwarding on the data plane. .

[0028] Furthermore, the control plane may comprise a service policy engine configured to analyze the collected health data and, based on the analysis, translate the health data into service policies. The service policies may include policies to enhance, i.e. scale out or scale down, the number of virtual machines, traffic classification engines, or backend servers, to remedy or repair failed virtual machines, to secure virtual machines, to introduce new virtual machines, to remove virtual machines, and so forth. Effectively, the service policies may influence load balancing, high availability as well as programming the SDN network. Therefore, based on the service policies, the SDN may scale out or scale down the use of traffic distribution devices through periods or dynamic loads and thereby optimize network resources respectively. The traffic distribution devices may be scaled out or scaled down based, for example, on time of the day. Furthermore, fixed-interval software updates may result in predictable network congestions and the load balancing may need to be scaled out to handle the flash crowd phenomenon and scaled down subsequently. Additionally, the popularity of the service may necessitate the need to scale up the service.

[0029] The SDN may comprise a controller enabling programmable control of routing the service requests, such as network traffic, without requiring physical access to network switches. In other words, the controller may be configured to steer the traffic across the network to server pools or virtual machine pools. The service policy engine may communicate with the controller and inject the service policies into the controller. The controller, in turn, may steer traffic across the network devices, such as server or virtual machines, according to the service policies.

[0030] In an example embodiment, the service data plane of the SDN may be configured as an application delivery controller (ADC). The control plane may communicate with the ADC by managing a set of service policies mapping service requests to one or more ADCs. The ADC may then relay the service requests to a backend server over a physical or logical network, namely over a server pool or a virtual machine pool.

[0031] Referring now to the drawings, FIG. 1 illustrates an environment 100 within which a method and a system for load distribution in an SDN can be implemented. The environment 100 may include a network 110, a client 120, a system 300 for load distribution, and servers 140. The client 120 may include a user or a host associated with the network 110.

[0032] The network 110 may include the Internet or any other network capable of communicating data between devices. Suitable networks may include or interface with any one or more of, for instance, a local intranet, a PAN (Personal Area Network), a LAN (Local Area Network), a WAN (Wide Area Network), a MAN (Metropolitan Area Network), a virtual private network (VPN), a storage area network (SAN), a frame relay connection, an Advanced Intelligent Network (AIN) connection, a synchronous optical network (SONET) connection, a digital Tl, T3, El or E3 line, Digital Data Service (DDS) connection, DSL (Digital Subscriber Line) connection, an Ethernet connection, an ISDN (Integrated Services Digital Network) line, a dial-up port such as a V.90, V.34 or V.34bis analog modem connection, a cable modem, an ATM (Asynchronous Transfer Mode) connection, or an FDDI (Fiber Distributed Data Interface) or CDDI (Copper Distributed Data Interface) connection. Furthermore, communications may also include links to any of a variety of wireless networks, including WAP (Wireless Application Protocol), GPRS (General Packet Radio Service), GSM (Global System for Mobile Communication), CDMA (Code Division Multiple Access) or TDMA (Time Division Multiple Access), cellular phone networks, GPS (Global Positioning System), CDPD (cellular digital packet data), RIM (Research in Motion, Limited) duplex paging network, Bluetooth radio, or an IEEE 802.11-based radio frequency network. The network 110 can further include or interface with any one or more of an RS-232 serial connection, an IEEE-1394 (Firewire) connection, a Fiber Channel connection, an IrDA (infrared) port, a SCSI (Small Computer Systems Interface) connection, a USB (Universal Serial Bus) connection or other wired or wireless, digital or analog interface or connection, mesh or Digi® networking. The network 110 may include a network of data processing nodes that are interconnected for the purpose of data communication. The network 110 may include an SDN. The SDN may include one or more of the above network types. Generally the network 110 may include a number of similar or dissimilar devices connected together by a transport medium enabling communication between the devices by using a predefined protocol. Those skilled in the art will recognize that the present disclosure may be practiced within a variety of network configuration environments and on a variety of computing devices.

[0033] As shown in FIG. 1, the client 120 may send service requests 150 to backend servers 140. The service requests 150 may include an HTTP request, a video streaming request, a file download request, a transaction request, a conference request, and so forth. The servers 140 may include a web server, a wireless application server, an interactive television server, and so forth. The system 300 for load distribution may balance flow of the service requests 150 among traffic forwarding devices of the network 110. The system 300 for load distribution may analyze the flow of the service requests 150 and determine which and how many traffic forwarding devices of the network 110 are needed to deliver the service requests 150 to the servers 140.

[0034] FIG. 2 is a process flow diagram showing a method 200 for service load distribution in an SDN, according to an example embodiment. The method 200 may be performed by processing logic that may comprise hardware (e.g., decision making logic, dedicated logic, programmable logic, and microcode), software (such as software running on a general-purpose computer system or a dedicated machine), or a combination of both.

[0035] The method 200 may commence with receiving network data associated with the SDN at operation 202. In an example embodiment, the network data associated with the SDN may be indicative of the health of the SDN, processing unit utilization, number of total connections, memory status, network connectivity, backend server capacity, and so forth. At operation 204, the method may comprise retrieving service node data associated with one or more service nodes. In an example embodiment, the one or more service nodes may include a virtual machine and a physical device. The service node data may be indicative of health of the node, dynamic state, node processing unit utilization, node memory status, network connectivity of the service nodes, responsiveness of the one or more service nodes, and so forth.

[0036] At operation 206, the retrieved network data and service node data may be analyzed. Based on the analysis, a service policy may be generated at operation 208. The service policy may include one or more of the following: a service address, a service node address, a traffic distribution policy, a service node load policy, and so forth. The method may further comprise providing, i.e.

pushing, the generated service policy to devices associated with the data network. The devices associated with the data network may include the service nodes and traffic classification engines.

[0037] The method 200 may continue with providing the generated service policy to the devices associated with the data network at operation 210. Upon receiving one or more service requests at operation 212, , the one or more service requests may be distributed among the one or more service nodes according to the service policy at operation 214. In an example embodiment, the method 200 may comprise developing, based on the analysis, a further service policy. The further service policy may be associated with scaling out, scaling down, remedying, removing services associated with the one or more service nodes, introducing a new service associated with the one or more service nodes, and so forth.

[0038] In an example embodiment, the method 200 may comprise performing health checks of a backend server by the devices associated with the data network. In further example embodiments, the method 200 may comprise scaling up or scaling down service nodes, backend servers, traffic classification engines, and cluster masters in a graceful manner with minimum to no disruption to the traffic flow. Furthermore, the services may be scaled up or scaled down in a graceful manner with minimum to no disruption to traffic flow. In the event of scaling up or scaling down of the service node, the service requests may be redirected to one or more other service nodes to continue processing data associated with the service request. In further example embodiments, the method 200 may comprise optimizing reverse traffic from backend servers to the service node handling the service.

[0039] FIG. 3 shows a block diagram illustrating various modules of an exemplary system 300 for service load distribution in an SDN. The system 300 may comprise a cluster of devices eligible as a cluster master. The system 300 may comprise a cluster master 305 elected from these devices. The cluster master 305 may be configured to keep track of the SDN and retrieve network data associated with the SDN. In an example embodiment, the network data may include one or more of the following: a number of total connections, processing unit utilization, a memory status, a network connectivity, backend server capacity, and so forth. Furthermore, the cluster master 305 may be configured to keep track of the service nodes and retrieve service node data associated with one or more service nodes. The service node data may include one or more of the following: health, dynamic state, responsiveness of the one or more service nodes, and so forth. In other words, the cluster master 305 may keep track of the health of the network and each service node associated with the system 300. The cluster master 305 may analyze the retrieved network data and service node data. Based on the analysis, the cluster master 305 may generate a service policy. The service policy may include a service address, a service node address, a service node load policy, a traffic distribution policy also referred to as a traffic mapping, and so forth. The cluster master 305 may provide the generated service policy to the devices associated with the data network, such as service nodes and traffic classification engines.

[0040] In an example embodiment, the cluster master 305 may be further configured to develop, based on the analysis, a further service policy. The further policy may be associated with scaling out, scaling down, remedying, removing devices, such as service nodes, traffic classification engines, backend servers and so forth, introducing new service nodes, traffic classification engines, backend servers, and so forth.

[0041] In an example embodiment, the cluster master 305 may be further configured to facilitate an application programmable interface (not shown) for a network administrator to enable the network administrator to develop, based on the analysis, a further service policy using the retrieved network data and service node data and analytics. This approach may allow application developers to write directly to the network without having to manage or understand all the underlying complexities and subsystems that compose the network.

[0042] In a further example embodiment, the cluster master 305 may include a backup unit (not shown) configured to replace the cluster master in case of a failure of the cluster master 305.

[0043] The system 300 may comprise a traffic classification engine 310. The traffic classification engine 310 may be implemented as one or more software modules, hardware modules, or a combination of hardware and software. The traffic classification engine 310 may include an engine configured to monitor data flows and classify the data flows based on one or more attributes associated with the data flows, e.g. uniform resource locators (URLs), IP addresses, port numbers, and so forth. Each resulting data flow class can be specifically designed to implement a certain service for a client. In an example embodiment, the cluster master 305 may send a service policy to the traffic classification engine 310. The traffic classification engine 310 may be configured to receive the service policy from the cluster master 305. Furthermore, the traffic classification engine 310 may be configured to receive one or more incoming service requests 315, e.g. incoming data traffic from routers or switches (not shown). Typically, the data traffic may be distributed from the routers or switches to each of the traffic classification engines 310 evenly. In an example embodiment, a router may perform a simple equal-cost multi-path (ECMP) routing to distribute the traffic equally to all the traffic classification engines 310. The traffic classification engines 310 may distribute the one or more service requests among one or more service nodes 320 according to the service policy. The traffic may be distributed to the one or more service nodes 320 in an asymmetric fashion. The traffic to the service nodes 320 may be direct or through a tunnel (IP-in-IP or other overlay techniques). The traffic classification engine 310 may be stateless or stateful, may act on a per packet basis, and direct each packet of the traffic to the corresponding service node 320. When there is a change in the service nodes state, the cluster master 305 may send a new service policy, such as a new traffic map, to the traffic classification engine 310.

[0044] The system 300 may comprise the one or more service nodes 320. The one or more service nodes 320 may include a virtual machine or a physical device that may serve a corresponding virtual service to which the traffic is directed. The cluster master 305 may send the service policy to the service nodes 320. The service nodes 320 may be configured to receive the service policy from the cluster master 305. Furthermore, the service nodes 320 may receive, based on the service policy, the one or more service requests 315 from the traffic classification engine 310. The one or more service nodes 320 may process the received one or more service requests 315 according to the service policy. The processing of the one or more service requests 315 may include forwarding the one or more service requests 315 to one or more backend destination servers (not shown). Each service node 320 may serve one or more virtual services. The service nodes 320 may be configured to send the service node data to the cluster master 305.

[0045] According to further example embodiment, an existing service node may redirect packets for existing flows to another service node if it is the new owner of the flow based on the redistribution of flows to the service nodes. In addition, a service node taking over the flow may redirect packets to the service node that was the old owner for the flows under consideration, for cases where the flow state needs to be pinned down to the old owner to maintain continuity of service.

[0046] Furthermore, in an example embodiment, the cluster master 305 may perform a periodic health check on the service nodes 320 and update the service nodes 320 with a service policy, such as a traffic map. When there is a change in the traffic assignment and a packet of the data traffic in a flow reaches a service node 320, the service node 320 may redirect the packet to another service node. Redirection may be direct or through a tunnel (e.g. IP-in-IP or other overlay techniques).

[0047] It should be noted that if each of the devices of the cluster in the network performs the backend server health check, it may lead to a large number of health check packets sent to an individual device. In view of this, the backend server health check may be performed by a few devices of the cluster and the result may be shared among the rest of the devices in the cluster. The health check may include a service check and a connectivity check. The service check may include determining whether the application or the backend server is still available. As already mentioned above, not every device in the cluster needs to perform this check. The check may be performed by a few devices and the result propagated to the rest of the devices in the cluster. A connectivity check includes determining whether the service node can reach the backend server. The path to the backend server may be specific to the service node, so this may not be distributed across service nodes, and each device in the cluster may perform its own check.

[0048] In an example embodiment, the system 300 may comprise an orchestrator 325. The orchestrator 325 may be configured to bring up and bring down the service nodes 320, the traffic classification engines 310, and backend servers. The orchestrator 325 may detect presence of the one or more service nodes 320 and transmit data associated with the presence of the one or more service nodes 320 to the cluster master 305. Furthermore, the orchestrator 325 may inform the cluster master 305 of bringing up or bringing down the service nodes 320. The orchestrator 325 may communicate with the cluster master 305 and the service nodes 320 using one or more Application Programming Interfaces (APIs).

[0049] In an example embodiment, a centralized or distributed network database may be used and shared among all devices in the cluster of the system 300, such as the cluster master, the traffic classification engine, and other service nodes. Each device may connect to the network database and update tables according to its role. Relevant database records may be replicated to the devices that are part of the cluster. The distributed network database may be used to store configurations and states of the devices, e.g. to store data associated with the cluster master, the traffic classification engine, the one or more service nodes, backend servers, and service policy data. The data stored in the distributed network database may include the network data and the service node data. The distributed network database may include tables with information concerning service types, availability of resources, traffic classification, network maps, and so forth. The cluster master 305 may be responsible for maintaining the distributed network database and replicating it to devices. The network database may be replicated to the traffic classification engines 310 and the service nodes 320. In an example embodiment, the network database may internally replicate data across the participant nodes.

[0050] In the embodiments described above, the system 300 may comprise a dedicated cluster master 305, dedicated traffic classification engines 310, and dedicated service nodes 320. In other words, specific devices may be responsible for acting as the cluster master, the traffic classification engine, and the service node. In further example embodiments, the system 300 may include no dedicated devices acting as a cluster master. In this case, the cluster master functionality may be provided by either the traffic classification engines or by the service nodes. Thus, one of the traffic classification engines or one of the service nodes may act as the cluster master. In case the traffic classification engine or service node acting as the cluster master fails, another traffic classification engine or service node may be elected as the cluster master. The traffic classification engines and the service nodes not elected as the cluster master may be configured as backup cluster masters and synchronized with the current cluster master. In an example embodiment, the cluster master may consist of multiple active devices which can act as a single master by sharing duties among the devices.

[0051] In further example embodiments, the system 300 may comprise a dedicated cluster master with no dedicated devices acting as traffic classification engines. In this case, the traffic classification may be performed by one of upstream routers or switches. Also, the service nodes may distribute the traffic among themselves. In an example embodiment, the cluster master and the service nodes may be configured to act as a traffic classification engine.

[0052] In further example embodiments, the system 300 may include no devices acting as cluster masters and traffic classification engines. In this case, one of the service nodes may also act as the cluster master. The traffic classification may be done by upstream routers or switches. The cluster master may program the upstream routers with the traffic mapping. Additionally, the service nodes may distribute the traffic among themselves.

[0053] It should be noted that bringing up new service nodes when the load increases and bringing down the service nodes when the load becomes normal may be performed gracefully, without affecting existing data traffic and connections. When the service node comes up, the distribution of traffic may change from distribution to n service nodes to distribution to **(n+1)** service nodes.

[0054] When a service node is about to be brought down, the traffic coming to this service node may be redirected to other service nodes. For this purpose, a redirection policy associated with the service node about to be brought down may be created by the cluster master and sent to the traffic distribution engine and/or the service nodes. Upon receiving the redirection policy, the traffic distribution engine may direct the traffic to another service node.

[0055] In an example embodiment, the system 300 may comprise, for example, a plurality of traffic distribution engines, each of which may serve traffic to multiple services. Each of the traffic distribution engines may communicate with a different set of service nodes. In case one of the traffic distribution engines fails, another traffic distribution engines may be configured to substitute the failed traffic distribution engine and to distribute the traffic of the failed traffic distribution engines to the corresponding service nodes. Therefore, each of the traffic distribution engines may comprise addresses of all service nodes and not only addresses associated with the service nodes currently in communication with the traffic distribution engine.

[0056] FIG. 4 shows a block diagram 400 for load distribution of an SDN. As shown, diagram 400 includes clients 120, e.g., a computer connected to a network 110. The network 110 may include the SDN. The clients 120 may send one or more service requests for services provided by one or more servers of the virtual machine/server pool 405. These servers may include web servers, wireless application servers, interactive television servers, and so forth. These service requests can be load balanced by a system for load distribution described above. In other words, the service requests of the clients 120 may be intelligently distributed among virtual machine or physical server pool 405 of the SDN.

[0057] The system for load distribution may include a service control plane 410. The service control plane 410 may include one or more data network applets 415, for example, a real time data network applet. The data network applets 415 may check the health and other data associated with the SDN 110 and the virtual machines 405. For example, the data network applets 415 may determine responsiveness of the virtual machines 405. Furthermore, the data network applets 415 may monitor the total connections, central processing unit utilization, memory, network connectivity on the virtual machines 405, and so forth. Therefore, the data network applets 415 may retrieve fine-grained, comprehensive information concerning the SDN and virtual machine service infrastructure.

[0058] The retrieved health data may be transmitted to a service policy engine 420. In example embodiments, a cluster master 305 described above may act as the service policy engine 420. The service policy engine 420 may analyze the health data and, upon the analysis, generate a set of service policies 430 to scale up/down the services, to secure services, to introduce new services, to remove services, to remedy or repair failed devices, and so forth. The system for load distribution may further comprise an orchestrator (not shown) configured to bring up more virtual machines on demand. Therefore, in order to deliver a smooth client experience, the service requests may be load balanced across the virtual machines 405.

[0059] Furthermore, the service policies 430 may be provided to an SDN controller 435. The SDN controller 435, in turn, may steer service requests, i.e. data traffic, across the network devices in the SDN. Effectively, these policies may influence load balancing, high availability as well as programming the SDN network to scale up or scale down services.

[0060] Generally speaking, by unlocking the data associated with the network, service nodes and the server/virtual machines from inside the network, transforming the data into relevant information and the service policies 430, and then presenting the service policies 430 to the SDN controller 435 for configuring the SDN 110, the described infrastructure may enable feedback loops between underlying infrastructure and applications to improve network optimization and application responsiveness.

[0061] The service control plane 410 working in conjunction with the controller 435 and the service policy engine 420 may create a number of deployment possibilities, which may offer an array of basic and advanced load distribution features. In particular, to provide a simple load balancing functionality, the SDN controller 435 and the service control plane 410 may provide some load balancing of their own by leveraging the capabilities of the

SDN 110 or, alternatively, work in conjunction with an ADC 440, also referred to as a service data plane included in the SDN 110 to optionally provide advanced additional functionality.

[0062] In an example embodiment, when the service control plane 410 may be standalone, i.e. without an ADC 440, virtual machines 405, when scaled up, may be programmed with a virtual Internet Protocol (VIP) address on a loopback interface of the virtual machines 405. Thus, for data traffic in need of simple service fulfillment, the service control plane 410 may establish simple policies for distributing service requests and instruct the SDN controller 435 to program network devices to distribute the service requests directly to different virtual machines/physical servers 405. This step may be performed over a physical or logical network.

[0063] In an example embodiment, when the service control plane 410 may work in cooperation with an ADC 440, for more sophisticated ADC functionality typically offered by a purpose built ADC device, the service control plane 410 may manage a set of service policy mapping service requests to one or more ADC devices. The service control plane 410 may instruct the SDN controller 435 to program network devices such that the service requests, i.e. the traffic, may reach one or more ADCs 440. The ADC 440 then may relay the service request to a backend server over a physical or logical network.

[0064] In the described embodiment several traffic flow scenarios may exist. In an example embodiment, only forward traffic may go through the ADC 440. If a simple functionality of the ADC 440, e.g. rate limiting, bandwidth limiting, scripting policies, is required, the forward traffic may traverse the ADC 440. The loopback interface on the servers may be programmed with the VIP address. Response traffic from the virtual machines 405 may bypass the ADC 440.

[0065] In a further example embodiment, forward and reverse traffic may traverse the ADC 440. In the ADC 440 providing a more advanced functionality, e.g. transmission control protocol (TCP) flow optimization, secure sockets layer (SSL) decryption, compression, caching and so forth, is required, the service control panel 410 may need to ensure both the forward and reverse traffic traverses through the ADC 440 by appropriately programming the SDN 110.

[0066] FIG. 5 shows a diagrammatic representation of a machine in the example electronic form of a computer system 500, within which a set of instructions for causing the machine to perform any one or more of the methodologies discussed herein may be executed. In various example embodiments, the machine operates as a standalone device or may be connected (e.g., networked) to other machines. In a networked deployment, the machine may operate in the capacity of a server or a client machine in a server-client network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a PC, a tablet PC, a set-top box (STB), a cellular telephone, a portable music player (e.g., a portable hard drive audio device such as an Moving Picture Experts Group Audio Layer 3 (MP3) player), a web appliance, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[0067] The example computer system 500 includes a processor or multiple processors 502 (e.g., a central processing unit (CPU), a graphics processing unit (GPU), or both), a main memory 504 and a static memory 506, which communicate with each other via a bus 508. The computer system 500 may further include a video display unit 510 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system 500 may also include an alphanumeric input device 512 (e.g., a keyboard), a cursor control device 514 (e.g., a mouse), a disk drive unit 516, a signal generation device 518 (e.g., a speaker), and a network interface device 520.

[0068] The disk drive unit 516 includes a non-transitory computer-readable medium 522, on which is stored one or more sets of instructions and data structures (e.g., instructions 524) embodying or utilized by any one or more of the methodologies or functions described herein. The instructions 524 may also reside, completely or at least partially, within the main memory 504 and/or within the processors 502 during execution thereof by the computer system 500. The main memory 504 and the processors 502 may also constitute machine-readable media.

[0069] The instructions 524 may further be transmitted or received over a network 526 via the network interface device 520 utilizing any one of a number of well-known transfer protocols (e.g., Hyper Text Transfer Protocol (HTTP)).

[0070] While the computer-readable medium 522 is shown in an example embodiment to be a single medium, the term "computer-readable medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database and/or associated caches and servers) that store the one or more sets of instructions. The term "computer-readable medium" shall also be taken to include any medium that is capable of storing, encoding, or carrying a set of instructions for execution by the machine and that causes the machine to perform any one or more of the methodologies of the present application, or that is capable of storing, encoding, or carrying data structures utilized by or associated with such a set of instructions. The term "computer-readable medium" shall accordingly be taken to include, but not be limited to, solid-state memories, optical and magnetic media, and carrier wave signals. Such media may also include, without limitation, hard disks, floppy disks, flash memory cards, digital video disks, random access memory (RAMs), read only memory (ROMs), and the like.

[0071] The example embodiments described herein can be implemented in an operating environment comprising computer-executable instructions (e.g., software) installed on a computer, in hardware, or in a combination of software and hardware. The computer-executable instructions can be written in a computer programming language or can be embodied in firmware logic. If written in a programming language conforming to a recognized standard, such instructions can be executed on a variety of hardware platforms and for interfaces to a variety of operating systems. Although not limited thereto, computer software programs for implementing the present method can be written in any number of suitable programming languages such as, for example, Hypertext Markup Language (HTML), Dynamic HTML, Extensible Markup Language (XML), Extensible Stylesheet Language (XSL), Document Style Semantics and Specification Language (DSSSL), Cascading Style Sheets (CSS), Synchronized Multimedia Integration Language (SMIL), Wireless Markup Language (WML), JavaTM, JiniTM, C, C++, Perl, UNIX Shell, Visual Basic or

Visual Basic Script, Virtual Reality Markup Language (VRML), ColdFusionTM or other compilers, assemblers, interpreters or other computer languages or platforms.

[0072] Thus, methods and systems for load distribution in an SDN are disclosed. Although embodiments have been described with reference to specific example embodiments, it will be evident that various modifications and changes can be made to these example embodiments without departing from the broader spirit and scope of the present application. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method for service load distribution in a data network, the method comprising:

generating a service policy;

providing the service policy to devices associated with the data network; and

distributing one or more service requests according to the service policy.

2. The method of claim 1, further comprising:

retrieving network data associated with the data network;

retrieving service node data associated with one or more service nodes;

analyzing the network data and the service node data;

generating a service policy based on the analysis of the network data and the service node data;

receiving one or more service requests; and

distributing the one or more service requests among the one or more service nodes according to the service policy;

3. The method of claim 2, further comprising pushing the service policy to devices associated with the data network, wherein the device associated with the data network include traffic classification engines and the service nodes.

4. The method of claim 2, wherein the network data includes at least one of health of a service node, a number of total connections, processing unit utilization, a memory status, backend server capacity, and network connectivity.

5. The method of claim 2, wherein the service node data includes at least one of a dynamic state, node processing unit utilization, a node memory status, and responsiveness of the one or more service nodes.

6. The method of claim 2, further comprising developing a further service policy based on the analysis, wherein the further service policy is associated with scaling up, scaling down, remedying, and removing one or more service nodes, traffic classification engines and backend servers, and introducing one or more new service nodes, traffic classification engines and backend servers.

7. The method of claim 1, wherein the devices associated with the data network include the service nodes and traffic classification engines.

8. The method of claim 1, wherein the data network includes a software driven network (SDN), the SDN comprising at least one of traffic classification engines, service nodes, and application delivery controllers.

9. The method of claim 1, wherein the service policy includes at least one of a traffic distribution policy and a service node load policy.

10. The method of claim 2, wherein the one or more service nodes include a virtual machine and a physical device.

11. The method of claim 1, further comprising:

facilitating an application programmable interface to a network administrator; and

developing a further service policy based on the analysis, the further service policy being for the network administrator via the application programmable interface.

12. The method of claim 1, further comprising performing a health check of a backend server by the devices associated with the data network.

13. The method of claim 1, further comprising scaling up service nodes, backend servers, traffic classification engines, cluster masters, and other devices in the SDN network while minimizing disruption to traffic flow.

14. The method of claim 1, further comprising scaling down service nodes, backend servers, traffic classification engines, cluster masters, and other devices in the SDN network while minimizing disruption to traffic flow.

15. The method of claim 1, further comprising scaling up and scaling down services while minimizing disruption to traffic flow.

16. The method of claim 6, wherein when a service node is scaled up or scaled down, the one or more service requests are redirected to the one or more service nodes to continue processing data associated with the service request.

17. The method of claim 2, further comprising optimizing reverse traffic from backend servers to the one or more service nodes.

18. The system of claim 1, further comprising an orchestrator that:

detects the one or more service nodes; and

transmits data associated with the one or more service nodes to the cluster master.

19. The system of claim 1, further comprising a network database that:

stores data associated with at least one of the cluster master, the traffic classification engine, the one or more service nodes, backend servers, and service policies; and

allows the data to be shared among the cluster master, the traffic classification engine, and the one or more service nodes.

20. The system of claim 2, further comprising an orchestrator that:

detects the one or more service nodes; and

transmits data associated with the one or more service nodes to the cluster master.

21. The system of claim 2, further comprising a network database that:

stores data associated with at least one of the cluster master, the traffic classification engine, the one or more service nodes, backend servers, and service policies; and

allows the data to be shared among the cluster master, the traffic classification engine, and the one or more service nodes.

22. A system for service load distribution in a data network, the system comprising:

a cluster master that:

retrieves network data associated with the data network;

retrieves service node data associated with one or more service nodes;

analyzes the network data and the service node data;

generates a service policy based on the analysis; and

provides the service policy to devices associated with the data network;

one or more service nodes that receive the service policy; and

a traffic classification engine that:

receives the service policy;

receives one or more service requests; and

distributes the one or more service requests among the one or more service nodes according to the service policy, the service nodes processing the one or more service requests according to the service policy.

23. The system of claim 22, wherein the devices associated with the data network include the service nodes and traffic classification engines.

24. The system of claim 23, wherein the service nodes and the traffic classification engines act as a cluster master, and wherein the cluster master and the service nodes act as a traffic classification engine.

25. The system of claim 22, further comprising an orchestrator that:

detects the one or more service nodes; and

transmits data associated with the one or more service nodes to the cluster master.

26. The system of claim 22, further comprising a network database that:

stores data associated with at least one of the cluster master, the traffic classification engine, the one or more service nodes, backend servers, and service policies; and

allows the data to be shared among the cluster master, the traffic classification engine, and the one or more service nodes.

27. The system of claim 26, wherein the stored data includes the network data and the service node data.

28. The system of claim 22, wherein the processing of the one or more service requests includes forwarding the one or more service requests to one or more backend servers.

29. The system of claim 22, wherein the one or more service nodes are further configured to send the service node data to the cluster master.

30. The system of claim 22, wherein the network data includes at least one of a number of total connections, processing unit utilization, a memory status, and network connectivity.

31. The system of claim 22, wherein the service node data includes at least one of a health of the service node, a dynamic state, and responsiveness of the one or more service nodes.

32. The system of claim 22, wherein the service policy includes at least one of service address, service node address, a traffic distribution policy, and a service node load policy.

33. The system of claim 22, wherein the cluster master includes a backup unit configured to replace the cluster master in case of a failure of the cluster master.

34. The system of claim 22, wherein the cluster master performs at least one of:

developing a further service policy based on the analysis, wherein the further service policy is associated with scaling down, scaling up, remedying, and removing services associated with the one or more service nodes, and introducing a new service associated with the one or more service nodes; and

facilitating provision of an application programmable interface to a network administrator to enable the network administrator to develop, based on the analysis, a further service policy.

35. The system of claim 22, further comprising performing a health check of a backend server by the devices associated with the data network.

36. The system of claim 22, further comprising scaling up service nodes, backend servers, traffic classification engines, cluster masters, and other devices in the SDN network while minimizing disruption to the traffic flow.

37. The system of claim 22, further comprising scaling down the service nodes, backend servers, traffic classification engines, cluster masters, and other devices in the SDN network while minimizing disruption to traffic flow.

38. The system of claim 22, further comprising scaling up and scaling down services while minimizing disruption to traffic flow.

39. The method of claim 34, wherein when services associated with a service node are scaled up or scaled down, the one or more service requests are redirected to the one or more service nodes to continue processing data associated with the service request.

40. The method of claim 22, further comprising optimizing reverse traffic from backend servers to the one or more service nodes.

41. A non-transitory processor-readable medium having instructions stored thereon, which when executed by one or more processors, cause the one or more processors to perform the following operations:

retrieving network data associated with a data network;

retrieving service node data associated with one or more service nodes;

analyzing the network data and the service node data;

generating a service policy;

receiving one or more service requests;

providing the service policy to devices associated with the data network;

distributing the one or more service requests among the one or more service nodes according to the service policy;

developing a further service policy based on the analysis, wherein the further service policy is associated with scaling up, scaling down, remedying, and removing services associated with the one or more service nodes, and introducing a new service associated with the one or more service nodes;

facilitating providing an application programmable interface to a network administrator;

developing a further service policy based on the analysis by the network administrator via the application programmable interface;

performing a health check of a backend server by the devices associated with the data network;

scaling up and scaling down at least one of service nodes, backend servers, traffic classification engines, and cluster masters while minimizing disruption to traffic flow;

scaling up and scaling down services while minimizing disruption to the traffic flow;

optimizing reverse traffic from backend servers to the service node; and

redirecting the one or more service requests to the one or more service nodes to continue processing data associated with the service request when at least one service node has been scaled up or down.

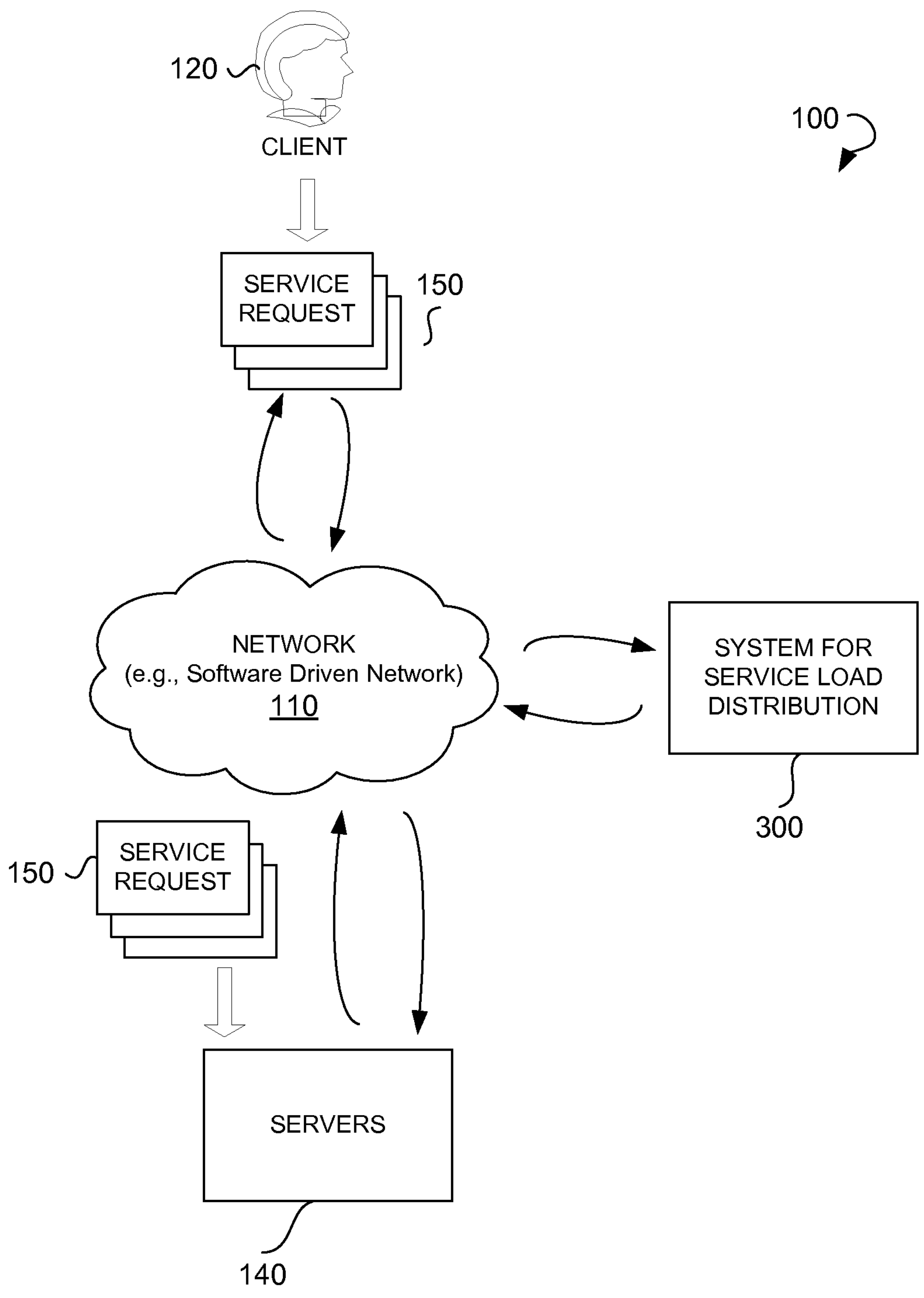
120
CLIENT
100
SERVICE REQUEST
150
NETWORK
(e.g., Software Driven Network)
110
SYSTEM FOR SERVICE LOAD DISTRIBUTION
300
150
SERVICE REQUEST
SERVERS
140

FIG. 1

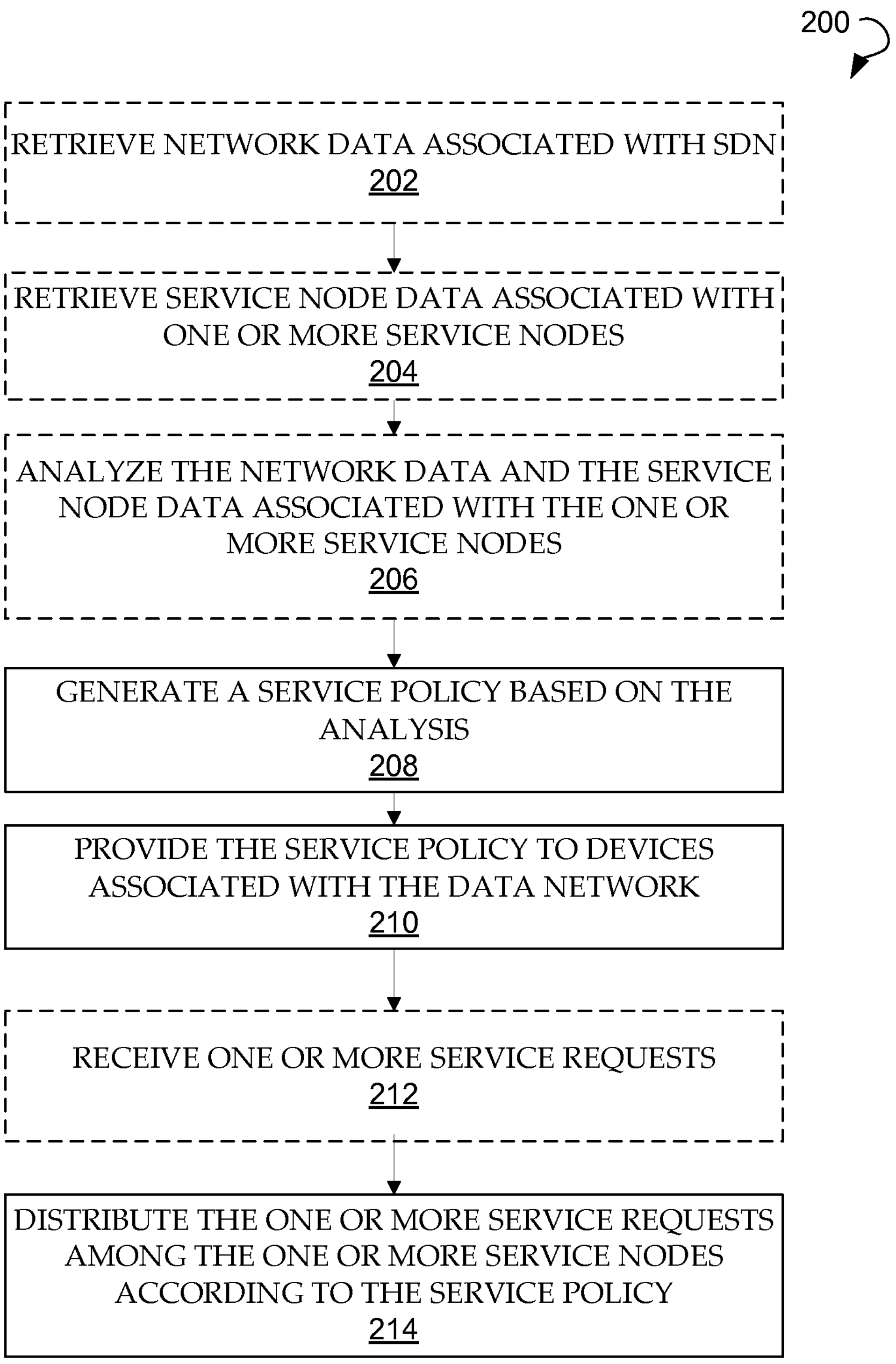
200
RETRIEVE NETWORK DATA ASSOCIATED WITH SDN 202
RETRIEVE SERVICE NODE DATA ASSOCIATED WITH ONE OR MORE SERVICE NODES 204
ANALYZE THE NETWORK DATA AND THE SERVICE NODE DATA ASSOCIATED WITH THE ONE OR MORE SERVICE NODES 206
GENERATE A SERVICE POLICY BASED ON THE ANALYSIS 208
PROVIDE THE SERVICE POLICY TO DEVICES ASSOCIATED WITH THE DATA NETWORK 210
RECEIVE ONE OR MORE SERVICE REQUESTS 212
DISTRIBUTE THE ONE OR MORE SERVICE REQUESTS AMONG THE ONE OR MORE SERVICE NODES ACCORDING TO THE SERVICE POLICY 214

FIG. 2

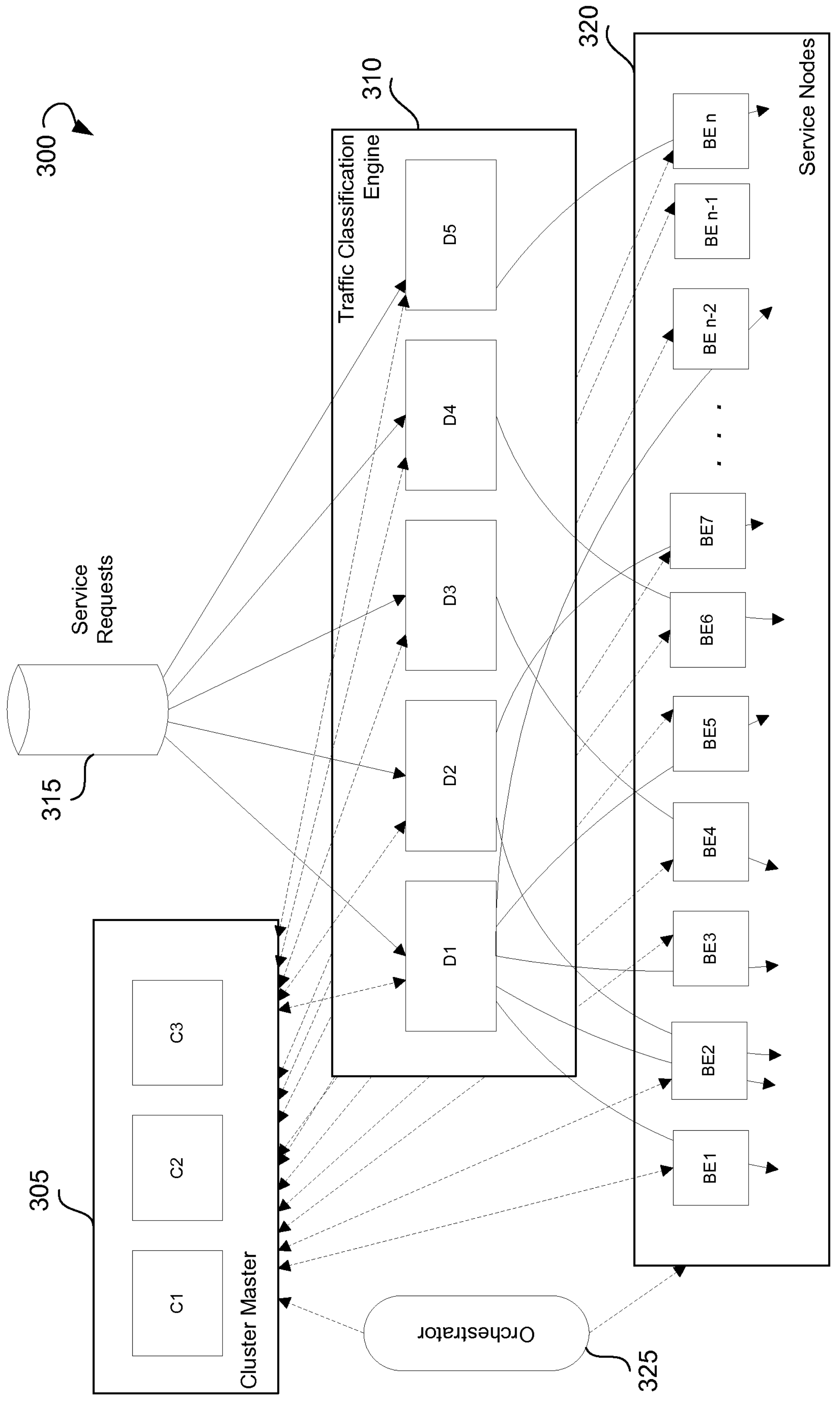
300
Service Requests
315
Traffic Classification Engine
310
D1
D2
D3
D4
D5
Service Nodes
320
BE1
BE2
BE3
BE4
BE5
BE6
BE7
. . . .
BE n-2
BE n-1
BE n
Cluster Master
305
C1
C2
C3
Orchestrator
325

FIG. 3

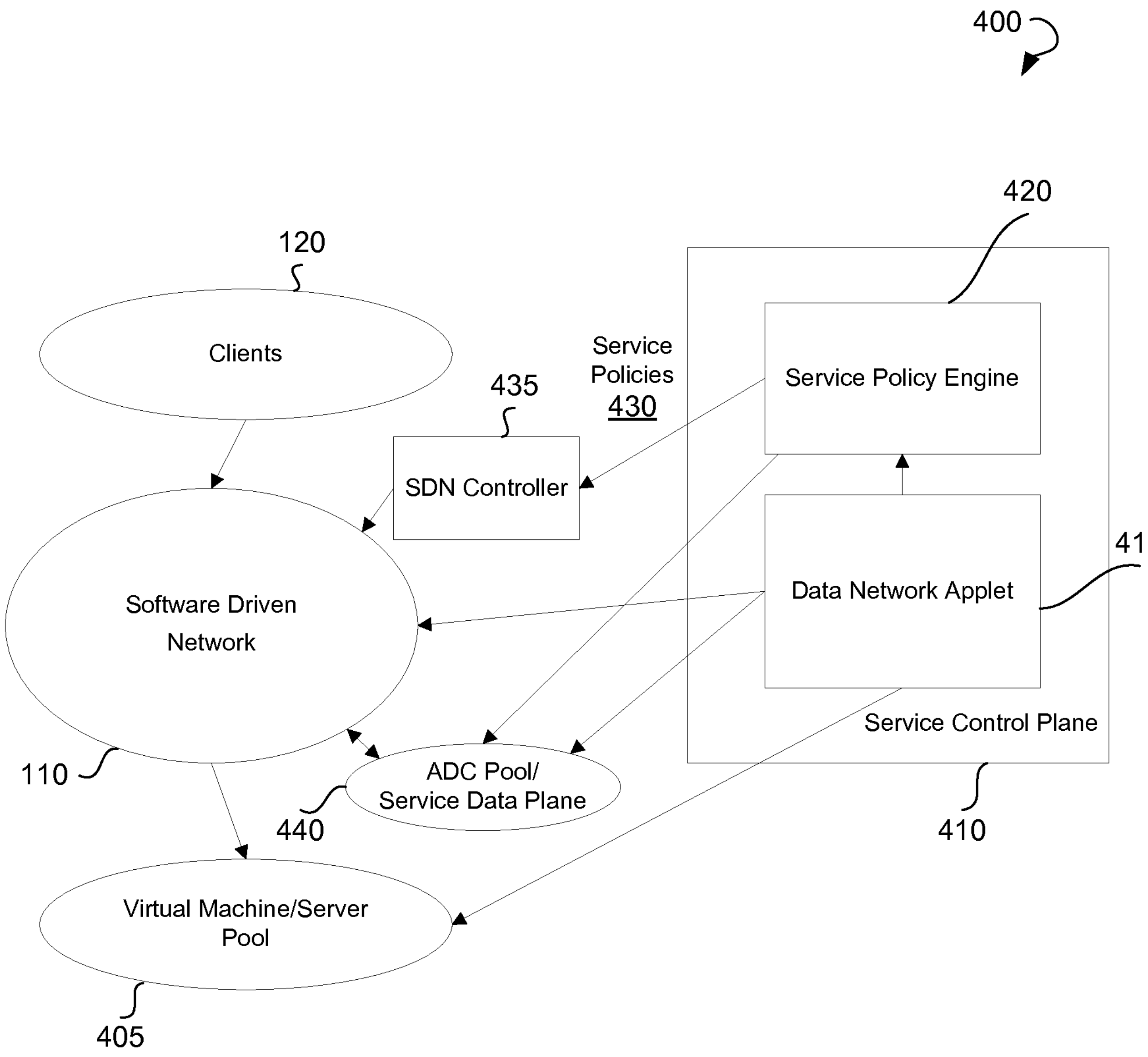
400
420
120
Clients
Service Policies 430
435
Service Policy Engine
SDN Controller
415
Data Network Applet
Software Driven Network
Service Control Plane
110
ADC Pool/ Service Data Plane
440
410
Virtual Machine/Server Pool
405

FIG. 4

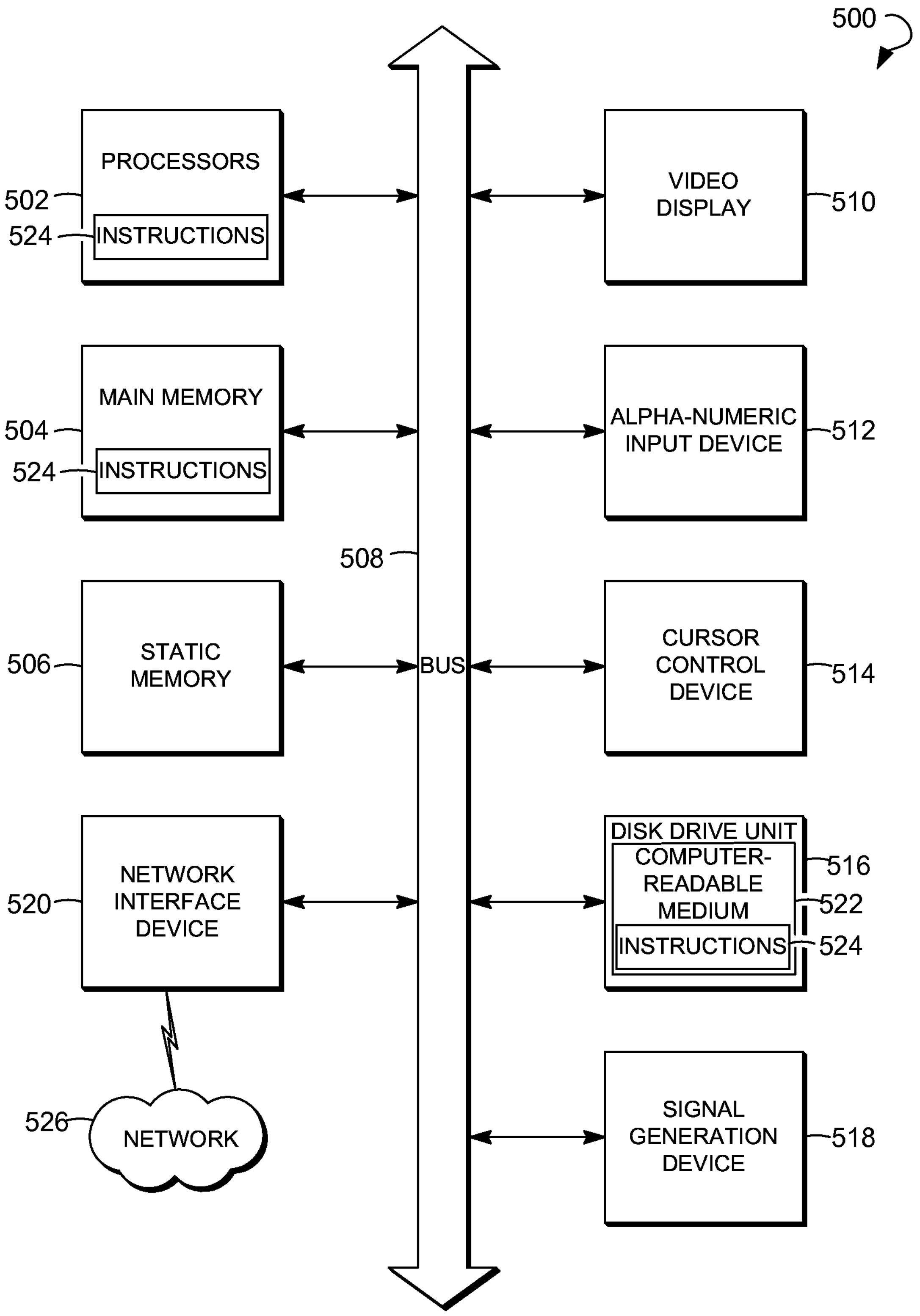
500
PROCESSORS
502
524
INSTRUCTIONS
MAIN MEMORY
504
524
INSTRUCTIONS
508
STATIC MEMORY
506
BUS
NETWORK INTERFACE DEVICE
520
526
NETWORK
VIDEO DISPLAY
510
ALPHA-NUMERIC INPUT DEVICE
512
CURSOR CONTROL DEVICE
514
DISK DRIVE UNIT
COMPUTER-READABLE MEDIUM
INSTRUCTIONS
516
522
524
SIGNAL GENERATION DEVICE
518

FIG. 5