



- (51) International Patent Classification:
H04L 9/40 (2022.01) *H04L 41/044* (2022.01)
- (21) International Application Number:
PCT/US2023/024897
- (22) International Filing Date:
09 June 2023 (09.06.2023)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
17/845,779 21 June 2022 (21.06.2022) US
- (71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US).
- (72) Inventors: **KREGER-STICKLES, Lucas Michael**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US). **TRACY, Leonard Thomas**; 500 Oracle Parkway M/S 50P7, Redwood Shores, California 94065 (US).

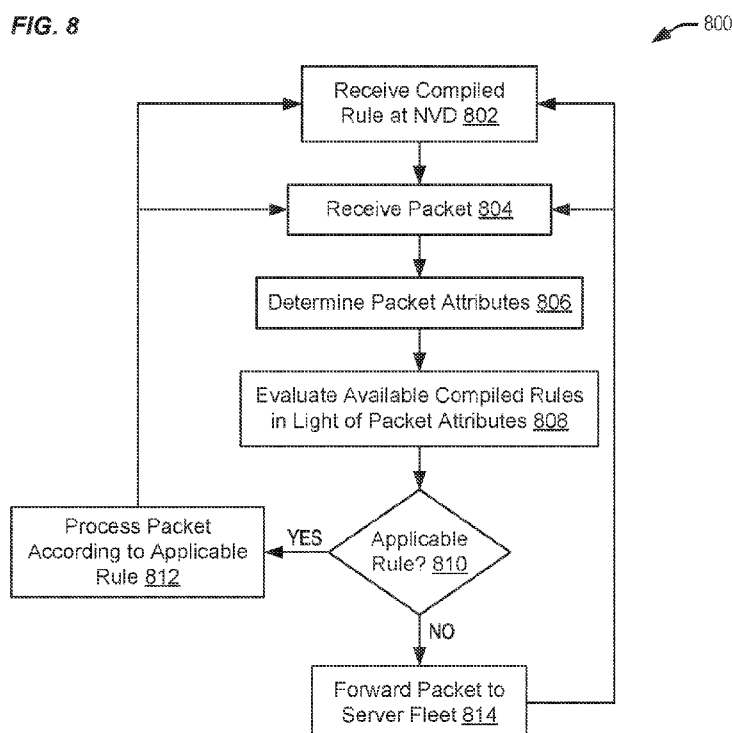
- (74) Agent: **CHRISTIANSEN, Richard B.** et al.; 1100 Peachtree Street NE, Suite 2800, Atlanta, Georgia 30309 (US).

- (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, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

- (84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ,

(54) Title: GEOMETRIC BASED FLOW PROGRAMMING

FIG. 8



(57) Abstract: Systems and methods for geometric based flow programming are disclosed herein. The method can include receiving at least one compiled rule at a first Network Virtualization Device ("NVD"), each of the at least one compiled rules can be applicable to a class of packets received by the first NVD for delivery to a Virtualized Network Interface Card ("VNIC"). The method can include receiving a first packet at the first NVD for delivery to a first VNIC, determining with the first NVD that a first rule of the at least one compiled rule is applicable to the first packet, and processing with the first NVD the first packet according to the first rule.

DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT,
LU, LV, MC, ME, 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).

Published:

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

GEOMETRIC BASED FLOW PROGRAMMING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Patent Application No. 17/845,779, filed June 21, 2022, entitled “GEOMETRIC BASED FLOW PROGRAMMING,” which is hereby
5 incorporated by reference in its entirety.

BACKGROUND

[0002] This application relates to packet processing within a network. Specifically, communications within a Virtual Cloud Network (VCN) pass between Virtual Network Interface Cards (VNIC) via, among other, one or several Network virtualization devices
10 (NVD). Rules can be preloaded on the NVD, one of which rules can be used to process the packet. In cumulative, these rules provided to an NVD can be large and cumbersome. Accordingly, improvements are desired.

BRIEF SUMMARY

15 [0003] One aspect of the present disclosure relates to a method including receiving at least one compiled rule at a first Network Virtualization Device (“NVD”). In some embodiments, each of the at least one compiled rules is applicable to a class of packets received by the first NVD for delivery to a Virtualized Network Interface Card (“VNIC”). The method can include receiving a first packet at the first NVD for delivery to a first VNIC, determining
20 with the first NVD that a first rule of the at least one compiled rule is applicable to the first packet, and processing with the first NVD the first packet according to the first rule.

[0004] In some embodiments, the first packet is processed according to the first rule and according to at least one attribute of the first packet. In some embodiments, the at least one attribute of the packet includes at least one of: a destination address, and a source address. In
25 some embodiments, the source address includes at least one of: a source IP address; and a source port identifier. In some embodiments, the destination address includes at least one of: a destination IP address; and a destination port identifier.

[0005] In some embodiments, each of the at least one compiled rule includes: a flow key; a security policy; and mapping information. In some embodiments, the security policy can
30 include a local security policy, and/or a remote security policy. In some embodiments, each

of the at least one compiled rule further includes packet transformation information, and throttle information.

[0006] In some embodiments, the method includes receiving a second packet at the first NVD for delivery to a second VNIC, determining with the first NVD that the first NVD does not have a rule for processing the second packet, and forwarding the second packet to at least one server. In some embodiments, the at least one server can process the second packet. In some embodiments, the at least one server can include configuration information for a set of VNICs in at least one virtual network.

[0007] In some embodiments, the method includes determining, with the at least one server, a rule for processing the second packet, and processing the second packet with the at least one server according to the determined rule. In some embodiments, the rule is determined based on the configuration information and at least one attribute of the second packet. In some embodiments, the at least one attribute of the packet includes at least one of a destination address, and a source address. In some embodiments, the source address includes at least one of a source IP address, and a source port identifier. In some embodiments, the destination address includes at least one of a destination IP address, and a destination port identifier.

[0008] In some embodiments, the method includes generating a new compiled rule, and providing the new compiled rule to the first NVD. In some embodiments, the new compiled rule includes the second rule. In some embodiments, generating the new compiled rule includes identifying a potential rule for each of a plurality of stages, each of which potential rules is relevant to a plurality of packet flows, and forming the new compiled rule from an intersection between the potential rules for each of the plurality of stages of the flow. In some embodiments, the method includes identifying a plurality of NVDs as similar to the first NVD, and providing the new compiled rule to the plurality of NVDs similar to the first NVD. In some embodiments, the plurality of stages includes two or more of a routing stage, a security stage, a throttle stage, and a transformation stage.

[0009] One aspect of the present disclosure relates to a system. The system includes at least one server including a plurality of compiled rules, and a first Network Virtualization Device (“NVD”). The first NVD can receive at least one compiled rule. In some embodiments, each of the at least one compiled rules is applicable to a class of packets received by the first NVD for delivery to a Virtualized Network Interface Card (“VNIC”). The first NVD can be further

configured to receive a first packet for delivery to a first VNIC, determine that a first rule of the at least one compiled rule is applicable to the first packet, and process the first packet according to the first rule. In some embodiments, the first NVD can further receive a second packet for delivery to a second VNIC, determine that the first NVD does not have a rule for processing the second packet, and forward the second packet to the at least one server. In some embodiments, the at least one server can include configuration information for a set of VNICs in at least one virtual network, and the at least one server can determine a rule for processing the second packet, which rule is determined based on the configuration information and at least one attribute of the second packet, and process the second packet according to the determined rule.

[0010] One aspect of the present disclosure relates to a non-transitory computer-readable storage medium storing a plurality of instructions executable by one or more processors of a first Network Virtualization Device (“NVD”). In some embodiments, the plurality of instructions when executed by the one or more processors of the first NVD cause the one or more processors of the first NVD to receive at least one compiled rule, receive a first packet for delivery to a first VNIC, determine that a first rule of the at least one compiled rule is applicable to the first packet, and process the first packet according to the first rule. In some embodiments, each of the at least one compiled rules is applicable to a class of packets received by the first NVD for delivery to a Virtualized Network Interface Card (“VNIC”).

20

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 is a high level diagram of a distributed environment showing a virtual or overlay cloud network hosted by a cloud service provider infrastructure according to certain embodiments.

25 [0012] FIG. 2 depicts a simplified architectural diagram of the physical components in the physical network within CSPI according to certain embodiments.

[0013] FIG. 3 shows an example arrangement within CSPI where a host machine is connected to multiple network virtualization devices (NVDs) according to certain embodiments.

30 [0014] FIG. 4 depicts connectivity between a host machine and an NVD for providing I/O virtualization for supporting multitenancy according to certain embodiments.

[0015] FIG. 5 depicts a simplified block diagram of a physical network provided by a CSPI according to certain embodiments.

[0016] FIG. 6 is a schematic illustration of one embodiment of an NVD.

[0017] FIG. 7 is a schematic illustration of one embodiment of a system for processing of
5 packets and the generation and distribution of a compiled rule.

[0018] FIG. 8 is a flowchart illustrating one embodiment of process for operation of an NVD.

[0019] FIG. 9 is a flowchart illustrating one embodiment of a process for operation of all or portions of the server fleet.

10 [0020] FIG. 10 is a graphical depiction of one embodiment of application of a modified LPM algorithm.

[0021] FIG. 11 is a depiction of a second embodiment of packet processing.

[0022] FIG. 12 is a block diagram illustrating one pattern for implementing a cloud infrastructure as a service system, according to at least one embodiment.

15 [0023] FIG. 13 is a block diagram illustrating another pattern for implementing a cloud infrastructure as a service system, according to at least one embodiment.

[0024] FIG. 14 is a block diagram illustrating another pattern for implementing a cloud infrastructure as a service system, according to at least one embodiment.

[0025] FIG. 15 is a block diagram illustrating another pattern for implementing a cloud
20 infrastructure as a service system, according to at least one embodiment.

[0026] FIG. 16 is a block diagram illustrating an example computer system, according to at least one embodiment.

DETAILED DESCRIPTION

25 **Example Virtual Networking Architecture**

[0027] The term cloud service is generally used to refer to a service that is made available by a cloud services provider (CSP) to users or customers on demand (e.g., via a subscription model) using systems and infrastructure (cloud infrastructure) provided by the CSP.

Typically, the servers and systems that make up the CSP's infrastructure are separate from the customer's own on-premise servers and systems. Customers can thus avail themselves of cloud services provided by the CSP without having to purchase separate hardware and software resources for the services. Cloud services are designed to provide a subscribing
5 customer easy, scalable access to applications and computing resources without the customer having to invest in procuring the infrastructure that is used for providing the services.

[0028] There are several cloud service providers that offer various types of cloud services. There are various different types or models of cloud services including Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS), and others.

10 [0029] A customer can subscribe to one or more cloud services provided by a CSP. The customer can be any entity such as an individual, an organization, an enterprise, and the like. When a customer subscribes to or registers for a service provided by a CSP, a tenancy or an account is created for that customer. The customer can then, via this account, access the subscribed-to one or more cloud resources associated with the account.

15 [0030] As noted above, infrastructure as a service (IaaS) is one particular type of cloud computing service. In an IaaS model, the CSP provides infrastructure (referred to as cloud services provider infrastructure or CSPI) that can be used by customers to build their own customizable networks and deploy customer resources. The customer's resources and networks are thus hosted in a distributed environment by infrastructure provided by a CSP.
20 This is different from traditional computing, where the customer's resources and networks are hosted by infrastructure provided by the customer.

[0031] The CSPI may comprise interconnected high-performance compute resources including various host machines, memory resources, and network resources that form a physical network, which is also referred to as a substrate network or an underlay network.
25 The resources in CSPI may be spread across one or more data centers that may be geographically spread across one or more geographical regions. Virtualization software may be executed by these physical resources to provide a virtualized distributed environment. The virtualization creates an overlay network (also known as a software-based network, a software-defined network, or a virtual network) over the physical network. The CSPI
30 physical network provides the underlying basis for creating one or more overlay or virtual networks on top of the physical network. The physical network (or substrate network or underlay network) comprises physical network devices such as physical switches, routers,

computers and host machines, and the like. An overlay network is a logical (or virtual) network that runs on top of a physical substrate network. A given physical network can support one or multiple overlay networks. Overlay networks typically use encapsulation techniques to differentiate between traffic belonging to different overlay networks. A virtual
5 or overlay network is also referred to as a virtual cloud network (VCN). The virtual networks are implemented using software virtualization technologies (e.g., hypervisors, virtualization functions implemented by network virtualization devices (NVDs) (e.g., smartNICs), top-of-rack (TOR) switches, smart TORs that implement one or more functions performed by an NVD, and other mechanisms) to create layers of network abstraction that can be run on top of
10 the physical network. Virtual networks can take on many forms, including peer-to-peer networks, IP networks, and others. Virtual networks are typically either Layer-3 IP networks or Layer-2 VLANs. This method of virtual or overlay networking is often referred to as virtual or overlay Layer-3 networking. Examples of protocols developed for virtual networks include IP-in-IP (or Generic Routing Encapsulation (GRE)), Virtual Extensible LAN
15 (VXLAN — IETF RFC 7348), Virtual Private Networks (VPNs) (e.g., MPLS Layer-3 Virtual Private Networks (RFC 4364)), VMware's NSX, GENEVE (Generic Network Virtualization Encapsulation), and others.

[0032] For IaaS, the infrastructure (CSPI) provided by a CSP can be configured to provide virtualized computing resources over a public network (e.g., the Internet). In an IaaS model,
20 a cloud computing services provider can host the infrastructure components (e.g., servers, storage devices, network nodes (e.g., hardware), deployment software, platform virtualization (e.g., a hypervisor layer), or the like). In some cases, an IaaS provider may also supply a variety of services to accompany those infrastructure components (e.g., billing, monitoring, logging, security, load balancing and clustering, etc.). Thus, as these services may be policy-
25 driven, IaaS users may be able to implement policies to drive load balancing to maintain application availability and performance. CSPI provides infrastructure and a set of complementary cloud services that enable customers to build and run a wide range of applications and services in a highly available hosted distributed environment. CSPI offers high-performance compute resources and capabilities and storage capacity in a flexible
30 virtual network that is securely accessible from various networked locations such as from a customer's on-premises network. When a customer subscribes to or registers for an IaaS service provided by a CSP, the tenancy created for that customer is a secure and isolated

partition within the CSPI where the customer can create, organize, and administer their cloud resources.

[0033] Customers can build their own virtual networks using compute, memory, and networking resources provided by CSPI. One or more customer resources or workloads, such as compute instances, can be deployed on these virtual networks. For example, a customer can use resources provided by CSPI to build one or multiple customizable and private virtual network(s) referred to as virtual cloud networks (VCNs). A customer can deploy one or more customer resources, such as compute instances, on a customer VCN. Compute instances can take the form of virtual machines, bare metal instances, and the like. The CSPI thus provides infrastructure and a set of complementary cloud services that enable customers to build and run a wide range of applications and services in a highly available virtual hosted environment. The customer does not manage or control the underlying physical resources provided by CSPI but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., firewalls).

[0034] The CSP may provide a console that enables customers and network administrators to configure, access, and manage resources deployed in the cloud using CSPI resources. In certain embodiments, the console provides a web-based user interface that can be used to access and manage CSPI. In some implementations, the console is a web-based application provided by the CSP.

[0035] CSPI may support single-tenancy or multi-tenancy architectures. In a single tenancy architecture, a software (e.g., an application, a database) or a hardware component (e.g., a host machine or a server) serves a single customer or tenant. In a multi-tenancy architecture, a software or a hardware component serves multiple customers or tenants. Thus, in a multi-tenancy architecture, CSPI resources are shared between multiple customers or tenants. In a multi-tenancy situation, precautions are taken and safeguards put in place within CSPI to ensure that each tenant's data is isolated and remains invisible to other tenants.

[0036] In a physical network, a network endpoint ("endpoint") refers to a computing device or system that is connected to a physical network and communicates back and forth with the network to which it is connected. A network endpoint in the physical network may be connected to a Local Area Network (LAN), a Wide Area Network (WAN), or other type of physical network. Examples of traditional endpoints in a physical network include modems, hubs, bridges, switches, routers, and other networking devices, physical computers (or host

machines), and the like. Each physical device in the physical network has a fixed network address that can be used to communicate with the device. This fixed network address can be a Layer-2 address (e.g., a MAC address), a fixed Layer-3 address (e.g., an IP address), and the like. In a virtualized environment or in a virtual network, the endpoints can include

5 various virtual endpoints such as virtual machines that are hosted by components of the physical network (e.g., hosted by physical host machines). These endpoints in the virtual network are addressed by overlay addresses such as overlay Layer-2 addresses (e.g., overlay MAC addresses) and overlay Layer-3 addresses (e.g., overlay IP addresses). Network overlays enable flexibility by allowing network managers to move around the overlay

10 addresses associated with network endpoints using software management (e.g., via software implementing a control plane for the virtual network). Accordingly, unlike in a physical network, in a virtual network, an overlay address (e.g., an overlay IP address) can be moved from one endpoint to another using network management software. Since the virtual network is built on top of a physical network, communications between components in the virtual

15 network involves both the virtual network and the underlying physical network. In order to facilitate such communications, the components of CSPI are configured to learn and store mappings that map overlay addresses in the virtual network to actual physical addresses in the substrate network, and vice versa. These mappings are then used to facilitate the communications. Customer traffic is encapsulated to facilitate routing in the virtual network.

20 [0037] Accordingly, physical addresses (e.g., physical IP addresses) are associated with components in physical networks and overlay addresses (e.g., overlay IP addresses) are associated with entities in virtual or overlay networks. A physical IP address is an IP address associated with a physical device (e.g., a network device) in the substrate or physical network. For example, each NVD has an associated physical IP address. An overlay IP

25 address is an overlay address associated with an entity in an overlay network, such as with a compute instance in a customer's virtual cloud network (VCN). Two different customers or tenants, each with their own private VCNs can potentially use the same overlay IP address in their VCNs without any knowledge of each other. Both the physical IP addresses and overlay IP addresses are types of real IP addresses. These are separate from virtual IP addresses. A

30 virtual IP address is typically a single IP address that is represents or maps to multiple real IP addresses. A virtual IP address provides a 1-to-many mapping between the virtual IP address and multiple real IP addresses. For example, a load balancer may use a VIP to map to or represent multiple servers, each server having its own real IP address.

[0038] The cloud infrastructure or CSPI is physically hosted in one or more data centers in one or more regions around the world. The CSPI may include components in the physical or substrate network and virtualized components (e.g., virtual networks, compute instances, virtual machines, etc.) that are in an virtual network built on top of the physical network components. In certain embodiments, the CSPI is organized and hosted in realms, regions and availability domains. A region is typically a localized geographic area that contains one or more data centers. Regions are generally independent of each other and can be separated by vast distances, for example, across countries or even continents. For example, a first region may be in Australia, another one in Japan, yet another one in India, and the like. CSPI resources are divided among regions such that each region has its own independent subset of CSPI resources. Each region may provide a set of core infrastructure services and resources, such as, compute resources (e.g., bare metal servers, virtual machine, containers and related infrastructure, etc.); storage resources (e.g., block volume storage, file storage, object storage, archive storage); networking resources (e.g., virtual cloud networks (VCNs), load balancing resources, connections to on-premise networks), database resources; edge networking resources (e.g., DNS); and access management and monitoring resources, and others. Each region generally has multiple paths connecting it to other regions in the realm.

[0039] Generally, an application is deployed in a region (i.e., deployed on infrastructure associated with that region) where it is most heavily used, because using nearby resources is faster than using distant resources. Applications can also be deployed in different regions for various reasons, such as redundancy to mitigate the risk of region-wide events such as large weather systems or earthquakes, to meet varying requirements for legal jurisdictions, tax domains, and other business or social criteria, and the like.

[0040] The data centers within a region can be further organized and subdivided into availability domains (ADs). An availability domain may correspond to one or more data centers located within a region. A region can be composed of one or more availability domains. In such a distributed environment, CSPI resources are either region-specific, such as a virtual cloud network (VCN), or availability domain-specific, such as a compute instance.

[0041] ADs within a region are isolated from each other, fault tolerant, and are configured such that they are very unlikely to fail simultaneously. This is achieved by the ADs not sharing critical infrastructure resources such as networking, physical cables, cable paths,

cable entry points, etc., such that a failure at one AD within a region is unlikely to impact the availability of the other ADs within the same region. The ADs within the same region may be connected to each other by a low latency, high bandwidth network, which makes it possible to provide high-availability connectivity to other networks (e.g., the Internet, customers' on-premise networks, etc.) and to build replicated systems in multiple ADs for both high-availability and disaster recovery. Cloud services use multiple ADs to ensure high availability and to protect against resource failure. As the infrastructure provided by the IaaS provider grows, more regions and ADs may be added with additional capacity. Traffic between availability domains is usually encrypted.

10 **[0042]** In certain embodiments, regions are grouped into realms. A realm is a logical collection of regions. Realms are isolated from each other and do not share any data. Regions in the same realm may communicate with each other, but regions in different realms cannot. A customer's tenancy or account with the CSP exists in a single realm and can be spread across one or more regions that belong to that realm. Typically, when a customer
15 subscribes to an IaaS service, a tenancy or account is created for that customer in the customer-specified region (referred to as the "home" region) within a realm. A customer can extend the customer's tenancy across one or more other regions within the realm. A customer cannot access regions that are not in the realm where the customer's tenancy exists.

[0043] An IaaS provider can provide multiple realms, each realm catered to a particular set
20 of customers or users. For example, a commercial realm may be provided for commercial customers. As another example, a realm may be provided for a specific country for customers within that country. As yet another example, a government realm may be provided for a government, and the like. For example, the government realm may be catered for a specific government and may have a heightened level of security than a commercial realm.
25 For example, Oracle Cloud Infrastructure (OCI) currently offers a realm for commercial regions and two realms (e.g., FedRAMP authorized and IL5 authorized) for government cloud regions.

[0044] In certain embodiments, an AD can be subdivided into one or more fault domains. A fault domain is a grouping of infrastructure resources within an AD to provide anti-affinity.
30 Fault domains allow for the distribution of compute instances such that the instances are not on the same physical hardware within a single AD. This is known as anti-affinity. A fault domain refers to a set of hardware components (computers, switches, and more) that share a

single point of failure. A compute pool is logically divided up into fault domains. Due to this, a hardware failure or compute hardware maintenance event that affects one fault domain does not affect instances in other fault domains. Depending on the embodiment, the number of fault domains for each AD may vary. For instance, in certain embodiments each AD
5 contains three fault domains. A fault domain acts as a logical data center within an AD.

[0045] When a customer subscribes to an IaaS service, resources from CSPI are provisioned for the customer and associated with the customer's tenancy. The customer can use these provisioned resources to build private networks and deploy resources on these networks. The customer networks that are hosted in the cloud by the CSPI are referred to as
10 virtual cloud networks (VCNs). A customer can set up one or more virtual cloud networks (VCNs) using CSPI resources allocated for the customer. A VCN is a virtual or software defined private network. The customer resources that are deployed in the customer's VCN can include compute instances (e.g., virtual machines, bare-metal instances) and other resources. These compute instances may represent various customer workloads such as
15 applications, load balancers, databases, and the like. A compute instance deployed on a VCN can communicate with public accessible endpoints ("public endpoints") over a public network such as the Internet, with other instances in the same VCN or other VCNs (e.g., the customer's other VCNs, or VCNs not belonging to the customer), with the customer's on-premise data centers or networks, and with service endpoints, and other types of endpoints.

20 [0046] The CSP may provide various services using the CSPI. In some instances, customers of CSPI may themselves act like service providers and provide services using CSPI resources. A service provider may expose a service endpoint, which is characterized by identification information (e.g., an IP Address, a DNS name and port). A customer's resource (e.g., a compute instance) can consume a particular service by accessing a service endpoint
25 exposed by the service for that particular service. These service endpoints are generally endpoints that are publicly accessible by users using public IP addresses associated with the endpoints via a public communication network such as the Internet. Network endpoints that are publicly accessible are also sometimes referred to as public endpoints.

[0047] In certain embodiments, a service provider may expose a service via an endpoint
30 (sometimes referred to as a service endpoint) for the service. Customers of the service can then use this service endpoint to access the service. In certain implementations, a service endpoint provided for a service can be accessed by multiple customers that intend to consume

that service. In other implementations, a dedicated service endpoint may be provided for a customer such that only that customer can access the service using that dedicated service endpoint.

[0048] In certain embodiments, when a VCN is created, it is associated with a private
5 overlay Classless Inter-Domain Routing (CIDR) address space, which is a range of private
overlay IP addresses that are assigned to the VCN (e.g., 10.0/16). A VCN includes
associated subnets, route tables, and gateways. A VCN resides within a single region but can
span one or more or all of the region's availability domains. A gateway is a virtual interface
that is configured for a VCN and enables communication of traffic to and from the VCN to
10 one or more endpoints outside the VCN. One or more different types of gateways may be
configured for a VCN to enable communication to and from different types of endpoints.

[0049] A VCN can be subdivided into one or more sub-networks such as one or more
subnets. A subnet is thus a unit of configuration or a subdivision that can be created within a
VCN. A VCN can have one or multiple subnets. Each subnet within a VCN is associated
15 with a contiguous range of overlay IP addresses (e.g., 10.0.0.0/24 and 10.0.1.0/24) that do not
overlap with other subnets in that VCN and which represent an address space subset within
the address space of the VCN.

[0050] Each compute instance is associated with a virtual network interface card (VNIC),
that enables the compute instance to participate in a subnet of a VCN. A VNIC is a logical
20 representation of physical Network Interface Card (NIC). In general, a VNIC is an interface
between an entity (e.g., a compute instance, a service) and a virtual network. A VNIC exists
in a subnet, has one or more associated IP addresses, and associated security rules or policies.
A VNIC is equivalent to a Layer-2 port on a switch. A VNIC is attached to a compute
instance and to a subnet within a VCN. A VNIC associated with a compute instance enables
25 the compute instance to be a part of a subnet of a VCN and enables the compute instance to
communicate (e.g., send and receive packets) with endpoints that are on the same subnet as
the compute instance, with endpoints in different subnets in the VCN, or with endpoints
outside the VCN. The VNIC associated with a compute instance thus determines how the
compute instance connects with endpoints inside and outside the VCN. A VNIC for a
30 compute instance is created and associated with that compute instance when the compute
instance is created and added to a subnet within a VCN. For a subnet comprising a set of

compute instances, the subnet contains the VNICs corresponding to the set of compute instances, each VNIC attached to a compute instance within the set of computer instances.

[0051] Each compute instance is assigned a private overlay IP address via the VNIC associated with the compute instance. This private overlay IP address is assigned to the
5 VNIC that is associated with the compute instance when the compute instance is created and used for routing traffic to and from the compute instance. All VNICs in a given subnet use the same route table, security lists, and DHCP options. As described above, each subnet within a VCN is associated with a contiguous range of overlay IP addresses (e.g., 10.0.0.0/24 and 10.0.1.0/24) that do not overlap with other subnets in that VCN and which represent an
10 address space subset within the address space of the VCN. For a VNIC on a particular subnet of a VCN, the private overlay IP address that is assigned to the VNIC is an address from the contiguous range of overlay IP addresses allocated for the subnet.

[0052] In certain embodiments, a compute instance may optionally be assigned additional overlay IP addresses in addition to the private overlay IP address, such as, for example, one
15 or more public IP addresses if in a public subnet. These multiple addresses are assigned either on the same VNIC or over multiple VNICs that are associated with the compute instance. Each instance however has a primary VNIC that is created during instance launch and is associated with the overlay private IP address assigned to the instance—this primary VNIC cannot be removed. Additional VNICs, referred to as secondary VNICs, can be added
20 to an existing instance in the same availability domain as the primary VNIC. All the VNICs are in the same availability domain as the instance. A secondary VNIC can be in a subnet in the same VCN as the primary VNIC, or in a different subnet that is either in the same VCN or a different one.

[0053] A compute instance may optionally be assigned a public IP address if it is in a
25 public subnet. A subnet can be designated as either a public subnet or a private subnet at the time the subnet is created. A private subnet means that the resources (e.g., compute instances) and associated VNICs in the subnet cannot have public overlay IP addresses. A public subnet means that the resources and associated VNICs in the subnet can have public IP addresses. A customer can designate a subnet to exist either in a single availability domain or
30 across multiple availability domains in a region or realm.

[0054] As described above, a VCN may be subdivided into one or more subnets. In certain embodiments, a Virtual Router (VR) configured for the VCN (referred to as the VCN VR or

just VR) enables communications between the subnets of the VCN. For a subnet within a VCN, the VR represents a logical gateway for that subnet that enables the subnet (i.e., the compute instances on that subnet) to communicate with endpoints on other subnets within the VCN, and with other endpoints outside the VCN. The VCN VR is a logical entity that is

5 configured to route traffic between VNICs in the VCN and virtual gateways ("gateways") associated with the VCN. Gateways are further described below with respect to FIG. 1. A VCN VR is a Layer-3/IP Layer concept. In one embodiment, there is one VCN VR for a VCN where the VCN VR has potentially an unlimited number of ports addressed by IP addresses, with one port for each subnet of the VCN. In this manner, the VCN VR has a

10 different IP address for each subnet in the VCN that the VCN VR is attached to. The VR is also connected to the various gateways configured for a VCN. In certain embodiments, a particular overlay IP address from the overlay IP address range for a subnet is reserved for a port of the VCN VR for that subnet. For example, consider a VCN having two subnets with associated address ranges 10.0/16 and 10.1/16, respectively. For the first subnet within the

15 VCN with address range 10.0/16, an address from this range is reserved for a port of the VCN VR for that subnet. In some instances, the first IP address from the range may be reserved for the VCN VR. For example, for the subnet with overlay IP address range 10.0/16, IP address 10.0.0.1 may be reserved for a port of the VCN VR for that subnet. For the second subnet within the same VCN with address range 10.1/16, the VCN VR may have a

20 port for that second subnet with IP address 10.1.0.1. The VCN VR has a different IP address for each of the subnets in the VCN.

[0055] In some other embodiments, each subnet within a VCN may have its own associated VR that is addressable by the subnet using a reserved or default IP address associated with the VR. The reserved or default IP address may, for example, be the first IP address from the

25 range of IP addresses associated with that subnet. The VNICs in the subnet can communicate (e.g., send and receive packets) with the VR associated with the subnet using this default or reserved IP address. In such an embodiment, the VR is the ingress/egress point for that subnet. The VR associated with a subnet within the VCN can communicate with other VRs associated with other subnets within the VCN. The VRs can also communicate with

30 gateways associated with the VCN. The VR function for a subnet is running on or executed by one or more NVDs executing VNICs functionality for VNICs in the subnet.

[0056] Route tables, security rules, and DHCP options may be configured for a VCN. Route tables are virtual route tables for the VCN and include rules to route traffic from

subnets within the VCN to destinations outside the VCN by way of gateways or specially configured instances. A VCN's route tables can be customized to control how packets are forwarded/routed to and from the VCN. DHCP options refers to configuration information that is automatically provided to the instances when they boot up.

5 **[0057]** Security rules configured for a VCN represent overlay firewall rules for the VCN. The security rules can include ingress and egress rules, and specify the types of traffic (e.g., based upon protocol and port) that is allowed in and out of the instances within the VCN. The customer can choose whether a given rule is stateful or stateless. For instance, the customer can allow incoming SSH traffic from anywhere to a set of instances by setting up a
10 stateful ingress rule with source CIDR 0.0.0.0/0, and destination TCP port 22. Security rules can be implemented using network security groups or security lists. A network security group consists of a set of security rules that apply only to the resources in that group. A security list, on the other hand, includes rules that apply to all the resources in any subnet that uses the security list. A VCN may be provided with a default security list with default
15 security rules. DHCP options configured for a VCN provide configuration information that is automatically provided to the instances in the VCN when the instances boot up.

[0058] In certain embodiments, the configuration information for a VCN is determined and stored by a VCN Control Plane. The configuration information for a VCN may include, for example, information about: the address range associated with the VCN, subnets within the
20 VCN and associated information, one or more VRs associated with the VCN, compute instances in the VCN and associated VNICs, NVDs executing the various virtualization network functions (e.g., VNICs, VRs, gateways) associated with the VCN, state information for the VCN, and other VCN-related information. In certain embodiments, a VCN Distribution Service publishes the configuration information stored by the VCN Control
25 Plane, or portions thereof, to the NVDs. The distributed information may be used to update information (e.g., forwarding tables, routing tables, etc.) stored and used by the NVDs to forward packets to and from the compute instances in the VCN.

[0059] In certain embodiments, the creation of VCNs and subnets are handled by a VCN Control Plane (CP) and the launching of compute instances is handled by a Compute Control
30 Plane. The Compute Control Plane is responsible for allocating the physical resources for the compute instance and then calls the VCN Control Plane to create and attach VNICs to the compute instance. The VCN CP also sends VCN data mappings to the VCN data plane that

is configured to perform packet forwarding and routing functions. In certain embodiments, the VCN CP provides a distribution service that is responsible for providing updates to the VCN data plane. Examples of a VCN Control Plane are also depicted in FIGs. 12, 13, 14, and 15 (see references 1216, 1316, 1416, and 1516) and described below.

5 [0060] A customer may create one or more VCNs using resources hosted by CSPI. A compute instance deployed on a customer VCN may communicate with different endpoints. These endpoints can include endpoints that are hosted by CSPI and endpoints outside CSPI.

[0061] Various different architectures for implementing cloud-based service using CSPI are depicted in FIGs. 1, 2, 3, 4, 5, 12, 13, 14, and 16, and are described below. FIG. 1 is a high
10 level diagram of a distributed environment 100 showing an overlay or customer VCN hosted by CSPI according to certain embodiments. The distributed environment depicted in FIG. 1 includes multiple components in the overlay network. Distributed environment 100 depicted in FIG. 1 is merely an example and is not intended to unduly limit the scope of claimed
15 embodiments. Many variations, alternatives, and modifications are possible. For example, in some implementations, the distributed environment depicted in FIG. 1 may have more or fewer systems or components than those shown in FIG. 1, may combine two or more systems, or may have a different configuration or arrangement of systems.

[0062] As shown in the example depicted in FIG. 1, distributed environment 100 comprises CSPI 101 that provides services and resources that customers can subscribe to and use to
20 build their virtual cloud networks (VCNs). In certain embodiments, CSPI 101 offers IaaS services to subscribing customers. The data centers within CSPI 101 may be organized into one or more regions. One example region "Region US" 102 is shown in FIG. 1. A customer has configured a customer VCN 104 for region 102. The customer may deploy various compute instances on VCN 104, where the compute instances may include virtual machines
25 or bare metal instances. Examples of instances include applications, database, load balancers, and the like.

[0063] In the embodiment depicted in FIG. 1, customer VCN 104 comprises two subnets, namely, "Subnet-1" and "Subnet-2", each subnet with its own CIDR IP address range. In
30 FIG. 1, the overlay IP address range for Subnet-1 is 10.0/16 and the address range for Subnet-2 is 10.1/16. A VCN Virtual Router 105 represents a logical gateway for the VCN that enables communications between subnets of the VCN 104, and with other endpoints outside the VCN. VCN VR 105 is configured to route traffic between VNICs in VCN 104 and

gateways associated with VCN 104. VCN VR 105 provides a port for each subnet of VCN 104. For example, VR 105 may provide a port with IP address 10.0.0.1 for Subnet-1 and a port with IP address 10.1.0.1 for Subnet-2.

[0064] Multiple compute instances may be deployed on each subnet, where the compute instances can be virtual machine instances, and/or bare metal instances. The compute instances in a subnet may be hosted by one or more host machines within CSPI 101. A compute instance participates in a subnet via a VNIC associated with the compute instance. For example, as shown in FIG. 1, a compute instance C1 is part of Subnet-1 via a VNIC associated with the compute instance. Likewise, compute instance C2 is part of Subnet-1 via a VNIC associated with C2. In a similar manner, multiple compute instances, which may be virtual machine instances or bare metal instances, may be part of Subnet-1. Via its associated VNIC, each compute instance is assigned a private overlay IP address and a MAC address. For example, in FIG. 1, compute instance C1 has an overlay IP address of 10.0.0.2 and a MAC address of M1, while compute instance C2 has an private overlay IP address of 10.0.0.3 and a MAC address of M2. Each compute instance in Subnet-1, including compute instances C1 and C2, has a default route to VCN VR 105 using IP address 10.0.0.1, which is the IP address for a port of VCN VR 105 for Subnet-1.

[0065] Subnet-2 can have multiple compute instances deployed on it, including virtual machine instances and/or bare metal instances. For example, as shown in FIG. 1, compute instances D1 and D2 are part of Subnet-2 via VNICs associated with the respective compute instances. In the embodiment depicted in FIG. 1, compute instance D1 has an overlay IP address of 10.1.0.2 and a MAC address of MM1, while compute instance D2 has an private overlay IP address of 10.1.0.3 and a MAC address of MM2. Each compute instance in Subnet-2, including compute instances D1 and D2, has a default route to VCN VR 105 using IP address 10.1.0.1, which is the IP address for a port of VCN VR 105 for Subnet-2.

[0066] VCN A 104 may also include one or more load balancers. For example, a load balancer may be provided for a subnet and may be configured to load balance traffic across multiple compute instances on the subnet. A load balancer may also be provided to load balance traffic across subnets in the VCN.

[0067] A particular compute instance deployed on VCN 104 can communicate with various different endpoints. These endpoints may include endpoints that are hosted by CSPI 200 and endpoints outside CSPI 200. Endpoints that are hosted by CSPI 101 may include: an

endpoint on the same subnet as the particular compute instance (e.g., communications between two compute instances in Subnet-1); an endpoint on a different subnet but within the same VCN (e.g., communication between a compute instance in Subnet-1 and a compute instance in Subnet-2); an endpoint in a different VCN in the same region (e.g.,

5 communications between a compute instance in Subnet-1 and an endpoint in a VCN in the same region 106 or 110, communications between a compute instance in Subnet-1 and an endpoint in service network 110 in the same region); or an endpoint in a VCN in a different region (e.g., communications between a compute instance in Subnet-1 and an endpoint in a VCN in a different region 108). A compute instance in a subnet hosted by CSPI 101 may

10 also communicate with endpoints that are not hosted by CSPI 101 (i.e., are outside CSPI 101). These outside endpoints include endpoints in the customer's on-premise network 116, endpoints within other remote cloud hosted networks 118, public endpoints 114 accessible via a public network such as the Internet, and other endpoints.

[0068] Communications between compute instances on the same subnet are facilitated

15 using VNICs associated with the source compute instance and the destination compute instance. For example, compute instance C1 in Subnet-1 may want to send packets to compute instance C2 in Subnet-1. For a packet originating at a source compute instance and whose destination is another compute instance in the same subnet, the packet is first processed by the VNIC associated with the source compute instance. Processing performed

20 by the VNIC associated with the source compute instance can include determining destination information for the packet from the packet headers, identifying any policies (e.g., security lists) configured for the VNIC associated with the source compute instance, determining a next hop for the packet, performing any packet encapsulation/decapsulation functions as needed, and then forwarding/routing the packet to the next hop with the goal of

25 facilitating communication of the packet to its intended destination. When the destination compute instance is in the same subnet as the source compute instance, the VNIC associated with the source compute instance is configured to identify the VNIC associated with the destination compute instance and forward the packet to that VNIC for processing. The VNIC associated with the destination compute instance is then executed and forwards the packet to

30 the destination compute instance.

[0069] For a packet to be communicated from a compute instance in a subnet to an endpoint in a different subnet in the same VCN, the communication is facilitated by the VNICs associated with the source and destination compute instances and the VCN VR. For

example, if compute instance C1 in Subnet-1 in FIG. 1 wants to send a packet to compute instance D1 in Subnet-2, the packet is first processed by the VNIC associated with compute instance C1. The VNIC associated with compute instance C1 is configured to route the packet to the VCN VR 105 using default route or port 10.0.0.1 of the VCN VR. VCN VR 105 is configured to route the packet to Subnet-2 using port 10.1.0.1. The packet is then received and processed by the VNIC associated with D1 and the VNIC forwards the packet to compute instance D1.

[0070] For a packet to be communicated from a compute instance in VCN 104 to an endpoint that is outside VCN 104, the communication is facilitated by the VNIC associated with the source compute instance, VCN VR 105, and gateways associated with VCN 104. One or more types of gateways may be associated with VCN 104. A gateway is an interface between a VCN and another endpoint, where the another endpoint is outside the VCN. A gateway is a Layer-3/IP layer concept and enables a VCN to communicate with endpoints outside the VCN. A gateway thus facilitates traffic flow between a VCN and other VCNs or networks. Various different types of gateways may be configured for a VCN to facilitate different types of communications with different types of endpoints. Depending upon the gateway, the communications may be over public networks (e.g., the Internet) or over private networks. Various communication protocols may be used for these communications.

[0071] For example, compute instance C1 may want to communicate with an endpoint outside VCN 104. The packet may be first processed by the VNIC associated with source compute instance C1. The VNIC processing determines that the destination for the packet is outside the Subnet-1 of C1. The VNIC associated with C1 may forward the packet to VCN VR 105 for VCN 104. VCN VR 105 then processes the packet and as part of the processing, based upon the destination for the packet, determines a particular gateway associated with VCN 104 as the next hop for the packet. VCN VR 105 may then forward the packet to the particular identified gateway. For example, if the destination is an endpoint within the customer's on-premise network, then the packet may be forwarded by VCN VR 105 to Dynamic Routing Gateway (DRG) gateway 122 configured for VCN 104. The packet may then be forwarded from the gateway to a next hop to facilitate communication of the packet to its final intended destination.

[0072] Various different types of gateways may be configured for a VCN. Examples of gateways that may be configured for a VCN are depicted in FIG. 1 and described below.

Examples of gateways associated with a VCN are also depicted in FIGs. 12, 13, 14, and 15 (for example, gateways referenced by reference numbers 1234, 1236, 1238, 1334, 1336, 1338, 1434, 1436, 1438, 1534, 1536, and 1538) and described below. As shown in the embodiment depicted in FIG. 1, a Dynamic Routing Gateway (DRG) 122 may be added to or
5 be associated with customer VCN 104 and provides a path for private network traffic communication between customer VCN 104 and another endpoint, where the another endpoint can be the customer's on-premise network 116, a VCN 108 in a different region of CSPI 101, or other remote cloud networks 118 not hosted by CSPI 101. Customer on-premise network 116 may be a customer network or a customer data center built using the
10 customer's resources. Access to customer on-premise network 116 is generally very restricted. For a customer that has both a customer on-premise network 116 and one or more VCNs 104 deployed or hosted in the cloud by CSPI 101, the customer may want their on-premise network 116 and their cloud-based VCN 104 to be able to communicate with each other. This enables a customer to build an extended hybrid environment encompassing the
15 customer's VCN 104 hosted by CSPI 101 and their on-premises network 116. DRG 122 enables this communication. To enable such communications, a communication channel 124 is set up where one endpoint of the channel is in customer on-premise network 116 and the other endpoint is in CSPI 101 and connected to customer VCN 104. Communication channel 124 can be over public communication networks such as the Internet or private
20 communication networks. Various different communication protocols may be used such as IPsec VPN technology over a public communication network such as the Internet, Oracle's FastConnect technology that uses a private network instead of a public network, and others. The device or equipment in customer on-premise network 116 that forms one end point for communication channel 124 is referred to as the customer premise equipment (CPE), such as
25 CPE 126 depicted in FIG. 1. On the CSPI 101 side, the endpoint may be a host machine executing DRG 122.

[0073] In certain embodiments, a Remote Peering Connection (RPC) can be added to a DRG, which allows a customer to peer one VCN with another VCN in a different region. Using such an RPC, customer VCN 104 can use DRG 122 to connect with a VCN 108 in
30 another region. DRG 122 may also be used to communicate with other remote cloud networks 118, not hosted by CSPI 101 such as a Microsoft Azure cloud, Amazon AWS cloud, and others.

[0074] As shown in FIG. 1, an Internet Gateway (IGW) 120 may be configured for customer VCN 104 the enables a compute instance on VCN 104 to communicate with public endpoints 114 accessible over a public network such as the Internet. IGW 120 is a gateway that connects a VCN to a public network such as the Internet. IGW 120 enables a public
5 subnet (where the resources in the public subnet have public overlay IP addresses) within a VCN, such as VCN 104, direct access to public endpoints 112 on a public network 114 such as the Internet. Using IGW 120, connections can be initiated from a subnet within VCN 104 or from the Internet.

[0075] A Network Address Translation (NAT) gateway 128 can be configured for
10 customer's VCN 104 and enables cloud resources in the customer's VCN, which do not have dedicated public overlay IP addresses, access to the Internet and it does so without exposing those resources to direct incoming Internet connections (e.g., L4-L7 connections). This enables a private subnet within a VCN, such as private Subnet-1 in VCN 104, with private access to public endpoints on the Internet. In NAT gateways, connections can be initiated
15 only from the private subnet to the public Internet and not from the Internet to the private subnet.

[0076] In certain embodiments, a Service Gateway (SGW) 126 can be configured for customer VCN 104 and provides a path for private network traffic between VCN 104 and supported services endpoints in a service network 110. In certain embodiments, service
20 network 110 may be provided by the CSP and may provide various services. An example of such a service network is Oracle's Services Network, which provides various services that can be used by customers. For example, a compute instance (e.g., a database system) in a private subnet of customer VCN 104 can back up data to a service endpoint (e.g., Object Storage) without needing public IP addresses or access to the Internet. In certain
25 embodiments, a VCN can have only one SGW, and connections can only be initiated from a subnet within the VCN and not from service network 110. If a VCN is peered with another, resources in the other VCN typically cannot access the SGW. Resources in on-premises networks that are connected to a VCN with FastConnect or VPN Connect can also use the service gateway configured for that VCN.

30 [0077] In certain implementations, SGW 126 uses the concept of a service Classless Inter-Domain Routing (CIDR) label, which is a string that represents all the regional public IP address ranges for the service or group of services of interest. The customer uses the service

CIDR label when they configure the SGW and related route rules to control traffic to the service. The customer can optionally utilize it when configuring security rules without needing to adjust them if the service's public IP addresses change in the future.

[0078] A Local Peering Gateway (LPG) 132 is a gateway that can be added to customer VCN 104 and enables VCN 104 to peer with another VCN in the same region. Peering means that the VCNs communicate using private IP addresses, without the traffic traversing a public network such as the Internet or without routing the traffic through the customer's on-premises network 116. In preferred embodiments, a VCN has a separate LPG for each peering it establishes. Local Peering or VCN Peering is a common practice used to establish network connectivity between different applications or infrastructure management functions.

[0079] Service providers, such as providers of services in service network 110, may provide access to services using different access models. According to a public access model, services may be exposed as public endpoints that are publicly accessible by compute instance in a customer VCN via a public network such as the Internet and or may be privately accessible via SGW 126. According to a specific private access model, services are made accessible as private IP endpoints in a private subnet in the customer's VCN. This is referred to as a Private Endpoint (PE) access and enables a service provider to expose their service as an instance in the customer's private network. A Private Endpoint resource represents a service within the customer's VCN. Each PE manifests as a VNIC (referred to as a PE-VNIC, with one or more private IPs) in a subnet chosen by the customer in the customer's VCN. A PE thus provides a way to present a service within a private customer VCN subnet using a VNIC. Since the endpoint is exposed as a VNIC, all the features associates with a VNIC such as routing rules, security lists, etc., are now available for the PE VNIC.

[0080] A service provider can register their service to enable access through a PE. The provider can associate policies with the service that restricts the service's visibility to the customer tenancies. A provider can register multiple services under a single virtual IP address (VIP), especially for multi-tenant services. There may be multiple such private endpoints (in multiple VCNs) that represent the same service.

[0081] Compute instances in the private subnet can then use the PE VNIC's private IP address or the service DNS name to access the service. Compute instances in the customer VCN can access the service by sending traffic to the private IP address of the PE in the customer VCN. A Private Access Gateway (PAGW) 130 is a gateway resource that can be

attached to a service provider VCN (e.g., a VCN in service network 110) that acts as an ingress/egress point for all traffic from/to customer subnet private endpoints. PAGW 130 enables a provider to scale the number of PE connections without utilizing its internal IP address resources. A provider needs only configure one PAGW for any number of services registered in a single VCN. Providers can represent a service as a private endpoint in multiple VCNs of one or more customers. From the customer's perspective, the PE VNIC, which, instead of being attached to a customer's instance, appears attached to the service with which the customer wishes to interact. The traffic destined to the private endpoint is routed via PAGW 130 to the service. These are referred to as customer-to-service private connections (C2S connections).

[0082] The PE concept can also be used to extend the private access for the service to customer's on-premises networks and data centers, by allowing the traffic to flow through FastConnect/IPsec links and the private endpoint in the customer VCN. Private access for the service can also be extended to the customer's peered VCNs, by allowing the traffic to flow between LPG 132 and the PE in the customer's VCN.

[0083] A customer can control routing in a VCN at the subnet level, so the customer can specify which subnets in the customer's VCN, such as VCN 104, use each gateway. A VCN's route tables are used to decide if traffic is allowed out of a VCN through a particular gateway. For example, in a particular instance, a route table for a public subnet within customer VCN 104 may send non-local traffic through IGW 120. The route table for a private subnet within the same customer VCN 104 may send traffic destined for CSP services through SGW 126. All remaining traffic may be sent via the NAT gateway 128. Route tables only control traffic going out of a VCN.

[0084] Security lists associated with a VCN are used to control traffic that comes into a VCN via a gateway via inbound connections. All resources in a subnet use the same route table and security lists. Security lists may be used to control specific types of traffic allowed in and out of instances in a subnet of a VCN. Security list rules may comprise ingress (inbound) and egress (outbound) rules. For example, an ingress rule may specify an allowed source address range, while an egress rule may specify an allowed destination address range. Security rules may specify a particular protocol (e.g., TCP, ICMP), a particular port (e.g., 22 for SSH, 3389 for Windows RDP), etc. In certain implementations, an instance's operating system may enforce its own firewall rules that are aligned with the security list rules. Rules

may be stateful (e.g., a connection is tracked and the response is automatically allowed without an explicit security list rule for the response traffic) or stateless.

[0085] Access from a customer VCN (i.e., by a resource or compute instance deployed on VCN 104) can be categorized as public access, private access, or dedicated access. Public
5 access refers to an access model where a public IP address or a NAT is used to access a public endpoint. Private access enables customer workloads in VCN 104 with private IP addresses (e.g., resources in a private subnet) to access services without traversing a public network such as the Internet. In certain embodiments, CSPI 101 enables customer VCN
10 workloads with private IP addresses to access the (public service endpoints of) services using a service gateway. A service gateway thus offers a private access model by establishing a virtual link between the customer's VCN and the service's public endpoint residing outside the customer's private network.

[0086] Additionally, CSPI may offer dedicated public access using technologies such as FastConnect public peering where customer on-premises instances can access one or more
15 services in a customer VCN using a FastConnect connection and without traversing a public network such as the Internet. CSPI also may also offer dedicated private access using FastConnect private peering where customer on-premises instances with private IP addresses can access the customer's VCN workloads using a FastConnect connection. FastConnect is a network connectivity alternative to using the public Internet to connect a customer's on-
20 premise network to CSPI and its services. FastConnect provides an easy, elastic, and economical way to create a dedicated and private connection with higher bandwidth options and a more reliable and consistent networking experience when compared to Internet-based connections.

[0087] FIG. 1 and the accompanying description above describes various virtualized
25 components in an example virtual network. As described above, the virtual network is built on the underlying physical or substrate network. FIG. 2 depicts a simplified architectural diagram of the physical components in the physical network within CSPI 200 that provide the underlay for the virtual network according to certain embodiments. As shown, CSPI 200 provides a distributed environment comprising components and resources (e.g., compute,
30 memory, and networking resources) provided by a cloud service provider (CSP). These components and resources are used to provide cloud services (e.g., IaaS services) to subscribing customers, i.e., customers that have subscribed to one or more services provided

by the CSP. Based upon the services subscribed to by a customer, a subset of resources (e.g., compute, memory, and networking resources) of CSPI 200 are provisioned for the customer. Customers can then build their own cloud-based (i.e., CSPI-hosted) customizable and private virtual networks using physical compute, memory, and networking resources provided by
5 CSPI 200. As previously indicated, these customer networks are referred to as virtual cloud networks (VCNs). A customer can deploy one or more customer resources, such as compute instances, on these customer VCNs. Compute instances can be in the form of virtual machines, bare metal instances, and the like. CSPI 200 provides infrastructure and a set of complementary cloud services that enable customers to build and run a wide range of
10 applications and services in a highly available hosted environment.

[0088] In the example embodiment depicted in FIG. 2, the physical components of CSPI 200 include one or more physical host machines or physical servers (e.g., 202, 206, 208), network virtualization devices (NVDs) (e.g., 210, 212), top-of-rack (TOR) switches (e.g., 214, 216), and a physical network (e.g., 218), and switches in physical network 218. The
15 physical host machines or servers may host and execute various compute instances that participate in one or more subnets of a VCN. The compute instances may include virtual machine instances, and bare metal instances. For example, the various compute instances depicted in FIG. 1 may be hosted by the physical host machines depicted in FIG. 2. The virtual machine compute instances in a VCN may be executed by one host machine or by
20 multiple different host machines. The physical host machines may also host virtual host machines, container-based hosts or functions, and the like. The VNICs and VCN VR depicted in FIG. 1 may be executed by the NVDs depicted in FIG. 2. The gateways depicted in FIG. 1 may be executed by the host machines and/or by the NVDs depicted in FIG. 2.

[0089] The host machines or servers may execute a hypervisor (also referred to as a virtual machine monitor or VMM) that creates and enables a virtualized environment on the host
25 machines. The virtualization or virtualized environment facilitates cloud-based computing. One or more compute instances may be created, executed, and managed on a host machine by a hypervisor on that host machine. The hypervisor on a host machine enables the physical computing resources of the host machine (e.g., compute, memory, and networking resources)
30 to be shared between the various compute instances executed by the host machine.

[0090] For example, as depicted in FIG. 2, host machines 202 and 208 execute hypervisors 260 and 266, respectively. These hypervisors may be implemented using software, firmware,

or hardware, or combinations thereof. Typically, a hypervisor is a process or a software layer that sits on top of the host machine's operating system (OS), which in turn executes on the hardware processors of the host machine. The hypervisor provides a virtualized environment by enabling the physical computing resources (e.g., processing resources such as
5 processors/cores, memory resources, networking resources) of the host machine to be shared among the various virtual machine compute instances executed by the host machine. For example, in FIG. 2, hypervisor 260 may sit on top of the OS of host machine 202 and enables the computing resources (e.g., processing, memory, and networking resources) of host machine 202 to be shared between compute instances (e.g., virtual machines) executed by
10 host machine 202. A virtual machine can have its own operating system (referred to as a guest operating system), which may be the same as or different from the OS of the host machine. The operating system of a virtual machine executed by a host machine may be the same as or different from the operating system of another virtual machine executed by the same host machine. A hypervisor thus enables multiple operating systems to be executed
15 alongside each other while sharing the same computing resources of the host machine. The host machines depicted in FIG. 2 may have the same or different types of hypervisors.

[0091] A compute instance can be a virtual machine instance or a bare metal instance. In FIG. 2, compute instances 268 on host machine 202 and 274 on host machine 208 are examples of virtual machine instances. Host machine 206 is an example of a bare metal
20 instance that is provided to a customer.

[0092] In certain instances, an entire host machine may be provisioned to a single customer, and all of the one or more compute instances (either virtual machines or bare metal instance) hosted by that host machine belong to that same customer. In other instances, a host machine may be shared between multiple customers (i.e., multiple tenants). In such a
25 multi-tenancy scenario, a host machine may host virtual machine compute instances belonging to different customers. These compute instances may be members of different VCNs of different customers. In certain embodiments, a bare metal compute instance is hosted by a bare metal server without a hypervisor. When a bare metal compute instance is provisioned, a single customer or tenant maintains control of the physical CPU, memory, and
30 network interfaces of the host machine hosting the bare metal instance and the host machine is not shared with other customers or tenants.

[0093] As previously described, each compute instance that is part of a VCN is associated with a VNIC that enables the compute instance to become a member of a subnet of the VCN. The VNIC associated with a compute instance facilitates the communication of packets or frames to and from the compute instance. A VNIC is associated with a compute instance
5 when the compute instance is created. In certain embodiments, for a compute instance executed by a host machine, the VNIC associated with that compute instance is executed by an NVD connected to the host machine. For example, in FIG. 2, host machine 202 executes a virtual machine compute instance 268 that is associated with VNIC 276, and VNIC 276 is executed by NVD 210 connected to host machine 202. As another example, bare metal
10 instance 272 hosted by host machine 206 is associated with VNIC 280 that is executed by NVD 212 connected to host machine 206. As yet another example, VNIC 284 is associated with compute instance 274 executed by host machine 208, and VNIC 284 is executed by NVD 212 connected to host machine 208.

[0094] For compute instances hosted by a host machine, an NVD connected to that host
15 machine also executes VCN VRs corresponding to VCNs of which the compute instances are members. For example, in the embodiment depicted in FIG. 2, NVD 210 executes VCN VR 277 corresponding to the VCN of which compute instance 268 is a member. NVD 212 may also execute one or more VCN VRs 283 corresponding to VCNs corresponding to the compute instances hosted by host machines 206 and 208.

[0095] A host machine may include one or more network interface cards (NIC) that enable
20 the host machine to be connected to other devices. A NIC on a host machine may provide one or more ports (or interfaces) that enable the host machine to be communicatively connected to another device. For example, a host machine may be connected to an NVD using one or more ports (or interfaces) provided on the host machine and on the NVD. A
25 host machine may also be connected to other devices such as another host machine.

[0096] For example, in FIG. 2, host machine 202 is connected to NVD 210 using link 220 that extends between a port 234 provided by a NIC 232 of host machine 202 and between a port 236 of NVD 210. Host machine 206 is connected to NVD 212 using link 224 that extends between a port 246 provided by a NIC 244 of host machine 206 and between a port
30 248 of NVD 212. Host machine 208 is connected to NVD 212 using link 226 that extends between a port 252 provided by a NIC 250 of host machine 208 and between a port 254 of NVD 212.

[0097] The NVDs are in turn connected via communication links to top-of-the-rack (TOR) switches, which are connected to physical network 218 (also referred to as the switch fabric). In certain embodiments, the links between a host machine and an NVD, and between an NVD and a TOR switch are Ethernet links. For example, in FIG. 2, NVDs 210 and 212 are
5 connected to TOR switches 214 and 216, respectively, using links 228 and 230. In certain embodiments, the links 220, 224, 226, 228, and 230 are Ethernet links. The collection of host machines and NVDs that are connected to a TOR is sometimes referred to as a rack.

[0098] Physical network 218 provides a communication fabric that enables TOR switches to communicate with each other. Physical network 218 can be a multi-tiered network. In
10 certain implementations, physical network 218 is a multi-tiered Clos network of switches, with TOR switches 214 and 216 representing the leaf level nodes of the multi-tiered and multi-node physical switching network 218. Different Clos network configurations are possible including but not limited to a 2-tier network, a 3-tier network, a 4-tier network, a 5-tier network, and in general a "n"-tiered network. An example of a Clos network is depicted
15 in FIG. 5 and described below.

[0099] Various different connection configurations are possible between host machines and NVDs such as one-to-one configuration, many-to-one configuration, one-to-many configuration, and others. In a one-to-one configuration implementation, each host machine is connected to its own separate NVD. For example, in FIG. 2, host machine 202 is
20 connected to NVD 210 via NIC 232 of host machine 202. In a many-to-one configuration, multiple host machines are connected to one NVD. For example, in FIG. 2, host machines 206 and 208 are connected to the same NVD 212 via NICs 244 and 250, respectively.

[00100] In a one-to-many configuration, one host machine is connected to multiple NVDs. FIG. 3 shows an example within CSPI 300 where a host machine is connected to multiple
25 NVDs. As shown in FIG. 3, host machine 302 comprises a network interface card (NIC) 304 that includes multiple ports 306 and 308. Host machine 300 is connected to a first NVD 310 via port 306 and link 320, and connected to a second NVD 312 via port 308 and link 322. Ports 306 and 308 may be Ethernet ports and the links 320 and 322 between host machine 302 and NVDs 310 and 312 may be Ethernet links. NVD 310 is in turn connected to a first
30 TOR switch 314 and NVD 312 is connected to a second TOR switch 316. The links between NVDs 310 and 312, and TOR switches 314 and 316 may be Ethernet links. TOR switches 314 and 316 represent the Tier-0 switching devices in multi-tiered physical network 318.

[00101] The arrangement depicted in FIG. 3 provides two separate physical network paths to and from physical switch network 318 to host machine 302: a first path traversing TOR switch 314 to NVD 310 to host machine 302, and a second path traversing TOR switch 316 to NVD 312 to host machine 302. The separate paths provide for enhanced availability (referred to as high availability) of host machine 302. If there are problems in one of the paths (e.g., a link in one of the paths goes down) or devices (e.g., a particular NVD is not functioning), then the other path may be used for communications to/from host machine 302.

[00102] In the configuration depicted in FIG. 3, the host machine is connected to two different NVDs using two different ports provided by a NIC of the host machine. In other embodiments, a host machine may include multiple NICs that enable connectivity of the host machine to multiple NVDs.

[00103] Referring back to FIG. 2, an NVD is a physical device or component that performs one or more network and/or storage virtualization functions. An NVD may be any device with one or more processing units (e.g., CPUs, Network Processing Units (NPU)s, FPGAs, packet processing pipelines, etc.), memory including cache, and ports. The various virtualization functions may be performed by software/firmware executed by the one or more processing units of the NVD.

[00104] An NVD may be implemented in various different forms. For example, in certain embodiments, an NVD is implemented as an interface card referred to as a smartNIC or an intelligent NIC with an embedded processor onboard. A smartNIC is a separate device from the NICs on the host machines. In FIG. 2, the NVDs 210 and 212 may be implemented as smartNICs that are connected to host machines 202, and host machines 206 and 208, respectively.

[00105] A smartNIC is however just one example of an NVD implementation. Various other implementations are possible. For example, in some other implementations, an NVD or one or more functions performed by the NVD may be incorporated into or performed by one or more host machines, one or more TOR switches, and other components of CSPI 200. For example, an NVD may be embodied in a host machine where the functions performed by an NVD are performed by the host machine. As another example, an NVD may be part of a TOR switch or a TOR switch may be configured to perform functions performed by an NVD that enables the TOR switch to perform various complex packet transformations that are used for a public cloud. A TOR that performs the functions of an NVD is sometimes referred to as

a smart TOR. In yet other implementations, where virtual machines (VMs) instances, but not bare metal (BM) instances, are offered to customers, functions performed by an NVD may be implemented inside a hypervisor of the host machine. In some other implementations, some of the functions of the NVD may be offloaded to a centralized service running on a fleet of
5 host machines.

[00106] In certain embodiments, such as when implemented as a smartNIC as shown in FIG. 2, an NVD may comprise multiple physical ports that enable it to be connected to one or more host machines and to one or more TOR switches. A port on an NVD can be classified as a host-facing port (also referred to as a "south port") or a network-facing or TOR-facing
10 port (also referred to as a "north port"). A host-facing port of an NVD is a port that is used to connect the NVD to a host machine. Examples of host-facing ports in FIG. 2 include port 236 on NVD 210, and ports 248 and 254 on NVD 212. A network-facing port of an NVD is a port that is used to connect the NVD to a TOR switch. Examples of network-facing ports in FIG. 2 include port 256 on NVD 210, and port 258 on NVD 212. As shown in FIG. 2, NVD
15 210 is connected to TOR switch 214 using link 228 that extends from port 256 of NVD 210 to the TOR switch 214. Likewise, NVD 212 is connected to TOR switch 216 using link 230 that extends from port 258 of NVD 212 to the TOR switch 216.

[00107] An NVD receives packets and frames from a host machine (e.g., packets and frames generated by a compute instance hosted by the host machine) via a host-facing port and, after
20 performing the necessary packet processing, may forward the packets and frames to a TOR switch via a network-facing port of the NVD. An NVD may receive packets and frames from a TOR switch via a network-facing port of the NVD and, after performing the necessary packet processing, may forward the packets and frames to a host machine via a host-facing port of the NVD.

[00108] In certain embodiments, there may be multiple ports and associated links between
25 an NVD and a TOR switch. These ports and links may be aggregated to form a link aggregator group of multiple ports or links (referred to as a LAG). Link aggregation allows multiple physical links between two end-points (e.g., between an NVD and a TOR switch) to be treated as a single logical link. All the physical links in a given LAG may operate in full-
30 duplex mode at the same speed. LAGs help increase the bandwidth and reliability of the connection between two endpoints. If one of the physical links in the LAG goes down, traffic is dynamically and transparently reassigned to one of the other physical links in the LAG.

The aggregated physical links deliver higher bandwidth than each individual link. The multiple ports associated with a LAG are treated as a single logical port. Traffic can be load-balanced across the multiple physical links of a LAG. One or more LAGs may be configured between two endpoints. The two endpoints may be between an NVD and a TOR switch,
5 between a host machine and an NVD, and the like.

[00109] An NVD implements or performs network virtualization functions. These functions are performed by software/firmware executed by the NVD. Examples of network virtualization functions include without limitation: packet encapsulation and de-capsulation functions; functions for creating a VCN network; functions for implementing network
10 policies such as VCN security list (firewall) functionality; functions that facilitate the routing and forwarding of packets to and from compute instances in a VCN; and the like. In certain embodiments, upon receiving a packet, an NVD is configured to execute a packet processing pipeline for processing the packet and determining how the packet is to be forwarded or routed. As part of this packet processing pipeline, the NVD may execute one or more virtual
15 functions associated with the overlay network such as executing VNICs associated with compute instances in the VCN, executing a Virtual Router (VR) associated with the VCN, the encapsulation and decapsulation of packets to facilitate forwarding or routing in the virtual network, execution of certain gateways (e.g., the Local Peering Gateway), the implementation of Security Lists, Network Security Groups, network address translation
20 (NAT) functionality (e.g., the translation of Public IP to Private IP on a host by host basis), throttling functions, and other functions.

[00110] In certain embodiments, the packet processing data path in an NVD may comprise multiple packet pipelines, each composed of a series of packet transformation stages. In certain implementations, upon receiving a packet, the packet is parsed and classified to a
25 single pipeline. The packet is then processed in a linear fashion, one stage after another, until the packet is either dropped or sent out over an interface of the NVD. These stages provide basic functional packet processing building blocks (e.g., validating headers, enforcing throttle, inserting new Layer-2 headers, enforcing L4 firewall, VCN
encapsulation/decapsulation, etc.) so that new pipelines can be constructed by composing
30 existing stages, and new functionality can be added by creating new stages and inserting them into existing pipelines.

[00111] An NVD may perform both control plane and data plane functions corresponding to a control plane and a data plane of a VCN. Examples of a VCN Control Plane are also depicted in FIGs. 12, 13, 14, and 15 (see references 1216, 1316, 1416, and 1516) and described below. Examples of a VCN Data Plane are depicted in FIGs. 12, 13, 14, and 15 (see references 1218, 1318, 1418, and 1518) and described below. The control plane functions include functions used for configuring a network (e.g., setting up routes and route tables, configuring VNICs, etc.) that controls how data is to be forwarded. In certain embodiments, a VCN Control Plane is provided that computes all the overlay-to-substrate mappings centrally and publishes them to the NVDs and to the virtual network edge devices such as various gateways such as the DRG, the SGW, the IGW, etc. Firewall rules may also be published using the same mechanism. In certain embodiments, an NVD only gets the mappings that are relevant for that NVD. The data plane functions include functions for the actual routing/forwarding of a packet based upon configuration set up using control plane. A VCN data plane is implemented by encapsulating the customer's network packets before they traverse the substrate network. The encapsulation/decapsulation functionality is implemented on the NVDs. In certain embodiments, an NVD is configured to intercept all network packets in and out of host machines and perform network virtualization functions.

[00112] As indicated above, an NVD executes various virtualization functions including VNICs and VCN VRs. An NVD may execute VNICs associated with the compute instances hosted by one or more host machines connected to the VNIC. For example, as depicted in FIG. 2, NVD 210 executes the functionality for VNIC 276 that is associated with compute instance 268 hosted by host machine 202 connected to NVD 210. As another example, NVD 212 executes VNIC 280 that is associated with bare metal compute instance 272 hosted by host machine 206, and executes VNIC 284 that is associated with compute instance 274 hosted by host machine 208. A host machine may host compute instances belonging to different VCNs, which belong to different customers, and the NVD connected to the host machine may execute the VNICs (i.e., execute VNICs-relate functionality) corresponding to the compute instances.

[00113] An NVD also executes VCN Virtual Routers corresponding to the VCNs of the compute instances. For example, in the embodiment depicted in FIG. 2, NVD 210 executes VCN VR 277 corresponding to the VCN to which compute instance 268 belongs. NVD 212 executes one or more VCN VRs 283 corresponding to one or more VCNs to which compute instances hosted by host machines 206 and 208 belong. In certain embodiments, the VCN

VR corresponding to that VCN is executed by all the NVDs connected to host machines that host at least one compute instance belonging to that VCN. If a host machine hosts compute instances belonging to different VCNs, an NVD connected to that host machine may execute VCN VRs corresponding to those different VCNs.

5 [00114] In addition to VNICs and VCN VRs, an NVD may execute various software (e.g., daemons) and include one or more hardware components that facilitate the various network virtualization functions performed by the NVD. For purposes of simplicity, these various components are grouped together as "packet processing components" shown in FIG. 2. For example, NVD 210 comprises packet processing components 286 and NVD 212 comprises
10 packet processing components 288. For example, the packet processing components for an NVD may include a packet processor that is configured to interact with the NVD's ports and hardware interfaces to monitor all packets received by and communicated using the NVD and store network information. The network information may, for example, include network flow information identifying different network flows handled by the NVD and per flow
15 information (e.g., per flow statistics). In certain embodiments, network flows information may be stored on a per VNIC basis. The packet processor may perform packet-by-packet manipulations as well as implement stateful NAT and L4 firewall (FW). As another example, the packet processing components may include a replication agent that is configured to replicate information stored by the NVD to one or more different replication
20 target stores. As yet another example, the packet processing components may include a logging agent that is configured to perform logging functions for the NVD. The packet processing components may also include software for monitoring the performance and health of the NVD and, also possibly of monitoring the state and health of other components connected to the NVD.

25 [00115] FIG. 1 shows the components of an example virtual or overlay network including a VCN, subnets within the VCN, compute instances deployed on subnets, VNICs associated with the compute instances, a VR for a VCN, and a set of gateways configured for the VCN. The overlay components depicted in FIG. 1 may be executed or hosted by one or more of the physical components depicted in FIG. 2. For example, the compute instances in a VCN may
30 be executed or hosted by one or more host machines depicted in FIG. 2. For a compute instance hosted by a host machine, the VNIC associated with that compute instance is typically executed by an NVD connected to that host machine (i.e., the VNIC functionality is provided by the NVD connected to that host machine). The VCN VR function for a VCN is

executed by all the NVDs that are connected to host machines hosting or executing the compute instances that are part of that VCN. The gateways associated with a VCN may be executed by one or more different types of NVDs. For example, certain gateways may be executed by smartNICs, while others may be executed by one or more host machines or other
5 implementations of NVDs.

[00116] As described above, a compute instance in a customer VCN may communicate with various different endpoints, where the endpoints can be within the same subnet as the source compute instance, in a different subnet but within the same VCN as the source compute instance, or with an endpoint that is outside the VCN of the source compute instance. These
10 communications are facilitated using VNICs associated with the compute instances, the VCN VRs, and the gateways associated with the VCNs.

[00117] For communications between two compute instances on the same subnet in a VCN, the communication is facilitated using VNICs associated with the source and destination compute instances. The source and destination compute instances may be hosted by the same
15 host machine or by different host machines. A packet originating from a source compute instance may be forwarded from a host machine hosting the source compute instance to an NVD connected to that host machine. On the NVD, the packet is processed using a packet processing pipeline, which can include execution of the VNIC associated with the source compute instance. Since the destination endpoint for the packet is within the same subnet,
20 execution of the VNIC associated with the source compute instance results in the packet being forwarded to an NVD executing the VNIC associated with the destination compute instance, which then processes and forwards the packet to the destination compute instance. The VNICs associated with the source and destination compute instances may be executed on the same NVD (e.g., when both the source and destination compute instances are hosted by
25 the same host machine) or on different NVDs (e.g., when the source and destination compute instances are hosted by different host machines connected to different NVDs). The VNICs may use routing/forwarding tables stored by the NVD to determine the next hop for the packet.

[00118] For a packet to be communicated from a compute instance in a subnet to an
30 endpoint in a different subnet in the same VCN, the packet originating from the source compute instance is communicated from the host machine hosting the source compute instance to the NVD connected to that host machine. On the NVD, the packet is processed

using a packet processing pipeline, which can include execution of one or more VNICs, and the VR associated with the VCN. For example, as part of the packet processing pipeline, the NVD executes or invokes functionality corresponding to the VNIC (also referred to as executes the VNIC) associated with source compute instance. The functionality performed by the VNIC may include looking at the VLAN tag on the packet. Since the packet's destination is outside the subnet, the VCN VR functionality is next invoked and executed by the NVD. The VCN VR then routes the packet to the NVD executing the VNIC associated with the destination compute instance. The VNIC associated with the destination compute instance then processes the packet and forwards the packet to the destination compute instance. The VNICs associated with the source and destination compute instances may be executed on the same NVD (e.g., when both the source and destination compute instances are hosted by the same host machine) or on different NVDs (e.g., when the source and destination compute instances are hosted by different host machines connected to different NVDs).

[00119] If the destination for the packet is outside the VCN of the source compute instance, then the packet originating from the source compute instance is communicated from the host machine hosting the source compute instance to the NVD connected to that host machine. The NVD executes the VNIC associated with the source compute instance. Since the destination end point of the packet is outside the VCN, the packet is then processed by the VCN VR for that VCN. The NVD invokes the VCN VR functionality, which may result in the packet being forwarded to an NVD executing the appropriate gateway associated with the VCN. For example, if the destination is an endpoint within the customer's on-premise network, then the packet may be forwarded by the VCN VR to the NVD executing the DRG gateway configured for the VCN. The VCN VR may be executed on the same NVD as the NVD executing the VNIC associated with the source compute instance or by a different NVD. The gateway may be executed by an NVD, which may be a smartNIC, a host machine, or other NVD implementation. The packet is then processed by the gateway and forwarded to a next hop that facilitates communication of the packet to its intended destination endpoint. For example, in the embodiment depicted in FIG. 2, a packet originating from compute instance 268 may be communicated from host machine 202 to NVD 210 over link 220 (using NIC 232). On NVD 210, VNIC 276 is invoked since it is the VNIC associated with source compute instance 268. VNIC 276 is configured to examine the encapsulated information in the packet, and determine a next hop for forwarding the packet

with the goal of facilitating communication of the packet to its intended destination endpoint, and then forward the packet to the determined next hop.

[00120] A compute instance deployed on a VCN can communicate with various different endpoints. These endpoints may include endpoints that are hosted by CSPI 200 and
5 endpoints outside CSPI 200. Endpoints hosted by CSPI 200 may include instances in the same VCN or other VCNs, which may be the customer's VCNs, or VCNs not belonging to the customer. Communications between endpoints hosted by CSPI 200 may be performed over physical network 218. A compute instance may also communicate with endpoints that are not hosted by CSPI 200, or are outside CSPI 200. Examples of these endpoints include
10 endpoints within a customer's on-premise network or data center, or public endpoints accessible over a public network such as the Internet. Communications with endpoints outside CSPI 200 may be performed over public networks (e.g., the Internet) (not shown in FIG. 2) or private networks (not shown in FIG. 2) using various communication protocols.

[00121] The architecture of CSPI 200 depicted in FIG. 2 is merely an example and is not
15 intended to be limiting. Variations, alternatives, and modifications are possible in alternative embodiments. For example, in some implementations, CSPI 200 may have more or fewer systems or components than those shown in FIG. 2, may combine two or more systems, or may have a different configuration or arrangement of systems. The systems, subsystems, and other components depicted in FIG. 2 may be implemented in software (e.g., code,
20 instructions, program) executed by one or more processing units (e.g., processors, cores) of the respective systems, using hardware, or combinations thereof. The software may be stored on a non-transitory storage medium (e.g., on a memory device).

[00122] FIG. 4 depicts connectivity between a host machine and an NVD for providing I/O virtualization for supporting multitenancy according to certain embodiments. As depicted in
25 FIG. 4, host machine 402 executes a hypervisor 404 that provides a virtualized environment. Host machine 402 executes two virtual machine instances, VM1 406 belonging to customer/tenant #1 and VM2 408 belonging to customer/tenant #2. Host machine 402 comprises a physical NIC 410 that is connected to an NVD 412 via link 414. Each of the compute instances is attached to a VNIC that is executed by NVD 412. In the embodiment in
30 FIG. 4, VM1 406 is attached to VNIC-VM1 420 and VM2 408 is attached to VNIC-VM2 422.

[00123] As shown in FIG. 4, NIC 410 comprises two logical NICs, logical NIC A 416 and logical NIC B 418. Each virtual machine is attached to and configured to work with its own logical NIC. For example, VM1 406 is attached to logical NIC A 416 and VM2 408 is attached to logical NIC B 418. Even though host machine 402 comprises only one physical
5 NIC 410 that is shared by the multiple tenants, due to the logical NICs, each tenant's virtual machine believes they have their own host machine and NIC.

[00124] In certain embodiments, each logical NIC is assigned its own VLAN ID. Thus, a specific VLAN ID is assigned to logical NIC A 416 for Tenant #1 and a separate VLAN ID is assigned to logical NIC B 418 for Tenant #2. When a packet is communicated from VM1
10 406, a tag assigned to Tenant #1 is attached to the packet by the hypervisor and the packet is then communicated from host machine 402 to NVD 412 over link 414. In a similar manner, when a packet is communicated from VM2 408, a tag assigned to Tenant #2 is attached to the packet by the hypervisor and the packet is then communicated from host machine 402 to NVD 412 over link 414. Accordingly, a packet 424 communicated from host machine 402 to
15 NVD 412 has an associated tag 426 that identifies a specific tenant and associated VM. On the NVD, for a packet 424 received from host machine 402, the tag 426 associated with the packet is used to determine whether the packet is to be processed by VNIC-VM1 420 or by VNIC-VM2 422. The packet is then processed by the corresponding VNIC. The configuration depicted in FIG. 4 enables each tenant's compute instance to believe that they
20 own their own host machine and NIC. The setup depicted in FIG. 4 provides for I/O virtualization for supporting multi-tenancy.

[00125] FIG. 5 depicts a simplified block diagram of a physical network 500 according to certain embodiments. The embodiment depicted in FIG. 5 is structured as a Clos network. A Clos network is a particular type of network topology designed to provide connection
25 redundancy while maintaining high bisection bandwidth and maximum resource utilization. A Clos network is a type of non-blocking, multistage or multi-tiered switching network, where the number of stages or tiers can be two, three, four, five, etc. The embodiment depicted in FIG. 5 is a 3-tiered network comprising tiers 1, 2, and 3. The TOR switches 504 represent Tier-0 switches in the Clos network. One or more NVDs are connected to the TOR
30 switches. Tier-0 switches are also referred to as edge devices of the physical network. The Tier-0 switches are connected to Tier-1 switches, which are also referred to as leaf switches. In the embodiment depicted in FIG. 5, a set of "n" Tier-0 TOR switches are connected to a set of "n" Tier-1 switches and together form a pod. Each Tier-0 switch in a pod is interconnected

to all the Tier-1 switches in the pod, but there is no connectivity of switches between pods. In certain implementations, two pods are referred to as a block. Each block is served by or connected to a set of "n" Tier-2 switches (sometimes referred to as spine switches). There can be several blocks in the physical network topology. The Tier-2 switches are in turn
 5 connected to "n" Tier-3 switches (sometimes referred to as super-spine switches). Communication of packets over physical network 500 is typically performed using one or more Layer-3 communication protocols. Typically, all the layers of the physical network, except for the TORs layer are n-ways redundant thus allowing for high availability. Policies may be specified for pods and blocks to control the visibility of switches to each other in the
 10 physical network so as to enable scaling of the physical network.

[00126] A feature of a Clos network is that the maximum hop count to reach from one Tier-0 switch to another Tier-0 switch (or from an NVD connected to a Tier-0 switch to another NVD connected to a Tier-0 switch) is fixed. For example, in a 3-Tiered Clos network at most seven hops are needed for a packet to reach from one NVD to another NVD, where the
 15 source and target NVDs are connected to the leaf tier of the Clos network. Likewise, in a 4-tiered Clos network, at most nine hops are needed for a packet to reach from one NVD to another NVD, where the source and target NVDs are connected to the leaf tier of the Clos network. Thus, a Clos network architecture maintains consistent latency throughout the network, which is important for communication within and between data centers. A Clos
 20 topology scales horizontally and is cost effective. The bandwidth/throughput capacity of the network can be easily increased by adding more switches at the various tiers (e.g., more leaf and spine switches) and by increasing the number of links between the switches at adjacent tiers.

[00127] In certain embodiments, each resource within CSPI is assigned a unique identifier called a Cloud Identifier (CID). This identifier is included as part of the resource's
 25 information and can be used to manage the resource, for example, via a Console or through APIs. An example syntax for a CID is:

ocid1.<RESOURCE TYPE>.<REALM>.[REGION][.FUTURE USE].<UNIQUE ID>

where,

30 ocid1: The literal string indicating the version of the CID;
 resource type: The type of resource (for example, instance, volume, VCN, subnet, user, group, and so on);

realm: The realm the resource is in. Example values are "c1" for the commercial realm, "c2" for the Government Cloud realm, or "c3" for the Federal Government Cloud realm, etc. Each realm may have its own domain name;

region: The region the resource is in. If the region is not applicable to the resource, this part might be blank;

future use: Reserved for future use.

unique ID: The unique portion of the ID. The format may vary depending on the type of resource or service.

Compiled Rule Generation and Provisioning

10 [00128] Communications within a VCN pass between VNICs via NVDs. Specifically, a packet can be received by an NVD, which can then pass, forward, redirect, and/or direct that packet to or towards a destination VNIC. An NVD utilizes information associated with a received packet and one or several rules to determine what to do with that received packet. These rules are typically stored on individual NVDs, such that when an NVD receives a
15 packet, the NVD processes the packet based on rules stored on that NVD. While each of these rules, on its own, is small and does not take up significant memory, cumulatively, these rules can grow quite large. The cumulative size of these rules can, in some embodiments, limit the number of VNICs on a NVDs, and thereby increase the number of NVDs in a VCN. This increase in the number of NVDs in a VCN increases complications and costs associated
20 with increasing the amount of hardware used in a network.

[00129] The present disclosure relates to techniques, systems, and methods for enabling increased VNIC density on a NVD. Specifically, this includes more efficiently utilizing memory on NVDs via limiting the number of rules stored on the NVD. This includes the creation of a compiled rule, which compiled rule is relevant to the processing of a plurality of
25 different packets and/or flows. In some embodiments, the efficiency of rules installed on the NVD can be increased and/or optimized by making the rules installed on that NVD as general as possible, in other words, each of the installed rules is made to cover the maximum number of possible flows. Efficient rule packing, or in other words, maximizing the applicability of a rule, minimizes the number of rules required in a NVD.

30 [00130] This compiled rule can be created by identifying a potential rule for each of a plurality of stages. These stages can include, for example, a routing stage, a security stage, a throttle stage, and a transformation stage. The intersection of the potential rules for each of

the plurality of stages is determined, and a compiled rule is formed from the intersection of the potential rules.

[00131] The routing stage includes information for resolving a next hop destination for a packet received by an NVD. This routing stage can be derived from, for example, customer
5 routing rules, L2 forwarding inside a VLAN, or L3 forwarding to a VNIC inside of a subnet and/or inside of a VCN. The security stage includes information for determining whether the packet is allowed. In some embodiments, the security stage can include, for example, one or several subnet security lists and/or network security groups that can be accessed to determine if the packet is allowed. The throttle stage can include information for setting maximum
10 and/or minimum delivery rates. The transformation stage can include information for performing any transformation and/or rewrite before delivering a packet. This can include information for performing Network Address Translation (NAT), Port Address Translation (PAT), resolving of MAC addresses, or the like.

[00132] Through the use of compiled rules, memory requirements for rules on a SmartNIC
15 can be diminished. Specifically, a typical application includes a single rule for each flow, whereas a compiled rule is applicable to multiple flows, thus minimizing the total number of rules to cover all potential flows for a network. In addition to this, embodiments disclosed herein relate to the providing of a limited number of compiled rules to a SmartNIC. This limited number can result in a SmartNIC not including rules to cover all potential flows
20 within a network. In other words, such a configured SmartNIC will not have rules for all potential flows, but rather may have rules for a subset of all potential flows. In some embodiments, this subset can be the subset of the most frequently used rules, rules that have been used by the SmartNIC, and/or rules that are expected to be used by the SmartNIC.

[00133] When a SmartNIC receives a packet for processing, the SmartNIC determines if it
25 has an applicable rule. If the SmartNIC has an applicable rule, then the SmartNIC processes the packet. If the SmartNIC does not have an applicable rule, then the SmartNIC passes the packet to at least one server, which at least one server can be configured to process the packet. The at least one server, which can comprise one or several servers and/or one or several fleets of servers, can, in some embodiments, include rules for all potential flows in the
30 network.

[00134] The at least one server receives the packet, identifies a rule for processing the packet, and then processes the packet according to the identified rule. The at least one server

then forms a compiled rule including the rule used for processing the packet. The formation of this compiled rule includes the identification of a potential rule for each of a plurality of stages of the rule, identifying an intersection between the potential rules, and forming the compiled rule from the intersection of the potential rules. The compiled rule can then be
5 provided to the SmartNIC from which the packet for processing by the at least one server was received. In some embodiments, the compiled rule can be provided to the SmartNIC from which the packet for processing by the at least one server was received, and to one or several similar SmartNICs.

[00135] FIG. 6 depicts an embodiment of an NVD 210/212. The NVD 210/212 includes
10 packet processing components 286/288. These packet processing components 286/288 can be configured to process packets received by the NVD 210/212, and specifically, a packet processor 602 of the packet processing components 286/288 can be configured to process packets received by the NVD 210/212. The processing of a received packet can include, for example, interacting with the NVD's 210/212 ports and hardware interfaces to monitor all
15 packets received by and communicated using the NVD 210/212. In some embodiments, the processing of a received packet can further include the identification of one or several rules applicable to the received packet, and the application of that one or several rules to the packet. In some embodiments, the processing of a received packet can include sending the packet to a next hop and/or to a destination, such as to a destination VNIC.

20 [00136] The packet processing components 286/288 can be further configured to store network information, and specifically, the packet processing components 286/288 can include a rule memory 604 which can be configured to store network information. The network information can, for example, include network flow information identifying different network flows handled by the NVD and per flow information (e.g., per flow statistics). In certain
25 embodiments, network flows information can be stored on a per VNIC basis. Based on this network information, which can include one or several rules, also referred to herein as configurations, the packet processing components 286/288 can process received packets. In some embodiments, rules stored by the rule memory 604 can include one or several rules relating to routing, security, throttling, and/or packet transformation.

30 [00137] The packet processing components 286/288 can perform packet-by-packet manipulations as well as implement stateful NAT and L4 firewall (FW). As another example, the packet processing components can include a replication agent 606 that is

configured to replicate information stored by the NVD to one or more different replication target stores. As yet another example, the packet processing components 286/288 may include a logging agent 608 that is configured to perform logging functions for the NVD 210/212. The packet processing components 286/288 may also include software, shown as
5 health monitor 610, for monitoring the performance and health of the NVD 210/212 and, also possibly of monitoring the state and health of other components connected to the NVD 210/212.

[00138] FIG. 7 is a schematic illustration of one embodiment of a system 700 for processing of packets and the generation and distribution of a compiled rule. The system 700 includes a
10 server fleet 702 comprising at least one instance, processor, computer, and/or server in communication with a plurality of NVDs 210. In some embodiments, the at least one processor, computer, and/or server in the server fleet 702 can be located in a single location, or can be located at different locations. In some embodiments, the server fleet 702 can be instantiated in physical hardware, and/or can be instantiated in software.

15 [00139] The server fleet can include a VCN distribution service 704, a configuration storage fleet 706, and a forwarding rule pre-fetch 708, also referred to herein as the pre-fetch service 708. Each of the VCN distribution service 704, the configuration storage fleet 706, and the forwarding rule pre-fetch 708 can be instantiated on hardware, or can be instantiated in software. In some embodiments, each of the VCN distribution service 704, the configuration
20 storage fleet 706, and the forwarding rule pre-fetch 708 can be instantiated on distinct hardware, including, for example, one or several computers, processors, and/or servers.

[00140] In some embodiments, the VCN distribution server 704 can be configured to deliver low level configuration elements to NVDs, to the configuration storage fleet 706, and/or to the forwarding rule pre-fetch 708. In some embodiments, this can include, for example, a low
25 level configuration element can comprise a rule applicable to a single flow and/or to a single endpoint in a network, and specifically in a virtual network.

[00141] A rule can comprise, for example, a flow key that can comprise match criteria. This match criteria can be used to match a rule to a packet. For a rule to be identified as matching a packet, match criteria of the rule can match attributes of the packet, and specifically can
30 match the match criteria as the packet is being received from a host port on a targeted endpoint. Flow keys can, in some embodiments, be general, so that flow offload instructions can refer to "flow classes" as opposed to specific 5 tuples. Flow keys can include, for

example, a VNIC Identifier, a protocol, a source and destination CIDR, and/or source and destination port ranges.

[00142] A rule can comprise a local security policy. The local security policy can indicate both ingress and egress security rules. In some embodiments, this can include indicating whether inbound and/or outbound packets are allowed by a stateful or stateless rule. In some
5 embodiments, if stateful, then the local NVD can perform state tracking to allow inbound return packets.

[00143] A rule can comprise an egress mapping override. The egress mapping override can indicate where to send packets, and can, in some embodiments, override local mappings
10 and/or route rules.

[00144] A rule can comprise a remote security policy. The remote security policy can comprise one or several entries, which can be stored and transmitted with any packets that are overridden using the flow offload policy. In some embodiments, if the local security policy are stateful, and thus, if there is stateful security policy, then the sender can include the local
15 flow state. In some embodiments, the remote security policy can indicate a type of security rule to allow traffic, and specifically to allow processing of the packet. In some embodiments, this rule can be, for example, stateful or stateless. In some embodiments, the remote security policy can include a Reverse Path Forwarding (RPF) key. The RPF key can include a signature to authenticate which entity provided override instructions. The RPF key can be
20 able to trace back to an original expected sender that would match the local RPF check.

[00145] A rule can comprise packet transformation information. The packet transformation information can comprise instructions on how to alter the packet. These instructions, can include, for example, instructions for NAT, PAT, or the like.

[00146] A rule can comprise throttle selection information. In some embodiments, the
25 throttle selection information can indicate and/or identify any classified throttle(s) for application to a packet.

[00147] A rule can comprise information identifying a configuration version. In some embodiments, this configuration version can be used to determine that the rule is still valid. In some embodiments, a rule should be checked to determine, based on its version, that the
30 rule is valid.

[00148] In some embodiments, and if a rule is pre-fetch rule, the rule can include information indicating that it is a pre-fetch rule. As used herein, a pre-fetch rule is a rule provided to an NVD before a packet necessitating use of that rule has been received by the NVD. In some embodiments, identification of a rule as a pre-fetch rule can be used to gather
5 information to prioritize rules based on actual traffic demand, to avoid cases where unused rules, and specifically where unused pre-fetch rules fill available memory. In such embodiments, an unused rule, and specifically, an unused pre-fetch rule can be evicted from the configuration cache first.

[00149] The configuration storage fleet 706, also referred to herein as the configuration
10 storage service 706, can hold the configuration for a set of VNICs. In some embodiments, the configuration storage fleet 706 can comprise configuration information that can be one or several rules for a set of VNICs, which, in some embodiments, can be in at least one virtual network. In some embodiments, for example, the configuration storage fleet 706 can hold the configuration, or in other words, the rule for delivery of a packet to some or all of the VNICs
15 in a network, including, for example, an availability domain (AD). The configuration storage fleet 706 processes packets for NVDs that are missing the relevant configuration, or in other words, do not have rules for processing those packets, and then installs those previously missing rules in the NVDs, thereby enabling those NVDs to process similar packets in the future. When these rules are installed on NVDs, the rules can also be streamed to the pre-
20 fetch fleet 708, so that the pre-fetch fleet 708 knows which rules are available to each VNIC.

[00150] The pre-fetch fleet 708 is configured to detect and install likely candidate forwarding rules on NVDs. This includes determining and/or predicting which rules may be applicable to an NVD, or in other words, determining and/or predicting which NVD(s) may need a rule.

25 [00151] Thus, in operation, as an NVD receives a packet, the recipient NVD can determine if it has a rule and/or configuration for processing that packet. If the NVD has a rule and/or configuration, then the NVD can process the packet. If the NVD does not have a rule and/or configuration, then the NVD can provide the packet to the server fleet 702, and specifically to the configuration storage fleet 706. The configuration storage service 706 can identify the
30 rule for processing the packet, and can process the packet. The configuration storage service 706 can further install that rule onto the NVD. In some embodiments, the installation of that rule can include the creation of a compiled rule which can include information for processing

a plurality of flows. The configurations storage service 706 can then notify the pre-fetch service 708 of the rule installed on the NVD. In some embodiments, this can result in different NVDs having different rules, and the rules can be related to traffic passing through the NVDs, thereby limiting wasted space used for storing unused rules.

5 [00152] FIG. 8 is a flowchart illustrating one embodiment of process 800 for operation of an NVD. This process 800 can be performed by some or all of the NVDs. The process 800 begins at block 802, wherein at least one rule is received at the NVD. In some embodiments, this rule can be provided by the server fleet, and specifically can be provided by the VCN distribution server 704 and/or the configurations storage service 706. In some embodiments,
10 this rule can be provided by the configurations storage service 706 when it is determined that the NVD does not have a rule and/or configuration for processing a packet received by that NVD.

[00153] In some embodiments, this rule can comprise a compiled rule. The compiled rule can be applicable to a class of packets received by the NVD. In some embodiments, the
15 compiled rule can be applicable to a class of packets received by the NVD for delivery, either directly, or indirectly, to a VNIC. In some embodiments, this compiled rule can be received from, for example, the configuration storage fleet 706. In some embodiments, each of the at least one rule received by the NVD can include a flow key, a security policy, and mapping information. In some embodiments, this security policy can include a local security policy
20 and/or a remote security policy. In some embodiments, each of the at least one rule can include packet transformation information and/or throttle information.

[00154] At block 804, a packet is received by the NVD. In some embodiments, this packet can be received by the NVD from another NVD and/or from an instance such as, for example, a compute instance, a bare metal instance, or the like. In some embodiments, the
25 packet may be received from another NVD and/or from an instance via intermediate components, such as via, for example, a TOR switch. This packet can, in some embodiments, be for delivery to a VNIC. In some embodiments, such as when that VNIC is directly coupled to the NVD, the packet can be for delivery directly to that VNIC. The packet can, in some embodiments, be for indirect delivery to a VNIC such as when the NVD sends to the packet
30 to another component, such as, for example, another NVD before the packet is delivered to the VNIC.

[00155] The NVD receiving the packet can determine packet attributes as indicated in block 806. The attributes of the packet can include, for example, at least one of: a source address, such as a source IP address and/or a source IP identifier; and a destination address, such as a destination IP address and/or a destination port identifier.

5 [00156] The NVD evaluates available rules in light of the packet attributes as indicated in block 808. In some embodiments, this can include evaluating available compiled rules in light of the packet attributes. This can include, for example, comparing the attributes of the packet determined in block 806 to the available rules, and specifically to the flow key of each of the available rules. At decision step 810, it is determined if there is an available, applicable
10 rule. In some embodiments, an available rule can be determined to be applicable based on the results of the evaluating of block 808. In some embodiments, an available rule can be determined to be applicable when the attributes of the packet match the flow key of one of the available rules.

[00157] If it is determined that at least one of the available rules is applicable, then the
15 process 800 proceeds to block 812, wherein the packet is processed according to the applicable rule. In some embodiments, the packet can be processed according to the applicable rule and/or to at least one attribute of the packet. In some embodiments, this can include resolving routing, security, throttling, and transformation according to the applicable rule. After the packet is processed, the process 800 returns to block 804 when a next packet is
20 received at the NVD, or to block 802 when a next rule is received at the NVD.

[00158] Returning again to decision step 810, if it is determined that there is no applicable rule, then the process 800 proceeds to block 814 and forwards the packet to the server fleet 702 for processing. After the packet has been forwarded to the server fleet 702, the process 800 returns to block 804 when a next packet is received at the NVD, or to block 802 when a
25 next rule is received at the NVD.

[00159] FIG. 9 is a flowchart illustrating one embodiment of a process 900 for operation of all or portions of the server fleet 702 is shown. In some embodiments, this can include packet processing by the server fleet 702, generation of a compiled rule by the server fleet 702, and/or delivery of a compiled rule to an NVD by the server fleet 702. The process 900 begins
30 at block 902, wherein one or several rules for packet processing are received and/or generated. In some embodiments, this can include, for example, the VCN distribution service

704 receiving one or several low level rules, and/or the VCN distribution service 704 providing low level rules to the configuration storage fleet 706.

[00160] At block 904, a packet is received by the server fleet 702, and in some embodiments by the configuration storage fleet 706. The packet can be received by the server fleet 702
5 from an NVD. In some embodiments, the packet can comprise a packet for which the NVD did not have an applicable rule and/or for which the NVD determined that it did not have an applicable rule.

[00161] The attributes of the packet can be determined as indicated in block 906. In some embodiments, these attributes of the packet can be determined by the server fleet 702, and
10 specifically can be determined by the configuration storage fleet 706. The attributes of the packet can include, for example, at least one of: a source address, such as a source IP address and/or a source IP identifier; and a destination address, such as a destination IP address and/or a destination port identifier.

[00162] At block 908, a rule for processing the packet received in block 904 is determined.
15 In some embodiments, this rule can be determined by the configuration storage fleet 706. In some embodiments, the rule can be determined by evaluating available rules in light of the packet attributes. These rule can be either compiled rules or uncompiled rules. In some embodiments, the determining a rule for processing the packet can include determining the rule based on at least one attribute of the packet and information relating to the rule and/or to
20 the configuration.

[00163] In some embodiments, evaluating available rules in light of the packet attributes can include, for example, comparing the attributes of the packet determined in block 806 to the available rules, and specifically to the flow key of each of the available rules. The rule having a flow key that matches the attributes of the packet can be identified and selected for use in
25 processing the packet. In some embodiments, the rule for processing the second packet can be determined by an attribute of the packet that includes at least one of: a destination address that can be, for example, at least one of a destination IP address and/or a destination port identifier; and a source address that can be, for example, at least one of source IP address and/or a source port identifier.

[00164] At block 910, the packet is processed according to the applicable rule determined in
30 block 908. In some embodiments, the packet can be processed according to the applicable

rule and/or to at least one attribute of the packet. In some embodiments, this can include resolving routing, security, throttling, and transformation according to the applicable rule.

[00165] At block 912 a compiled rule including the rule determined in block 910 is generated and/or identified. In some embodiments, generating and/or identifying a compiled rule can include, for example, identifying a potential rule for each of a plurality of stages. In some embodiments, these stages can include, for example, a routing stage, a security stage, a throttle stage, and a transformation stage. In some embodiments, the plurality of stages can include two or more of, for example, a routing stage, a security stage, a throttle stage, and a transformation stage. In some embodiments, each of the potential rules is relevant to a plurality of packet flows. In some embodiments, each of these potential rules can be the most generalized rule for its stage and that includes the rule determined in block 908.

[00166] The generation of the compiled rule, also referred to herein as rule-packing, minimizes table space for rules installed on an NVD. This is achieved by maximizing the coverage of each compiled rule that is installed on an NVD via maximizing the volume covered by a compiled rule in the n-dimensional space of possible packet flows. By maximizing the volume of coverage of a compiled rule, the number of rules needed by a VNIC decreases.

[00167] In some embodiments, identifying and/or generating a compiled rule can include, for example, forming the new compiled rule from an intersection between the potential rules for each of the plurality of stages of the flow. In some embodiments, the identification of the intersection between the potential rules for each of the plurality of stages of the flow can result in identifying the most generalized rule common to all of the plurality of stages. The compiled rules can be stored in the configuration storage fleet 706.

[00168] The rule packing can be performed across multiple stage including, for example, a routing stage, a security stage, a throttle stage, and a transformation stage. The routing stage can resolve the next hop destination for a packet. This can be performed based on, for example, customer route rules, L2 forwarding inside a VLAN, L3 forwarding to a VNIC inside of a subnet and/or a VCN, or the like. In some embodiments, the output of the routing stage can be an identifier of the next hop destination for a packet. In some embodiments, the output key CIDR, or in other words, the potential rule for the routing stage, is the largest CIDR matching the packet such that no more specific route pointing to a different target overlaps with the potential rule.

[00169] In some embodiments, the potential rule for the routing stage can be generated through a "Longest Prefix Match" (LPM) algorithm. In some embodiments, this longest prefix match algorithm can be performed with a Trie tree, which Trie tree can be a binary tree where each node in the tree corresponds to a bit in the IP address of a packet. The root of the tree can correspond to the first bit in the destination address being considered. Such an algorithm would traverse to the child pointer based on the value of the current bit, i.e. if the current bit is 0 move to the left child and if the bit is 1 move to the right child. Nodes with matching route rules are marked with a pointer to the next hop. The LPM can be found by traversing through the Trie tree until a node is reached, which node has no more children that match the packet (i.e. the current bit is 0 and there is no left child). The last matching route rule above the termination point is the LPM.

[00170] In some embodiments, the LPM algorithm can be modified to generate and/or determine the output key for the routing stage. The algorithm can traverse the Trie tree until the LPM is found. After the LPM has been found, the algorithm continues to follow the Trie traversing the node of the LPM until a first terminal node is found. This first terminal node can be the terminal node, or in other words, a leaf node, located closest in the Trie to the node of the LPM. This first terminal node along the lookup address is then identified as the output key. In some embodiments, this algorithm can ignore nodes that have the same next hop as the current node.

[00171] An exemplary embodiment of application of this modified LPM algorithm is shown in FIG. 10. For IP address 10.1.1.1, and with the depicted Trie tree, 10.0.0.0/8 is the LPM, and the output key for 10.1.1.1 is 10.1.1.0/24.

[00172] The security stage determines whether delivery of a packet is allowed. This includes looking up the packet in one or several security lists, which security lists can be subnet security lists and/or network security lists. In some embodiments, the packet is looked up in one or several security lists to determine if the packet is allowed. The output of the security stage can be a determination and/or an indication of a determination of whether the packet is allowed or should be dropped. In some embodiments, the output key of the security stage is the largest block matching the packet such that all other flows covered by the potential rule have the same security disposition.

[00173] In some embodiments, the selection of a potential rule for the security stage can be performed via a Binary Decision Diagram (BDD) algorithm. The BDD algorithm can provide

fast lookups and strong rule compression to be able to process security rules quickly for each packet. In some embodiments, this can complicate the process of determining the potential forwarding rule volume because information on the individual rule that caused the packet to be allowed or denied can be lost. To optimize rule packing while minimizing the impact to

5 BDD performance, some embodiments include an additional step in the BDD calculation that tracks rule segments. In such an embodiment, each segment can correspond to a volume of space assigned to allow or deny. In some embodiments, such segments provide optimal volume for a potential rule, and in some embodiments, such segments provide sub-optimal volume for a potential rule. Although such segments may not provide optimal volume, they

10 provide a volume sufficient to yield efficient rule packing.

[00174] In some embodiments, a BDD algorithm has two terminal point, which can be, for example, a terminal endpoint for “allow”, and a terminal endpoint for “deny”. While having only these two nodes compresses the BDD table, significant information can be lost due to this compression.

15 [00175] In some embodiments, a BDD algorithm can be compressed while retaining desired information. In such an embodiment, rules are pre-processed to combine adjacent rules. In some embodiments, the rules can be pre-processed to combine any purely adjacent rules. For example, a 0/0 rule allowing source ports 300-400 to port 80 and a 0/0 rule allowing source ports 400-500 to port 80 can be combined to 0/0 allowing source ports 300-500 to port 80.

20 The nodes at the edge of the graph can be evaluated to determine whether each node is an allow node or a deny node. In some embodiments, an identifier can be associated with each of the nodes at the edge of the graph to indicate whether a node is an allow node or a deny node. In some embodiments, this can include coloring the edge of the graph such that each allow node is colored green and each deny node is colored red.

25 [00176] Each edge node of the BDD algorithm can be tied to a set of rules associated with that edge. In the event that an edge node is tied to multiple rules, one of the multiple rules can be selected to be the “primary rule” for that edge node. In some embodiments, the largest of the multiple nodes can be selected to be the primary node. In some embodiments, the rule relevant to the largest number of flows is the largest rule and can be designated the primary

30 rule. In some embodiments, a new terminal node can be created for each of the chosen primary rules, after which BDD compression can continue.

[00177] Applying this compression can reduce the compression capability of the BDD algorithm as it will use more nodes to track rule information, however, this compression enables each endpoint be mapped to a contiguous area (defined by the primary rules) that can be used for determining the rule packing.

5 [00178] The throttle stage include information relating to the subjecting of one or several packets to throttles. In some embodiments, this can include information for determining whether data transmission rates for the packet are within allowed limits. In some embodiments, the security stage can be used to determine and/or identify one or several throttles for application to a packet and/or to the forwarding rule applicable to the packet. In
10 some embodiments, the output key of the throttle stage is the largest block of flows having the same set of throttles.

[00179] The transformation stage can include information to perform any desired and/or necessary rewrites before delivering the packet. This can include, for example, NAT of the overlay packet, resolving MAC addresses when delivering to a host, PAT, or the like. In
15 some embodiments, transformations of a packet can be a function of the routing next hop and/or the sender. In some embodiments, a potential rule for the transformation stage can be crafted so that the same rewrites are required for all packets to which the potential rule is applicable. In some embodiments, such as for egress flows, the output key for the transformation stage can be the same key as the output key for the potential route stage rule.
20 In such an embodiment, the rewrite can be based on the routing destination. In some embodiments, such as for ingress flows, the desired transformation can depend on the sender, and the output key can be limited to the specific VNIC or gateway sending the packet.

[00180] At block 914, one or several NVDs are identified for receipt of the compiled rule. These one or several NVDs can include the NVD from which the packet of block 904 was
25 received. In some embodiments, the one or several NVDs can further include one or several NVDs similar to the NVD from which the packet of block 904 was received.

[00181] At block 916, the compiled rule from block 912 is provided to the one or several NVDs identified in block 914. In some embodiments, the compiled rule can be provided to the one or several NVDs by the server fleet 702, and specifically by the configuration storage
30 fleet 706. In some embodiments, and as a part of providing the compiled rule, the configuration storage fleet 706 can provide information, and specifically can provide identification of the one or several NVDs receiving the compiled rule, and can provide the

compiled rule and/or an identifier of the compiled rule to the pre-fetch fleet 708. In some embodiments, the pre-fetch fleet 708 can update a database identifying which rules are available to each VNIC based on this information.

Exemplary Embodiments of Packet Processing

5 [00182] In a first embodiment, one or several compiled rules can be received by an NVD from the server fleet 702 as indicated in block 802. A first packet can be received by the NVD as indicated in block 804, which first packet can be for delivery to a first VNIC. Attributes of the first packet can be determined by the NVD as indicated in block 806, and the NVD can determine and/or identify an applicable rule based at least in part on the
10 attributes of the first packet as indicated in block 808 and decision step 810. The first packet can then be processed by the NVD according to the applicable rule as indicated in block 812.

[00183] A second packet can be received by the NVD as indicated in block 804, which second packet can be for delivery to, for example, a second VNIC. The NVD can determine attributes of the second packet as indicated in block 806, and can determine that the NVD
15 does not contain or have a rule for processing the second packet as indicated in block 808 and decision step 810. The NVD can then forward the second packet to the server fleet 702 as indicated in block 814.

[00184] The second packet can be received by the server fleet 702 as indicated in block 904, and the server fleet 702 can determine attributes of the packet as indicated in block 906, and
20 can determine a rule for processing the packet as indicated in block 908. The packet can be processed by the server fleet 702 as indicated in block 910, and the server fleet 702 can generate a compiled rule including the rule used to process the second packet. The server fleet 702 can identify one or several NVDs for receipt of the compiled rule, and can provide the compiled rule to those one or several NVDs. In some embodiments, the NVD from which
25 the second packet was sent to the server fleet 702 can receive the compiled rule. In some embodiments, one or several NVDs can be identified as similar to the NVD from which the second packet was sent to the server fleet 702, and the compiled rule can be sent to each of these similar NVDs.

[00185] FIG. 11 depicts a second embodiment of packet processing. As seen in FIG. 11,
30 After the packet is processed, the process 900. The packet is sent from VNIC A 1102, having a first IP address 10.0.0.1, to VNIC B 1104, having a second IP address 10.1.1.1. Each of the

VNICs 1102, 1104 includes a VNIC configuration 1106, 1108, each configuration corresponding to a rule. Each of these configurations includes rules for a plurality of stages, including, a route stage, a security stage, a throttle stage, and a transformation (rewrite) stage.

[00186] Table 1, below, shows processing for VNIC A 1102. Specifically, Table 1 shows the VNIC configuration 1106 of VNIC A 1102 and describes application of that configuration to the packet being sent from VNIC A 1102 to VNIC B 1104. The output key of the VNIC configuration 1106 of VNIC A 1102 is: **Output key: Src IP: 0.0.0.0/0 Dest IP: 10.1.1.1/32 Source port: All Dest port: 80.**

Stage	Description	Output key
Routing	Matches route rule 10.1.1.1/32	<u>Src IP:</u> 0.0.0.0/0 <u>Dest IP:</u> 10.1.1.1/32 <u>Source port:</u> All <u>Dest port:</u> All
Security	Match egress rule 0.0.0.0/0 from all ports to port 80	<u>Src IP:</u> 0.0.0.0/0 <u>Dest IP:</u> 0.0.0.0/0 <u>Source port:</u> All <u>Dest port:</u> 80
Throttle	Standard aggregate throttle only based on route destination	<u>Src IP:</u> 0.0.0.0/0 <u>Dest IP:</u> 10.1.1.1/32 <u>Source port:</u> All <u>Dest port:</u> All
NAT / Rewrite	No rewrites required, based on route destination.	<u>Src IP:</u> 0.0.0.0/0 <u>Dest IP:</u> 10.1.1.1/32 <u>Source port:</u> All <u>Dest port:</u> All

Table 1 Depicts VNIC A processing of an egress packet from host to network.

This includes the VNIC Configuration of VNIC A

[00187] Table 2, below, shows processing for VNIC B 1104. Specifically, Table 2 shows the VNIC configuration 1108 of VNIC B 1104 and describes application of that configuration to the packet being received from VNIC A 1102 by VNIC B 1104. The output key of the VNIC configuration 1108 of VNIC B 1104 is: **Output key: Src IP: 10.0.0.1/32 Dest IP: 10.1.1.1/32 Source port: All Dest port: 80.**

Stage	Description	Output key
Routing	Ingress to <u>VNIC</u>	<u>Src IP</u> : 0.0.0.0/0 <u>Dest IP</u> : 10.1.1.1/32 <u>Source port</u> : All <u>Dest port</u> : All
Security	Match ingress rule <u>rule</u> 10.0.0.0/8 from all ports to port 80	<u>Src IP</u> : 10.0.0.0/8 <u>Dest IP</u> : 0.0.0.0/0 <u>Source port</u> : All <u>Dest port</u> : 80
Throttle	Standard aggregate throttle only based on route destination	<u>Src IP</u> : 0.0.0.0/0 <u>Dest IP</u> : 10.1.1.1/32 <u>Source port</u> : All <u>Dest port</u> : All
NAT / Rewrite	Add MAC and <u>VLAN</u> for delivery to host. Both are specific to sending and receiving IP address.	<u>Src IP</u> : 10.0.0.1/32 <u>Dest IP</u> : 10.1.1.1/32 <u>Source port</u> : All <u>Dest port</u> : All

Table 2 Depicts VNIC B processing of an ingress packet from network to host.

This includes the VNIC Configuration of VNIC B

Exemplary Implementation

5 [00188] As noted above, infrastructure as a service (IaaS) is one particular type of cloud computing. IaaS can be configured to provide virtualized computing resources over a public network (e.g., the Internet). In an IaaS model, a cloud computing provider can host the infrastructure components (e.g., servers, storage devices, network nodes (e.g., hardware), deployment software, platform virtualization (e.g., a hypervisor layer), or the like). In some cases, an IaaS provider may also supply a variety of services to accompany those

10 infrastructure components (e.g., billing, monitoring, logging, load balancing and clustering, etc.). Thus, as these services may be policy-driven, IaaS users may be able to implement policies to drive load balancing to maintain application availability and performance.

[00189] In some instances, IaaS customers may access resources and services through a

15 wide area network (WAN), such as the Internet, and can use the cloud provider's services to install the remaining elements of an application stack. For example, the user can log in to the IaaS platform to create virtual machines (VMs), install operating systems (OSs) on each VM, deploy middleware such as databases, create storage buckets for workloads and backups, and even install enterprise software into that VM. Customers can then use the provider's services

20 to perform various functions, including balancing network traffic, troubleshooting application issues, monitoring performance, managing disaster recovery, etc.

[00190] In most cases, a cloud computing model will require the participation of a cloud provider. The cloud provider may, but need not be, a third-party service that specializes in providing (e.g., offering, renting, selling) IaaS. An entity might also opt to deploy a private cloud, becoming its own provider of infrastructure services.

5 [00191] In some examples, IaaS deployment is the process of putting a new application, or a new version of an application, onto a prepared application server or the like. It may also include the process of preparing the server (e.g., installing libraries, daemons, etc.). This is often managed by the cloud provider, below the hypervisor layer (e.g., the servers, storage, network hardware, and virtualization). Thus, the customer may be responsible for handling
10 (OS), middleware, and/or application deployment (e.g., on self-service virtual machines (e.g., that can be spun up on demand) or the like.

[00192] In some examples, IaaS provisioning may refer to acquiring computers or virtual hosts for use, and even installing needed libraries or services on them. In most cases, deployment does not include provisioning, and the provisioning may need to be performed
15 first.

[00193] In some cases, there are two different challenges for IaaS provisioning. First, there is the initial challenge of provisioning the initial set of infrastructure before anything is running. Second, there is the challenge of evolving the existing infrastructure (e.g., adding new services, changing services, removing services, etc.) once everything has been
20 provisioned. In some cases, these two challenges may be addressed by enabling the configuration of the infrastructure to be defined declaratively. In other words, the infrastructure (e.g., what components are needed and how they interact) can be defined by one or more configuration files. Thus, the overall topology of the infrastructure (e.g., what resources depend on which, and how they each work together) can be described declaratively.
25 In some instances, once the topology is defined, a workflow can be generated that creates and/or manages the different components described in the configuration files.

[00194] In some examples, an infrastructure may have many interconnected elements. For example, there may be one or more virtual private clouds (VPCs) (e.g., a potentially on-demand pool of configurable and/or shared computing resources), also known as a core
30 network. In some examples, there may also be one or more inbound/outbound traffic group rules provisioned to define how the inbound and/or outbound traffic of the network will be set up and one or more virtual machines (VMs). Other infrastructure elements may also be

provisioned, such as a load balancer, a database, or the like. As more and more infrastructure elements are desired and/or added, the infrastructure may incrementally evolve.

[00195] In some instances, continuous deployment techniques may be employed to enable deployment of infrastructure code across various virtual computing environments.

5 Additionally, the described techniques can enable infrastructure management within these environments. In some examples, service teams can write code that is desired to be deployed to one or more, but often many, different production environments (e.g., across various different geographic locations, sometimes spanning the entire world). However, in some examples, the infrastructure on which the code will be deployed must first be set up. In some
10 instances, the provisioning can be done manually, a provisioning tool may be utilized to provision the resources, and/or deployment tools may be utilized to deploy the code once the infrastructure is provisioned.

[00196] FIG. 12 is a block diagram 1200 illustrating an example pattern of an IaaS architecture, according to at least one embodiment. Service operators 1202 can be
15 communicatively coupled to a secure host tenancy 1204 that can include a virtual cloud network (VCN) 1206 and a secure host subnet 1208. In some examples, the service operators 1202 may be using one or more client computing devices, which may be portable handheld devices (e.g., an iPhone®, cellular telephone, an iPad®, computing tablet, a personal digital assistant (PDA)) or wearable devices (e.g., a Google Glass® head mounted display), running
20 software such as Microsoft Windows Mobile®, and/or a variety of mobile operating systems such as iOS, Windows Phone, Android, BlackBerry 8, Palm OS, and the like, and being Internet, e-mail, short message service (SMS), Blackberry®, or other communication protocol enabled. Alternatively, the client computing devices can be general purpose personal computers including, by way of example, personal computers and/or laptop computers
25 running various versions of Microsoft Windows®, Apple Macintosh®, and/or Linux operating systems. The client computing devices can be workstation computers running any of a variety of commercially-available UNIX® or UNIX-like operating systems, including without limitation the variety of GNU/Linux operating systems, such as for example, Google Chrome OS. Alternatively, or in addition, client computing devices may be any other
30 electronic device, such as a thin-client computer, an Internet-enabled gaming system (e.g., a Microsoft Xbox gaming console with or without a Kinect® gesture input device), and/or a personal messaging device, capable of communicating over a network that can access the VCN 1206 and/or the Internet.

[00197] The VCN 1206 can include a local peering gateway (LPG) 1210 that can be communicatively coupled to a secure shell (SSH) VCN 1212 via an LPG 1210 contained in the SSH VCN 1212. The SSH VCN 1212 can include an SSH subnet 1214, and the SSH VCN 1212 can be communicatively coupled to a control plane VCN 1216 via the LPG 1210
 5 contained in the control plane VCN 1216. Also, the SSH VCN 1212 can be communicatively coupled to a data plane VCN 1218 via an LPG 1210. The control plane VCN 1216 and the data plane VCN 1218 can be contained in a service tenancy 1219 that can be owned and/or operated by the IaaS provider.

[00198] The control plane VCN 1216 can include a control plane demilitarized zone (DMZ) tier 1220 that acts as a perimeter network (e.g., portions of a corporate network between the corporate intranet and external networks). The DMZ-based servers may have restricted responsibilities and help keep breaches contained. Additionally, the DMZ tier 1220 can include one or more load balancer (LB) subnet(s) 1222, a control plane app tier 1224 that can include app subnet(s) 1226, a control plane data tier 1228 that can include database (DB) subnet(s) 1230 (e.g., frontend DB subnet(s) and/or backend DB subnet(s)). The LB subnet(s) 1222 contained in the control plane DMZ tier 1220 can be communicatively coupled to the app subnet(s) 1226 contained in the control plane app tier 1224 and an Internet gateway 1234 that can be contained in the control plane VCN 1216, and the app subnet(s) 1226 can be communicatively coupled to the DB subnet(s) 1230 contained in the control plane data tier 1228 and a service gateway 1236 and a network address translation (NAT) gateway 1238.
 15 20 The control plane VCN 1216 can include the service gateway 1236 and the NAT gateway 1238.

[00199] The control plane VCN 1216 can include a data plane mirror app tier 1240 that can include app subnet(s) 1226. The app subnet(s) 1226 contained in the data plane mirror app tier 1240 can include a virtual network interface controller (VNIC) 1242 that can execute a compute instance 1244. The compute instance 1244 can communicatively couple the app subnet(s) 1226 of the data plane mirror app tier 1240 to app subnet(s) 1226 that can be contained in a data plane app tier 1246.
 25

[00200] The data plane VCN 1218 can include the data plane app tier 1246, a data plane DMZ tier 1248, and a data plane data tier 1250. The data plane DMZ tier 1248 can include LB subnet(s) 1222 that can be communicatively coupled to the app subnet(s) 1226 of the data plane app tier 1246 and the Internet gateway 1234 of the data plane VCN 1218. The app
 30

subnet(s) 1226 can be communicatively coupled to the service gateway 1236 of the data plane VCN 1218 and the NAT gateway 1238 of the data plane VCN 1218. The data plane data tier 1250 can also include the DB subnet(s) 1230 that can be communicatively coupled to the app subnet(s) 1226 of the data plane app tier 1246.

5 **[00201]** The Internet gateway 1234 of the control plane VCN 1216 and of the data plane VCN 1218 can be communicatively coupled to a metadata management service 1252 that can be communicatively coupled to public Internet 1254. Public Internet 1254 can be communicatively coupled to the NAT gateway 1238 of the control plane VCN 1216 and of the data plane VCN 1218. The service gateway 1236 of the control plane VCN 1216 and of
10 the data plane VCN 1218 can be communicatively couple to cloud services 1256.

[00202] In some examples, the service gateway 1236 of the control plane VCN 1216 or of the data plane VCN 1218 can make application programming interface (API) calls to cloud services 1256 without going through public Internet 1254. The API calls to cloud services 1256 from the service gateway 1236 can be one-way: the service gateway 1236 can make
15 API calls to cloud services 1256, and cloud services 1256 can send requested data to the service gateway 1236. But, cloud services 1256 may not initiate API calls to the service gateway 1236.

[00203] In some examples, the secure host tenancy 1204 can be directly connected to the service tenancy 1219, which may be otherwise isolated. The secure host subnet 1208 can
20 communicate with the SSH subnet 1214 through an LPG 1210 that may enable two-way communication over an otherwise isolated system. Connecting the secure host subnet 1208 to the SSH subnet 1214 may give the secure host subnet 1208 access to other entities within the service tenancy 1219.

[00204] The control plane VCN 1216 may allow users of the service tenancy 1219 to set up
25 or otherwise provision desired resources. Desired resources provisioned in the control plane VCN 1216 may be deployed or otherwise used in the data plane VCN 1218. In some examples, the control plane VCN 1216 can be isolated from the data plane VCN 1218, and the data plane mirror app tier 1240 of the control plane VCN 1216 can communicate with the data plane app tier 1246 of the data plane VCN 1218 via VNICs 1242 that can be contained
30 in the data plane mirror app tier 1240 and the data plane app tier 1246.

[00205] In some examples, users of the system, or customers, can make requests, for example create, read, update, or delete (CRUD) operations, through public Internet 1254 that can communicate the requests to the metadata management service 1252. The metadata management service 1252 can communicate the request to the control plane VCN 1216 through the Internet gateway 1234. The request can be received by the LB subnet(s) 1222 contained in the control plane DMZ tier 1220. The LB subnet(s) 1222 may determine that the request is valid, and in response to this determination, the LB subnet(s) 1222 can transmit the request to app subnet(s) 1226 contained in the control plane app tier 1224. If the request is validated and requires a call to public Internet 1254, the call to public Internet 1254 may be transmitted to the NAT gateway 1238 that can make the call to public Internet 1254. Memory that may be desired to be stored by the request can be stored in the DB subnet(s) 1230.

[00206] In some examples, the data plane mirror app tier 1240 can facilitate direct communication between the control plane VCN 1216 and the data plane VCN 1218. For example, changes, updates, or other suitable modifications to configuration may be desired to be applied to the resources contained in the data plane VCN 1218. Via a VNIC 1242, the control plane VCN 1216 can directly communicate with, and can thereby execute the changes, updates, or other suitable modifications to configuration to, resources contained in the data plane VCN 1218.

[00207] In some embodiments, the control plane VCN 1216 and the data plane VCN 1218 can be contained in the service tenancy 1219. In this case, the user, or the customer, of the system may not own or operate either the control plane VCN 1216 or the data plane VCN 1218. Instead, the IaaS provider may own or operate the control plane VCN 1216 and the data plane VCN 1218, both of which may be contained in the service tenancy 1219. This embodiment can enable isolation of networks that may prevent users or customers from interacting with other users', or other customers', resources. Also, this embodiment may allow users or customers of the system to store databases privately without needing to rely on public Internet 1254, which may not have a desired level of threat prevention, for storage.

[00208] In other embodiments, the LB subnet(s) 1222 contained in the control plane VCN 1216 can be configured to receive a signal from the service gateway 1236. In this embodiment, the control plane VCN 1216 and the data plane VCN 1218 may be configured to be called by a customer of the IaaS provider without calling public Internet 1254. Customers of the IaaS provider may desire this embodiment since database(s) that the

customers use may be controlled by the IaaS provider and may be stored on the service tenancy 1219, which may be isolated from public Internet 1254.

[00209] FIG. 13 is a block diagram 1300 illustrating another example pattern of an IaaS architecture, according to at least one embodiment. Service operators 1302 (e.g., service operators 1202 of FIG. 12) can be communicatively coupled to a secure host tenancy 1304 (e.g., the secure host tenancy 1204 of FIG. 12) that can include a virtual cloud network (VCN) 1306 (e.g., the VCN 1206 of FIG. 12) and a secure host subnet 1308 (e.g., the secure host subnet 1208 of FIG. 12). The VCN 1306 can include a local peering gateway (LPG) 1310 (e.g., the LPG 1210 of FIG. 12) that can be communicatively coupled to a secure shell (SSH) VCN 1312 (e.g., the SSH VCN 1212 of FIG. 12) via an LPG 1210 contained in the SSH VCN 1312. The SSH VCN 1312 can include an SSH subnet 1314 (e.g., the SSH subnet 1214 of FIG. 12), and the SSH VCN 1312 can be communicatively coupled to a control plane VCN 1316 (e.g., the control plane VCN 1216 of FIG. 12) via an LPG 1310 contained in the control plane VCN 1316. The control plane VCN 1316 can be contained in a service tenancy 1319 (e.g., the service tenancy 1219 of FIG. 12), and the data plane VCN 1318 (e.g., the data plane VCN 1218 of FIG. 12) can be contained in a customer tenancy 1321 that may be owned or operated by users, or customers, of the system.

[00210] The control plane VCN 1316 can include a control plane DMZ tier 1320 (e.g., the control plane DMZ tier 1220 of FIG. 12) that can include LB subnet(s) 1322 (e.g., LB subnet(s) 1222 of FIG. 12), a control plane app tier 1324 (e.g., the control plane app tier 1224 of FIG. 12) that can include app subnet(s) 1326 (e.g., app subnet(s) 1226 of FIG. 12), a control plane data tier 1328 (e.g., the control plane data tier 1228 of FIG. 12) that can include database (DB) subnet(s) 1330 (e.g., similar to DB subnet(s) 1230 of FIG. 12). The LB subnet(s) 1322 contained in the control plane DMZ tier 1320 can be communicatively coupled to the app subnet(s) 1326 contained in the control plane app tier 1324 and an Internet gateway 1334 (e.g., the Internet gateway 1234 of FIG. 12) that can be contained in the control plane VCN 1316, and the app subnet(s) 1326 can be communicatively coupled to the DB subnet(s) 1330 contained in the control plane data tier 1328 and a service gateway 1336 (e.g., the service gateway 1236 of FIG. 12) and a network address translation (NAT) gateway 1338 (e.g., the NAT gateway 1238 of FIG. 12). The control plane VCN 1316 can include the service gateway 1336 and the NAT gateway 1338.

- [00211] The control plane VCN 1316 can include a data plane mirror app tier 1340 (e.g., the data plane mirror app tier 1240 of FIG. 12) that can include app subnet(s) 1326. The app subnet(s) 1326 contained in the data plane mirror app tier 1340 can include a virtual network interface controller (VNIC) 1342 (e.g., the VNIC of 1242) that can execute a compute instance 1344 (e.g., similar to the compute instance 1244 of FIG. 12). The compute instance 1344 can facilitate communication between the app subnet(s) 1326 of the data plane mirror app tier 1340 and the app subnet(s) 1326 that can be contained in a data plane app tier 1346 (e.g., the data plane app tier 1246 of FIG. 12) via the VNIC 1342 contained in the data plane mirror app tier 1340 and the VNIC 1342 contained in the data plane app tier 1346.
- [00212] The Internet gateway 1334 contained in the control plane VCN 1316 can be communicatively coupled to a metadata management service 1352 (e.g., the metadata management service 1252 of FIG. 12) that can be communicatively coupled to public Internet 1354 (e.g., public Internet 1254 of FIG. 12). Public Internet 1354 can be communicatively coupled to the NAT gateway 1338 contained in the control plane VCN 1316. The service gateway 1336 contained in the control plane VCN 1316 can be communicatively couple to cloud services 1356 (e.g., cloud services 1256 of FIG. 12).

- [00213] In some examples, the data plane VCN 1318 can be contained in the customer tenancy 1321. In this case, the IaaS provider may provide the control plane VCN 1316 for each customer, and the IaaS provider may, for each customer, set up a unique compute instance 1344 that is contained in the service tenancy 1319. Each compute instance 1344 may allow communication between the control plane VCN 1316, contained in the service tenancy 1319, and the data plane VCN 1318 that is contained in the customer tenancy 1321. The compute instance 1344 may allow resources, that are provisioned in the control plane VCN 1316 that is contained in the service tenancy 1319, to be deployed or otherwise used in the data plane VCN 1318 that is contained in the customer tenancy 1321.

- [00214] In other examples, the customer of the IaaS provider may have databases that live in the customer tenancy 1321. In this example, the control plane VCN 1316 can include the data plane mirror app tier 1340 that can include app subnet(s) 1326. The data plane mirror app tier 1340 can reside in the data plane VCN 1318, but the data plane mirror app tier 1340 may not live in the data plane VCN 1318. That is, the data plane mirror app tier 1340 may have access to the customer tenancy 1321, but the data plane mirror app tier 1340 may not exist in the data plane VCN 1318 or be owned or operated by the customer of the IaaS provider. The data

plane mirror app tier 1340 may be configured to make calls to the data plane VCN 1318 but may not be configured to make calls to any entity contained in the control plane VCN 1316. The customer may desire to deploy or otherwise use resources in the data plane VCN 1318 that are provisioned in the control plane VCN 1316, and the data plane mirror app tier 1340
5 can facilitate the desired deployment, or other usage of resources, of the customer.

[00215] In some embodiments, the customer of the IaaS provider can apply filters to the data plane VCN 1318. In this embodiment, the customer can determine what the data plane VCN 1318 can access, and the customer may restrict access to public Internet 1354 from the data plane VCN 1318. The IaaS provider may not be able to apply filters or otherwise control
10 access of the data plane VCN 1318 to any outside networks or databases. Applying filters and controls by the customer onto the data plane VCN 1318, contained in the customer tenancy 1321, can help isolate the data plane VCN 1318 from other customers and from public Internet 1354.

[00216] In some embodiments, cloud services 1356 can be called by the service gateway
15 1336 to access services that may not exist on public Internet 1354, on the control plane VCN 1316, or on the data plane VCN 1318. The connection between cloud services 1356 and the control plane VCN 1316 or the data plane VCN 1318 may not be live or continuous. Cloud services 1356 may exist on a different network owned or operated by the IaaS provider. Cloud services 1356 may be configured to receive calls from the service gateway 1336 and
20 may be configured to not receive calls from public Internet 1354. Some cloud services 1356 may be isolated from other cloud services 1356, and the control plane VCN 1316 may be isolated from cloud services 1356 that may not be in the same region as the control plane VCN 1316. For example, the control plane VCN 1316 may be located in “Region 1,” and cloud service “Deployment 12,” may be located in Region 1 and in “Region 2.” If a call to
25 Deployment 12 is made by the service gateway 1336 contained in the control plane VCN 1316 located in Region 1, the call may be transmitted to Deployment 12 in Region 1. In this example, the control plane VCN 1316, or Deployment 12 in Region 1, may not be communicatively coupled to, or otherwise in communication with, Deployment 12 in Region 2.

30 [00217] FIG. 14 is a block diagram 1400 illustrating another example pattern of an IaaS architecture, according to at least one embodiment. Service operators 1402 (e.g., service operators 1202 of FIG. 12) can be communicatively coupled to a secure host tenancy 1404

(e.g., the secure host tenancy 1204 of FIG. 12) that can include a virtual cloud network (VCN) 1406 (e.g., the VCN 1206 of FIG. 12) and a secure host subnet 1408 (e.g., the secure host subnet 1208 of FIG. 12). The VCN 1406 can include an LPG 1410 (e.g., the LPG 1210 of FIG. 12) that can be communicatively coupled to an SSH VCN 1412 (e.g., the SSH VCN 1212 of FIG. 12) via an LPG 1410 contained in the SSH VCN 1412. The SSH VCN 1412 can include an SSH subnet 1414 (e.g., the SSH subnet 1214 of FIG. 12), and the SSH VCN 1412 can be communicatively coupled to a control plane VCN 1416 (e.g., the control plane VCN 1216 of FIG. 12) via an LPG 1410 contained in the control plane VCN 1416 and to a data plane VCN 1418 (e.g., the data plane 1218 of FIG. 12) via an LPG 1410 contained in the data plane VCN 1418. The control plane VCN 1416 and the data plane VCN 1418 can be contained in a service tenancy 1419 (e.g., the service tenancy 1219 of FIG. 12).

[00218] The control plane VCN 1416 can include a control plane DMZ tier 1420 (e.g., the control plane DMZ tier 1220 of FIG. 12) that can include load balancer (LB) subnet(s) 1422 (e.g., LB subnet(s) 1222 of FIG. 12), a control plane app tier 1424 (e.g., the control plane app tier 1224 of FIG. 12) that can include app subnet(s) 1426 (e.g., similar to app subnet(s) 1226 of FIG. 12), a control plane data tier 1428 (e.g., the control plane data tier 1228 of FIG. 12) that can include DB subnet(s) 1430. The LB subnet(s) 1422 contained in the control plane DMZ tier 1420 can be communicatively coupled to the app subnet(s) 1426 contained in the control plane app tier 1424 and to an Internet gateway 1434 (e.g., the Internet gateway 1234 of FIG. 12) that can be contained in the control plane VCN 1416, and the app subnet(s) 1426 can be communicatively coupled to the DB subnet(s) 1430 contained in the control plane data tier 1428 and to a service gateway 1436 (e.g., the service gateway of FIG. 12) and a network address translation (NAT) gateway 1438 (e.g., the NAT gateway 1238 of FIG. 12). The control plane VCN 1416 can include the service gateway 1436 and the NAT gateway 1438.

[00219] The data plane VCN 1418 can include a data plane app tier 1446 (e.g., the data plane app tier 1246 of FIG. 12), a data plane DMZ tier 1448 (e.g., the data plane DMZ tier 1248 of FIG. 12), and a data plane data tier 1450 (e.g., the data plane data tier 1250 of FIG. 12). The data plane DMZ tier 1448 can include LB subnet(s) 1422 that can be communicatively coupled to trusted app subnet(s) 1460 and untrusted app subnet(s) 1462 of the data plane app tier 1446 and the Internet gateway 1434 contained in the data plane VCN 1418. The trusted app subnet(s) 1460 can be communicatively coupled to the service gateway 1436 contained in the data plane VCN 1418, the NAT gateway 1438 contained in the data plane VCN 1418, and DB subnet(s) 1430 contained in the data plane data tier 1450. The

untrusted app subnet(s) 1462 can be communicatively coupled to the service gateway 1436 contained in the data plane VCN 1418 and DB subnet(s) 1430 contained in the data plane data tier 1450. The data plane data tier 1450 can include DB subnet(s) 1430 that can be communicatively coupled to the service gateway 1436 contained in the data plane VCN 1418.

5 **[00220]** The untrusted app subnet(s) 1462 can include one or more primary VNICs 1464(1)-(N) that can be communicatively coupled to tenant virtual machines (VMs) 1466(1)-(N). Each tenant VM 1466(1)-(N) can be communicatively coupled to a respective app subnet 1467(1)-(N) that can be contained in respective container egress VCNs 1468(1)-(N) that can be contained in respective customer tenancies 1470(1)-(N). Respective secondary VNICs
10 1472(1)-(N) can facilitate communication between the untrusted app subnet(s) 1462 contained in the data plane VCN 1418 and the app subnet contained in the container egress VCNs 1468(1)-(N). Each container egress VCNs 1468(1)-(N) can include a NAT gateway 1438 that can be communicatively coupled to public Internet 1454 (e.g., public Internet 1254 of FIG. 12).

15 **[00221]** The Internet gateway 1434 contained in the control plane VCN 1416 and contained in the data plane VCN 1418 can be communicatively coupled to a metadata management service 1452 (e.g., the metadata management system 1252 of FIG. 12) that can be communicatively coupled to public Internet 1454. Public Internet 1454 can be communicatively coupled to the NAT gateway 1438 contained in the control plane VCN
20 1416 and contained in the data plane VCN 1418. The service gateway 1436 contained in the control plane VCN 1416 and contained in the data plane VCN 1418 can be communicatively couple to cloud services 1456.

[00222] In some embodiments, the data plane VCN 1418 can be integrated with customer tenancies 1470. This integration can be useful or desirable for customers of the IaaS provider
25 in some cases such as a case that may desire support when executing code. The customer may provide code to run that may be destructive, may communicate with other customer resources, or may otherwise cause undesirable effects. In response to this, the IaaS provider may determine whether to run code given to the IaaS provider by the customer.

[00223] In some examples, the customer of the IaaS provider may grant temporary network
30 access to the IaaS provider and request a function to be attached to the data plane app tier 1446. Code to run the function may be executed in the VMs 1466(1)-(N), and the code may not be configured to run anywhere else on the data plane VCN 1418. Each VM 1466(1)-(N)

may be connected to one customer tenancy 1470. Respective containers 1471(1)-(N) contained in the VMs 1466(1)-(N) may be configured to run the code. In this case, there can be a dual isolation (e.g., the containers 1471(1)-(N) running code, where the containers 1471(1)-(N) may be contained in at least the VM 1466(1)-(N) that are contained in the

5 untrusted app subnet(s) 1462), which may help prevent incorrect or otherwise undesirable code from damaging the network of the IaaS provider or from damaging a network of a different customer. The containers 1471(1)-(N) may be communicatively coupled to the customer tenancy 1470 and may be configured to transmit or receive data from the customer tenancy 1470. The containers 1471(1)-(N) may not be configured to transmit or receive data

10 from any other entity in the data plane VCN 1418. Upon completion of running the code, the IaaS provider may kill or otherwise dispose of the containers 1471(1)-(N).

[00224] In some embodiments, the trusted app subnet(s) 1460 may run code that may be owned or operated by the IaaS provider. In this embodiment, the trusted app subnet(s) 1460 may be communicatively coupled to the DB subnet(s) 1430 and be configured to execute

15 CRUD operations in the DB subnet(s) 1430. The untrusted app subnet(s) 1462 may be communicatively coupled to the DB subnet(s) 1430, but in this embodiment, the untrusted app subnet(s) may be configured to execute read operations in the DB subnet(s) 1430. The containers 1471(1)-(N) that can be contained in the VM 1466(1)-(N) of each customer and that may run code from the customer may not be communicatively coupled with the DB

20 subnet(s) 1430.

[00225] In other embodiments, the control plane VCN 1416 and the data plane VCN 1418 may not be directly communicatively coupled. In this embodiment, there may be no direct communication between the control plane VCN 1416 and the data plane VCN 1418. However, communication can occur indirectly through at least one method. An LPG 1410

25 may be established by the IaaS provider that can facilitate communication between the control plane VCN 1416 and the data plane VCN 1418. In another example, the control plane VCN 1416 or the data plane VCN 1418 can make a call to cloud services 1456 via the service gateway 1436. For example, a call to cloud services 1456 from the control plane VCN 1416 can include a request for a service that can communicate with the data plane VCN 1418.

30 **[00226]** FIG. 15 is a block diagram 1500 illustrating another example pattern of an IaaS architecture, according to at least one embodiment. Service operators 1502 (e.g., service operators 1202 of FIG. 12) can be communicatively coupled to a secure host tenancy 1504

(e.g., the secure host tenancy 1204 of FIG. 12) that can include a virtual cloud network (VCN) 1506 (e.g., the VCN 1206 of FIG. 12) and a secure host subnet 1508 (e.g., the secure host subnet 1208 of FIG. 12). The VCN 1506 can include an LPG 1510 (e.g., the LPG 1210 of FIG. 12) that can be communicatively coupled to an SSH VCN 1512 (e.g., the SSH VCN 1212 of FIG. 12) via an LPG 1510 contained in the SSH VCN 1512. The SSH VCN 1512 can include an SSH subnet 1514 (e.g., the SSH subnet 1214 of FIG. 12), and the SSH VCN 1512 can be communicatively coupled to a control plane VCN 1516 (e.g., the control plane VCN 1216 of FIG. 12) via an LPG 1510 contained in the control plane VCN 1516 and to a data plane VCN 1518 (e.g., the data plane 1218 of FIG. 12) via an LPG 1510 contained in the data plane VCN 1518. The control plane VCN 1516 and the data plane VCN 1518 can be contained in a service tenancy 1519 (e.g., the service tenancy 1219 of FIG. 12).

[00227] The control plane VCN 1516 can include a control plane DMZ tier 1520 (e.g., the control plane DMZ tier 1220 of FIG. 12) that can include LB subnet(s) 1522 (e.g., LB subnet(s) 1222 of FIG. 12), a control plane app tier 1524 (e.g., the control plane app tier 1224 of FIG. 12) that can include app subnet(s) 1526 (e.g., app subnet(s) 1226 of FIG. 12), a control plane data tier 1528 (e.g., the control plane data tier 1228 of FIG. 12) that can include DB subnet(s) 1530 (e.g., DB subnet(s) 1430 of FIG. 14). The LB subnet(s) 1522 contained in the control plane DMZ tier 1520 can be communicatively coupled to the app subnet(s) 1526 contained in the control plane app tier 1524 and to an Internet gateway 1534 (e.g., the Internet gateway 1234 of FIG. 12) that can be contained in the control plane VCN 1516, and the app subnet(s) 1526 can be communicatively coupled to the DB subnet(s) 1530 contained in the control plane data tier 1528 and to a service gateway 1536 (e.g., the service gateway of FIG. 12) and a network address translation (NAT) gateway 1538 (e.g., the NAT gateway 1238 of FIG. 12). The control plane VCN 1516 can include the service gateway 1536 and the NAT gateway 1538.

[00228] The data plane VCN 1518 can include a data plane app tier 1546 (e.g., the data plane app tier 1246 of FIG. 12), a data plane DMZ tier 1548 (e.g., the data plane DMZ tier 1248 of FIG. 12), and a data plane data tier 1550 (e.g., the data plane data tier 1250 of FIG. 12). The data plane DMZ tier 1548 can include LB subnet(s) 1522 that can be communicatively coupled to trusted app subnet(s) 1560 (e.g., trusted app subnet(s) 1460 of FIG. 14) and untrusted app subnet(s) 1562 (e.g., untrusted app subnet(s) 1462 of FIG. 14) of the data plane app tier 1546 and the Internet gateway 1534 contained in the data plane VCN 1518. The trusted app subnet(s) 1560 can be communicatively coupled to the service gateway

1536 contained in the data plane VCN 1518, the NAT gateway 1538 contained in the data plane VCN 1518, and DB subnet(s) 1530 contained in the data plane data tier 1550. The untrusted app subnet(s) 1562 can be communicatively coupled to the service gateway 1536 contained in the data plane VCN 1518 and DB subnet(s) 1530 contained in the data plane data tier 1550. The data plane data tier 1550 can include DB subnet(s) 1530 that can be communicatively coupled to the service gateway 1536 contained in the data plane VCN 1518.

[00229] The untrusted app subnet(s) 1562 can include primary VNICs 1564(1)-(N) that can be communicatively coupled to tenant virtual machines (VMs) 1566(1)-(N) residing within the untrusted app subnet(s) 1562. Each tenant VM 1566(1)-(N) can run code in a respective container 1567(1)-(N), and be communicatively coupled to an app subnet 1526 that can be contained in a data plane app tier 1546 that can be contained in a container egress VCN 1568. Respective secondary VNICs 1572(1)-(N) can facilitate communication between the untrusted app subnet(s) 1562 contained in the data plane VCN 1518 and the app subnet contained in the container egress VCN 1568. The container egress VCN can include a NAT gateway 1538 that can be communicatively coupled to public Internet 1554 (e.g., public Internet 1254 of FIG. 12).

[00230] The Internet gateway 1534 contained in the control plane VCN 1516 and contained in the data plane VCN 1518 can be communicatively coupled to a metadata management service 1552 (e.g., the metadata management system 1252 of FIG. 12) that can be communicatively coupled to public Internet 1554. Public Internet 1554 can be communicatively coupled to the NAT gateway 1538 contained in the control plane VCN 1516 and contained in the data plane VCN 1518. The service gateway 1536 contained in the control plane VCN 1516 and contained in the data plane VCN 1518 can be communicatively couple to cloud services 1556.

[00231] In some examples, the pattern illustrated by the architecture of block diagram 1500 of FIG. 15 may be considered an exception to the pattern illustrated by the architecture of block diagram 1400 of FIG. 14 and may be desirable for a customer of the IaaS provider if the IaaS provider cannot directly communicate with the customer (e.g., a disconnected region). The respective containers 1567(1)-(N) that are contained in the VMs 1566(1)-(N) for each customer can be accessed in real-time by the customer. The containers 1567(1)-(N) may be configured to make calls to respective secondary VNICs 1572(1)-(N) contained in app subnet(s) 1526 of the data plane app tier 1546 that can be contained in the container egress

VCN 1568. The secondary VNICs 1572(1)-(N) can transmit the calls to the NAT gateway 1538 that may transmit the calls to public Internet 1554. In this example, the containers 1567(1)-(N) that can be accessed in real-time by the customer can be isolated from the control plane VCN 1516 and can be isolated from other entities contained in the data plane VCN 1518. The containers 1567(1)-(N) may also be isolated from resources from other customers.

[00232] In other examples, the customer can use the containers 1567(1)-(N) to call cloud services 1556. In this example, the customer may run code in the containers 1567(1)-(N) that requests a service from cloud services 1556. The containers 1567(1)-(N) can transmit this request to the secondary VNICs 1572(1)-(N) that can transmit the request to the NAT gateway that can transmit the request to public Internet 1554. Public Internet 1554 can transmit the request to LB subnet(s) 1522 contained in the control plane VCN 1516 via the Internet gateway 1534. In response to determining the request is valid, the LB subnet(s) can transmit the request to app subnet(s) 1526 that can transmit the request to cloud services 1556 via the service gateway 1536.

[00233] It should be appreciated that IaaS architectures 1200, 1300, 1400, 1500 depicted in the figures may have other components than those depicted. Further, the embodiments shown in the figures are only some examples of a cloud infrastructure system that may incorporate an embodiment of the disclosure. In some other embodiments, the IaaS systems may have more or fewer components than shown in the figures, may combine two or more components, or may have a different configuration or arrangement of components.

[00234] In certain embodiments, the IaaS systems described herein may include a suite of applications, middleware, and database service offerings that are delivered to a customer in a self-service, subscription-based, elastically scalable, reliable, highly available, and secure manner. An example of such an IaaS system is the Oracle Cloud Infrastructure (OCI) provided by the present assignee.

[00235] FIG. 16 illustrates an example computer system 1600, in which various embodiments may be implemented. The system 1600 may be used to implement any of the computer systems described above. As shown in the figure, computer system 1600 includes a processing unit 1604 that communicates with a number of peripheral subsystems via a bus subsystem 1602. These peripheral subsystems may include a processing acceleration unit 1606, an I/O subsystem 1608, a storage subsystem 1618 and a communications subsystem

1624. Storage subsystem 1618 includes tangible computer-readable storage media 1622 and a system memory 1610.

[00236] Bus subsystem 1602 provides a mechanism for letting the various components and subsystems of computer system 1600 communicate with each other as intended. Although
5 bus subsystem 1602 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple buses. Bus subsystem 1602 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. For example, such architectures may include an Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus,
10 Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus, which can be implemented as a Mezzanine bus manufactured to the IEEE P1386.1 standard.

[00237] Processing unit 1604, which can be implemented as one or more integrated circuits (e.g., a conventional microprocessor or microcontroller), controls the operation of computer
15 system 1600. One or more processors may be included in processing unit 1604. These processors may include single core or multicore processors. In certain embodiments, processing unit 1604 may be implemented as one or more independent processing units 1632 and/or 1634 with single or multicore processors included in each processing unit. In other embodiments, processing unit 1604 may also be implemented as a quad-core processing unit
20 formed by integrating two dual-core processors into a single chip.

[00238] In various embodiments, processing unit 1604 can execute a variety of programs in response to program code and can maintain multiple concurrently executing programs or processes. At any given time, some or all of the program code to be executed can be resident in processor(s) 1604 and/or in storage subsystem 1618. Through suitable programming,
25 processor(s) 1604 can provide various functionalities described above. Computer system 1600 may additionally include a processing acceleration unit 1606, which can include a digital signal processor (DSP), a special-purpose processor, and/or the like.

[00239] I/O subsystem 1608 may include user interface input devices and user interface output devices. User interface input devices may include a keyboard, pointing devices such as
30 a mouse or trackball, a touchpad or touch screen incorporated into a display, a scroll wheel, a click wheel, a dial, a button, a switch, a keypad, audio input devices with voice command recognition systems, microphones, and other types of input devices. User interface input

- devices may include, for example, motion sensing and/or gesture recognition devices such as the Microsoft Kinect® motion sensor that enables users to control and interact with an input device, such as the Microsoft Xbox® 360 game controller, through a natural user interface using gestures and spoken commands. User interface input devices may also include eye
- 5 gesture recognition devices such as the Google Glass® blink detector that detects eye activity (e.g., 'blinking' while taking pictures and/or making a menu selection) from users and transforms the eye gestures as input into an input device (e.g., Google Glass®). Additionally, user interface input devices may include voice recognition sensing devices that enable users to interact with voice recognition systems (e.g., Siri® navigator), through voice commands.
- 10 **[00240]** User interface input devices may also include, without limitation, three dimensional (3D) mice, joysticks or pointing sticks, gamepads and graphic tablets, and audio/visual devices such as speakers, digital cameras, digital camcorders, portable media players, webcams, image scanners, fingerprint scanners, barcode reader 3D scanners, 3D printers, laser rangefinders, and eye gaze tracking devices. Additionally, user interface input devices
- 15 may include, for example, medical imaging input devices such as computed tomography, magnetic resonance imaging, position emission tomography, medical ultrasonography devices. User interface input devices may also include, for example, audio input devices such as MIDI keyboards, digital musical instruments and the like.
- [00241]** User interface output devices may include a display subsystem, indicator lights, or
- 20 non-visual displays such as audio output devices, etc. The display subsystem may be a cathode ray tube (CRT), a flat-panel device, such as that using a liquid crystal display (LCD) or plasma display, a projection device, a touch screen, and the like. In general, use of the term "output device" is intended to include all possible types of devices and mechanisms for outputting information from computer system 1600 to a user or other computer. For example,
- 25 user interface output devices may include, without limitation, a variety of display devices that visually convey text, graphics and audio/video information such as monitors, printers, speakers, headphones, automotive navigation systems, plotters, voice output devices, and modems.
- [00242]** Computer system 1600 may comprise a storage subsystem 1618 that comprises
- 30 software elements, shown as being currently located within a system memory 1610. System memory 1610 may store program instructions that are loadable and executable on processing unit 1604, as well as data generated during the execution of these programs.

[00243] Depending on the configuration and type of computer system 1600, system memory 1610 may be volatile (such as random access memory (RAM)) and/or non-volatile (such as read-only memory (ROM), flash memory, etc.) The RAM typically contains data and/or program modules that are immediately accessible to and/or presently being operated and executed by processing unit 1604. In some implementations, system memory 1610 may include multiple different types of memory, such as static random access memory (SRAM) or dynamic random access memory (DRAM). In some implementations, a basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within computer system 1600, such as during start-up, may typically be stored in the ROM. By way of example, and not limitation, system memory 1610 also illustrates application programs 1612, which may include client applications, Web browsers, mid-tier applications, relational database management systems (RDBMS), etc., program data 1614, and an operating system 1616. By way of example, operating system 1616 may include various versions of Microsoft Windows®, Apple Macintosh®, and/or Linux operating systems, a variety of commercially-available UNIX® or UNIX-like operating systems (including without limitation the variety of GNU/Linux operating systems, the Google Chrome® OS, and the like) and/or mobile operating systems such as iOS, Windows® Phone, Android® OS, BlackBerry® OS, and Palm® OS operating systems.

[00244] Storage subsystem 1618 may also provide a tangible computer-readable storage medium for storing the basic programming and data constructs that provide the functionality of some embodiments. Software (programs, code modules, instructions) that when executed by a processor provide the functionality described above may be stored in storage subsystem 1618. These software modules or instructions may be executed by processing unit 1604. Storage subsystem 1618 may also provide a repository for storing data used in accordance with the present disclosure.

[00245] Storage subsystem 1600 may also include a computer-readable storage media reader 1620 that can further be connected to computer-readable storage media 1622. Together and, optionally, in combination with system memory 1610, computer-readable storage media 1622 may comprehensively represent remote, local, fixed, and/or removable storage devices plus storage media for temporarily and/or more permanently containing, storing, transmitting, and retrieving computer-readable information.

[00246] Computer-readable storage media 1622 containing code, or portions of code, can also include any appropriate media known or used in the art, including storage media and communication media, such as but not limited to, volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage and/or
5 transmission of information. This can include tangible computer-readable storage media such as RAM, ROM, electronically erasable programmable ROM (EEPROM), flash memory or other memory technology, CD-ROM, digital versatile disk (DVD), or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or other tangible computer readable media. This can also include nontangible computer-
10 readable media, such as data signals, data transmissions, or any other medium which can be used to transmit the desired information and which can be accessed by computing system 1600.

[00247] By way of example, computer-readable storage media 1622 may include a hard disk drive that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic
15 disk drive that reads from or writes to a removable, nonvolatile magnetic disk, and an optical disk drive that reads from or writes to a removable, nonvolatile optical disk such as a CD ROM, DVD, and Blu-Ray® disk, or other optical media. Computer-readable storage media 1622 may include, but is not limited to, Zip® drives, flash memory cards, universal serial bus (USB) flash drives, secure digital (SD) cards, DVD disks, digital video tape, and the like.
20 Computer-readable storage media 1622 may also include, solid-state drives (SSD) based on non-volatile memory such as flash-memory based SSDs, enterprise flash drives, solid state ROM, and the like, SSDs based on volatile memory such as solid state RAM, dynamic RAM, static RAM, DRAM-based SSDs, magnetoresistive RAM (MRAM) SSDs, and hybrid SSDs that use a combination of DRAM and flash memory based SSDs. The disk drives and their
25 associated computer-readable media may provide non-volatile storage of computer-readable instructions, data structures, program modules, and other data for computer system 1600.

[00248] Communications subsystem 1624 provides an interface to other computer systems and networks. Communications subsystem 1624 serves as an interface for receiving data from and transmitting data to other systems from computer system 1600. For example,
30 communications subsystem 1624 may enable computer system 1600 to connect to one or more devices via the Internet. In some embodiments communications subsystem 1624 can include radio frequency (RF) transceiver components for accessing wireless voice and/or data networks (e.g., using cellular telephone technology, advanced data network technology, such

as 3G, 4G or EDGE (enhanced data rates for global evolution), WiFi (IEEE 802.11 family standards, or other mobile communication technologies, or any combination thereof), global positioning system (GPS) receiver components, and/or other components. In some embodiments communications subsystem 1624 can provide wired network connectivity (e.g., Ethernet) in addition to or instead of a wireless interface.

[00249] In some embodiments, communications subsystem 1624 may also receive input communication in the form of structured and/or unstructured data feeds 1626, event streams 1628, event updates 1630, and the like on behalf of one or more users who may use computer system 1600.

10 [00250] By way of example, communications subsystem 1624 may be configured to receive data feeds 1626 in real-time from users of social networks and/or other communication services such as Twitter® feeds, Facebook® updates, web feeds such as Rich Site Summary (RSS) feeds, and/or real-time updates from one or more third party information sources.

15 [00251] Additionally, communications subsystem 1624 may also be configured to receive data in the form of continuous data streams, which may include event streams 1628 of real-time events and/or event updates 1630, that may be continuous or unbounded in nature with no explicit end. Examples of applications that generate continuous data may include, for example, sensor data applications, financial tickers, network performance measuring tools (e.g., network monitoring and traffic management applications), clickstream analysis tools, automobile traffic monitoring, and the like.

[00252] Communications subsystem 1624 may also be configured to output the structured and/or unstructured data feeds 1626, event streams 1628, event updates 1630, and the like to one or more databases that may be in communication with one or more streaming data source computers coupled to computer system 1600.

25 [00253] Computer system 1600 can be one of various types, including a handheld portable device (e.g., an iPhone® cellular phone, an iPad® computing tablet, a PDA), a wearable device (e.g., a Google Glass® head mounted display), a PC, a workstation, a mainframe, a kiosk, a server rack, or any other data processing system.

30 [00254] Due to the ever-changing nature of computers and networks, the description of computer system 1600 depicted in the figure is intended only as a specific example. Many other configurations having more or fewer components than the system depicted in the figure

are possible. For example, customized hardware might also be used and/or particular elements might be implemented in hardware, firmware, software (including applets), or a combination. Further, connection to other computing devices, such as network input/output devices, may be employed. Based on the disclosure and teachings provided herein, a person
5 of ordinary skill in the art will appreciate other ways and/or methods to implement the various embodiments.

[00255] Although specific embodiments have been described, various modifications, alterations, alternative constructions, and equivalents are also encompassed within the scope of the disclosure. Embodiments are not restricted to operation within certain specific data
10 processing environments, but are free to operate within a plurality of data processing environments. Additionally, although embodiments have been described using a particular series of transactions and steps, it should be apparent to those skilled in the art that the scope of the present disclosure is not limited to the described series of transactions and steps. Various features and aspects of the above-described embodiments may be used individually
15 or jointly.

[00256] Further, while embodiments have been described using a particular combination of hardware and software, it should be recognized that other combinations of hardware and software are also within the scope of the present disclosure. Embodiments may be implemented only in hardware, or only in software, or using combinations thereof. The
20 various processes described herein can be implemented on the same processor or different processors in any combination. Accordingly, where components or modules are described as being configured to perform certain operations, such configuration can be accomplished, e.g., by designing electronic circuits to perform the operation, by programming programmable electronic circuits (such as microprocessors) to perform the operation, or any combination
25 thereof. Processes can communicate using a variety of techniques including but not limited to conventional techniques for inter process communication, and different pairs of processes may use different techniques, or the same pair of processes may use different techniques at different times.

[00257] The specification and drawings are, accordingly, to be regarded in an illustrative
30 rather than a restrictive sense. It will, however, be evident that additions, subtractions, deletions, and other modifications and changes may be made thereunto without departing from the broader spirit and scope as set forth in the claims. Thus, although specific disclosure

embodiments have been described, these are not intended to be limiting. Various modifications and equivalents are within the scope of the following claims.

[00258] The use of the terms “a” and “an” and “the” and similar referents in the context of describing the disclosed embodiments (especially in the context of the following claims) are to be construed to cover both the singular and the plural, unless otherwise indicated herein or clearly contradicted by context. The terms “comprising,” “having,” “including,” and “containing” are to be construed as open-ended terms (i.e., meaning “including, but not limited to,”) unless otherwise noted. The term “connected” is to be construed as partly or wholly contained within, attached to, or joined together, even if there is something intervening. Recitation of ranges of values herein are merely intended to serve as a shorthand method of referring individually to each separate value falling within the range, unless otherwise indicated herein and each separate value is incorporated into the specification as if it were individually recited herein. All methods described herein can be performed in any suitable order unless otherwise indicated herein or otherwise clearly contradicted by context. The use of any and all examples, or exemplary language (e.g., “such as”) provided herein, is intended merely to better illuminate embodiments and does not pose a limitation on the scope of the disclosure unless otherwise claimed. No language in the specification should be construed as indicating any non-claimed element as essential to the practice of the disclosure.

[00259] Disjunctive language such as the phrase “at least one of X, Y, or Z,” unless specifically stated otherwise, is intended to be understood within the context as used in general to present that an item, term, etc., may be either X, Y, or Z, or any combination thereof (e.g., X, Y, and/or Z). Thus, such disjunctive language is not generally intended to, and should not, imply that certain embodiments require at least one of X, at least one of Y, or at least one of Z to each be present.

[00260] Preferred embodiments of this disclosure are described herein, including the best mode known for carrying out the disclosure. Variations of those preferred embodiments may become apparent to those of ordinary skill in the art upon reading the foregoing description. Those of ordinary skill should be able to employ such variations as appropriate and the disclosure may be practiced otherwise than as specifically described herein. Accordingly, this disclosure includes all modifications and equivalents of the subject matter recited in the claims appended hereto as permitted by applicable law. Moreover, any combination of the

above-described elements in all possible variations thereof is encompassed by the disclosure unless otherwise indicated herein.

[00261] All references, including publications, patent applications, and patents, cited herein are hereby incorporated by reference to the same extent as if each reference were individually
5 and specifically indicated to be incorporated by reference and were set forth in its entirety herein.

[00262] In the foregoing specification, aspects of the disclosure are described with reference to specific embodiments thereof, but those skilled in the art will recognize that the disclosure is not limited thereto. Various features and aspects of the above-described disclosure may be
10 used individually or jointly. Further, embodiments can be utilized in any number of environments and applications beyond those described herein without departing from the broader spirit and scope of the specification. The specification and drawings are, accordingly, to be regarded as illustrative rather than restrictive.

15

WHAT IS CLAIMED IS:

1. A method comprising:
receiving at least one compiled rule at a first Network Virtualization Device (“NVD”), each of the at least one compiled rules is applicable to a class of packets received
5 by the first NVD for delivery to a Virtualized Network Interface Card (“VNIC”);
receiving a first packet at the first NVD for delivery to a first VNIC;
determining with the first NVD that a first rule of the at least one compiled rule is applicable to the first packet; and
processing with the first NVD the first packet according to the first rule.
- 10 2. The method of claim 1, wherein the first packet is processed according to the first rule and according to at least one attribute of the first packet.
3. The method of claim 2, wherein the at least one attribute of the packet comprises at least one of: a destination address, and a source address.
4. The method of claim 3, wherein the source address comprises at least
15 one of: a source IP address; and a source port identifier.
5. The method of claim 3, wherein the destination address comprises at least one of: a destination IP address; and a destination port identifier.
6. The method of claim 1, wherein each of the at least one compiled rule comprises: a flow key; a security policy; and mapping information.
- 20 7. The method of claim 6, wherein the security policy comprises: a local security policy; and a remote security policy.
8. The method of claim 6, wherein each of the at least one compiled rule further comprises: packet transformation information; and throttle information.
9. The method of claim 1, further comprising:
25 receiving a second packet at the first NVD for delivery to a second VNIC;
determining with the first NVD that the first NVD does not have a rule for processing the second packet; and

forwarding the second packet to at least one server, wherein the at least one server is configured to process the second packet.

10. The method of claim 9, wherein the at least one server comprises configuration information for a set of VNICs in at least one virtual network.

5 11. The method of claim 10, further comprising:
determining, with the at least one server, a rule for processing the second packet, wherein the rule is determined based on the configuration information and at least one attribute of the second packet; and
processing the second packet with the at least one server according to the
10 determined rule.

12. The method of claim 11, wherein the at least one attribute of the packet comprises at least one of: a destination address, and a source address.

13. The method of claim 12, wherein the source address comprises at least one of: a source IP address; and a source port identifier.

15 14. The method of claim 12, wherein the destination address comprises at least one of: a destination IP address; and a destination port identifier.

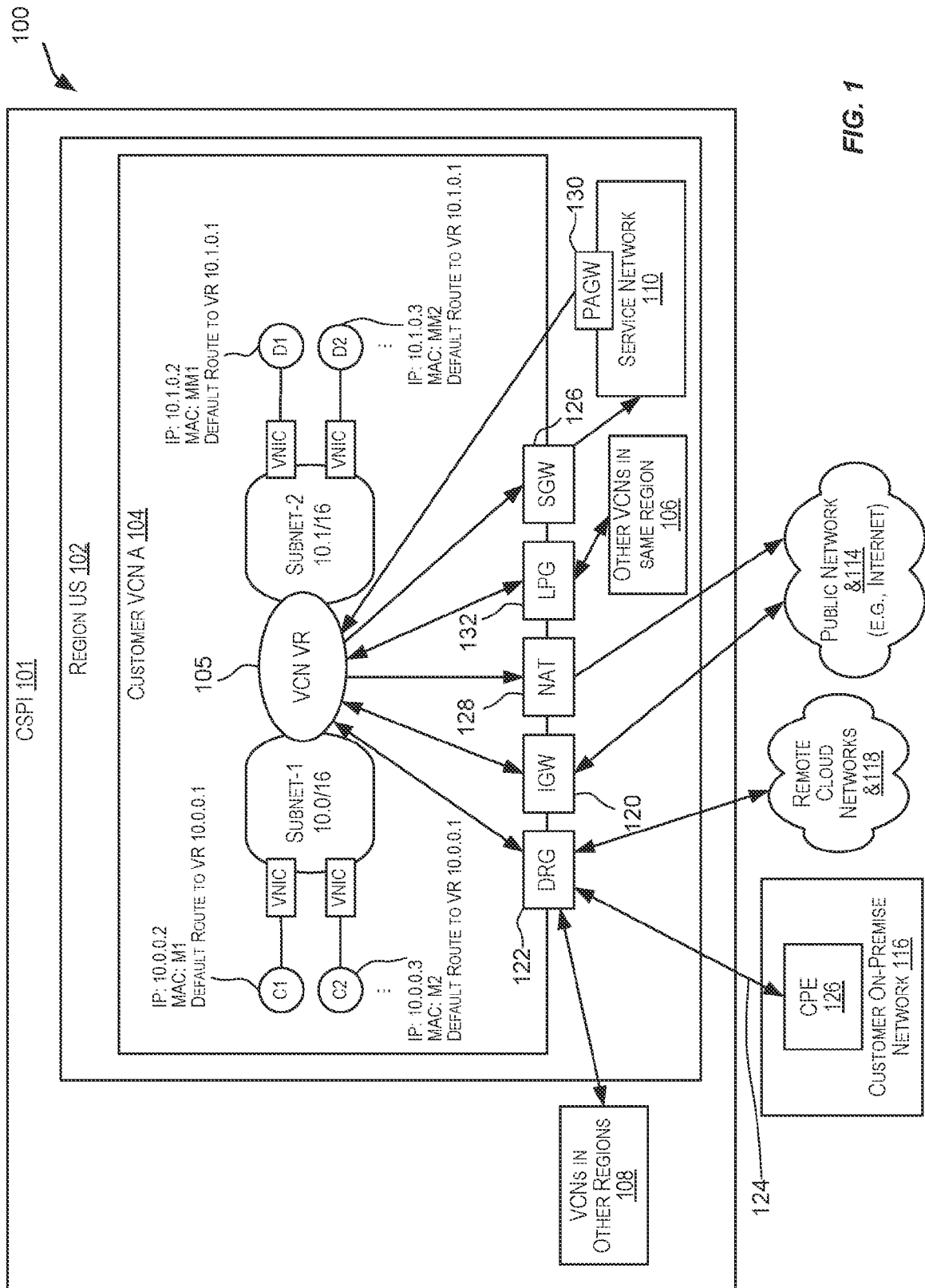
15. The method of claim 11, further comprising:
generating a new compiled rule, the new compiled rule including the second rule, wherein generating the new compiled rule comprises:
20 identifying a potential rule for each of a plurality of stages, wherein each of the potential rules is relevant to a plurality of packet flows; and
forming the new compiled rule from an intersection between the potential rules for each of the plurality of stages of the flow; and
providing the new compiled rule to the first NVD.

25 16. The method of claim 15, further comprising: identifying a plurality of NVDs as similar to the first NVD; and providing the new compiled rule to the plurality of NVDs similar to the first NVD.

17. The method of claim 15, wherein the plurality of stages comprise two or more of: a routing stage; a security stage; a throttle stage; and a transformation stage.

18. A system comprising:
at least one server comprising a plurality of compiled rules; and
a first Network Virtualization Device (“NVD”), wherein the NVD is
configured to:
- 5 receive at least one compiled rule, wherein each of the at least one
compiled rules is applicable to a class of packets received by the first NVD for
delivery to a Virtualized Network Interface Card (“VNIC”);
receive a first packet for delivery to a first VNIC;
determine that a first rule of the at least one compiled rule is applicable
10 to the first packet; and
process the first packet according to the first rule.
19. The system of claim 18, wherein the first NVD is further configured
to:
- 15 receive a second packet for delivery to a second VNIC;
determine that the first NVD does not have a rule for processing the second
packet; and
forward the second packet to the at least one server,
wherein the at least one server comprises configuration information for a set of
VNICs in at least one virtual network, and wherein the at least one server is configured to:
- 20 determine a rule for processing the second packet, wherein the rule is
determined based on the configuration information and at least one attribute of the
second packet; and
process the second packet according to the determined rule.
20. A non-transitory computer-readable storage medium storing a plurality
25 of instructions executable by one or more processors of a first Network Virtualization Device
 (“NVD”), the plurality of instructions when executed by the one or more processors of the
first NVD cause the one or more processors of the first NVD to:
- receive at least one compiled rule, wherein each of the at least one compiled
rules is applicable to a class of packets received by the first NVD for delivery to a Virtualized
30 Network Interface Card (“VNIC”);
receive a first packet for delivery to a first VNIC;

determine that a first rule of the at least one compiled rule is applicable to the first packet; and
process the first packet according to the first rule.



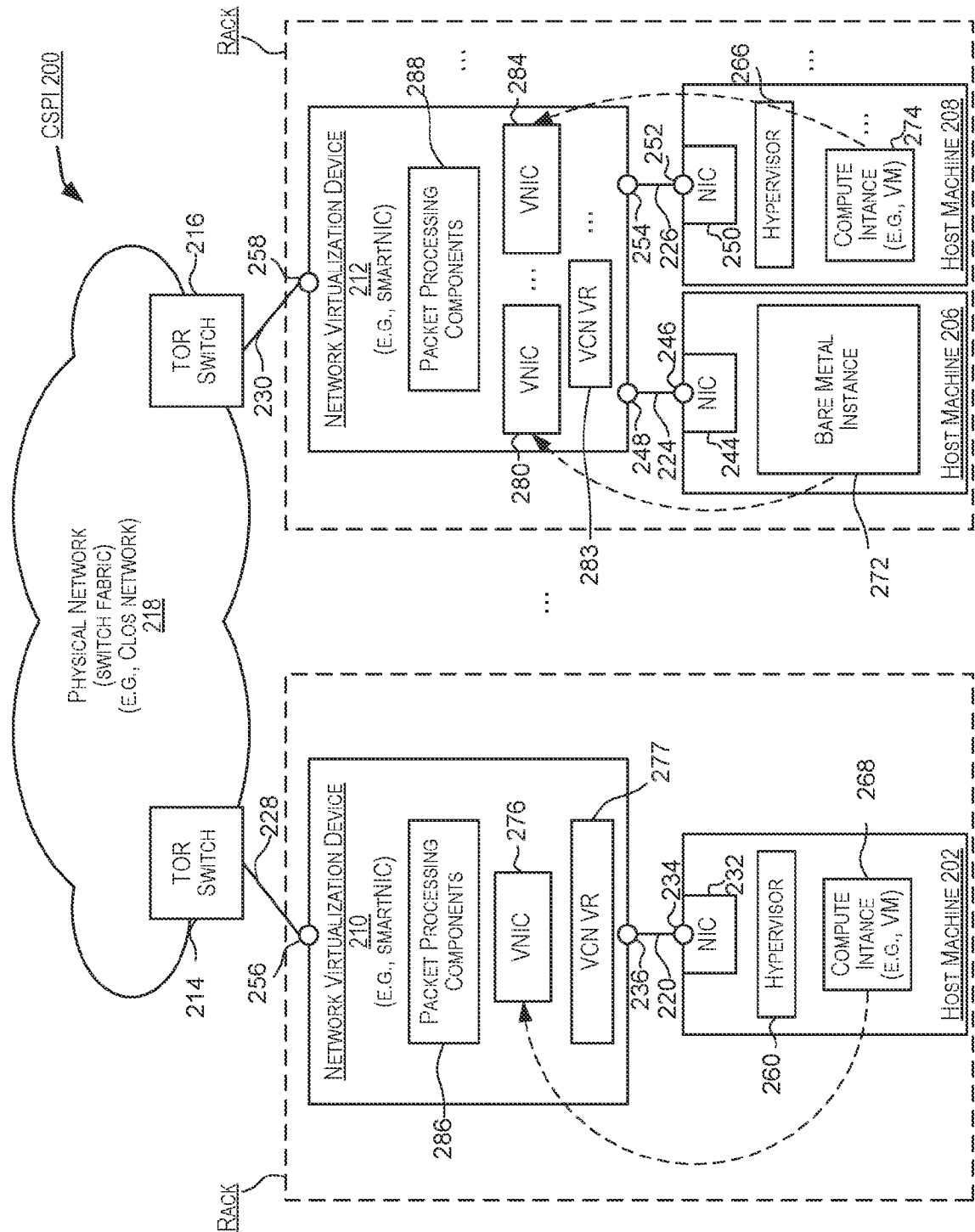


FIG. 2

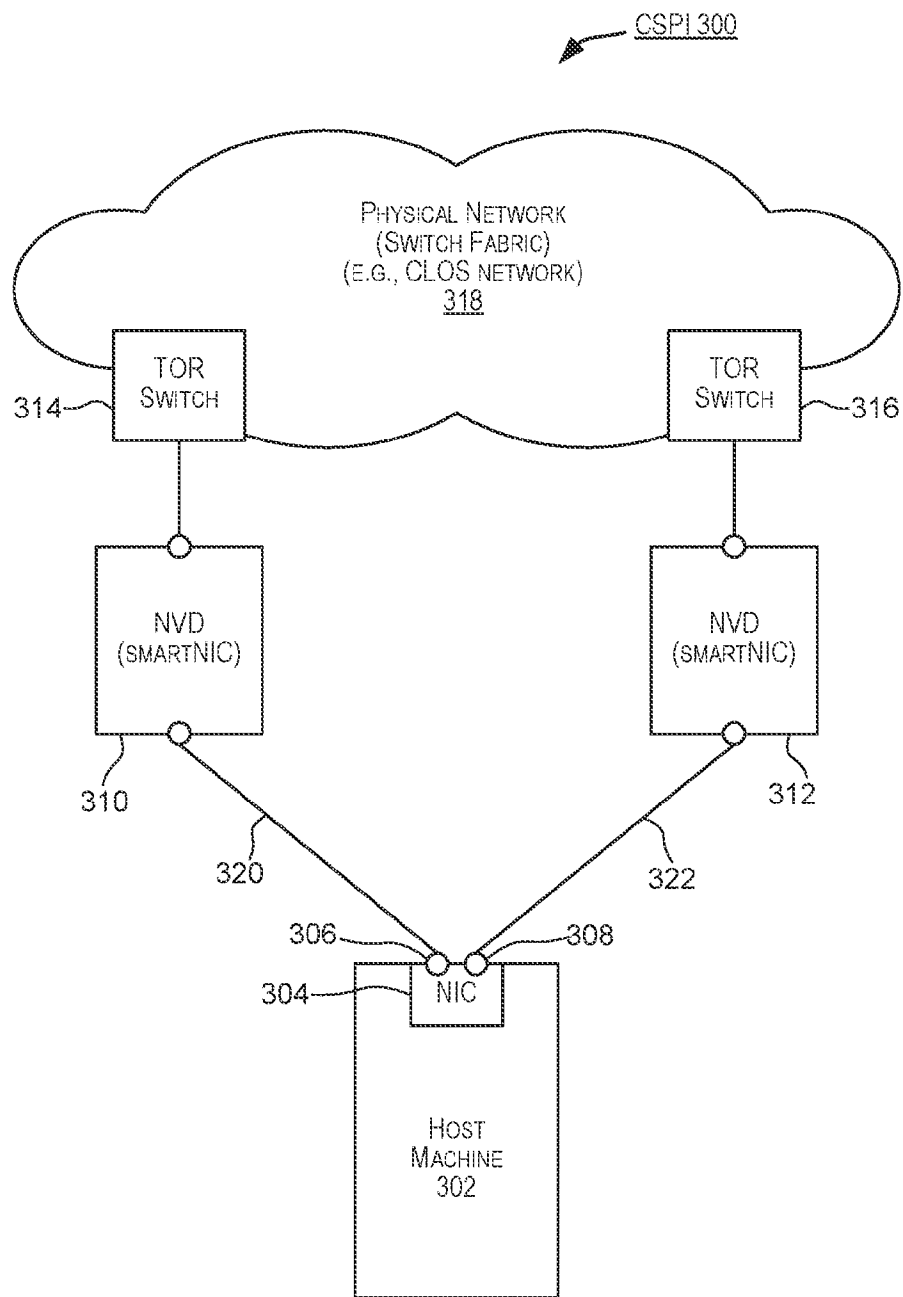


FIG. 3

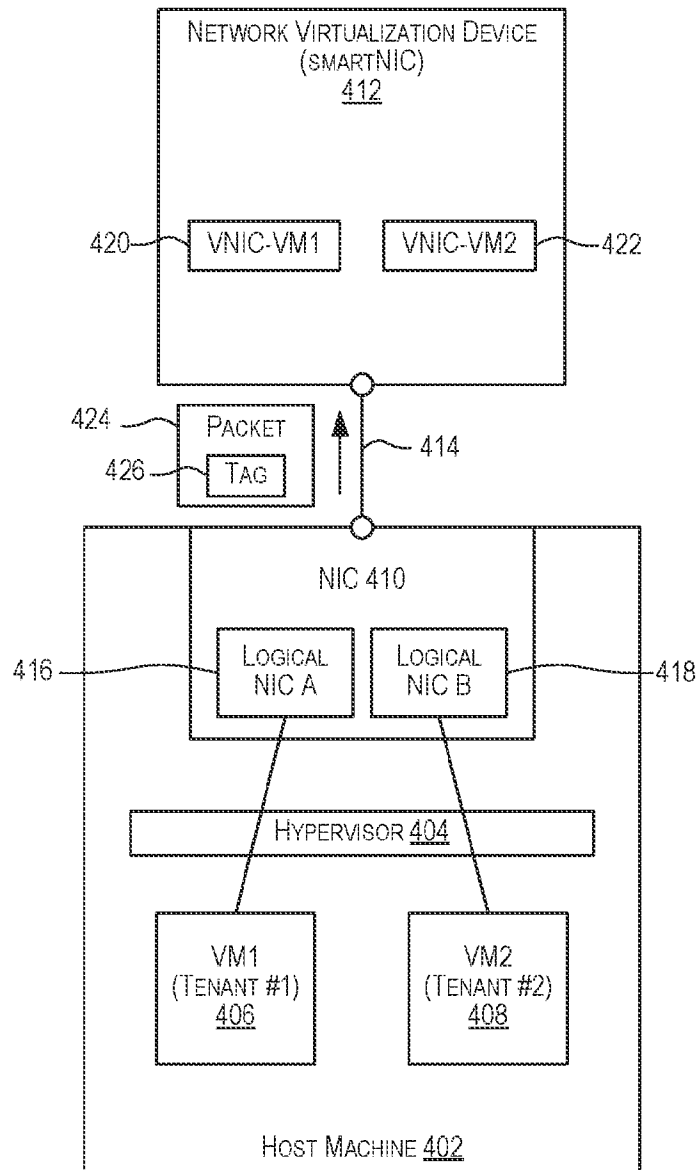


FIG. 4

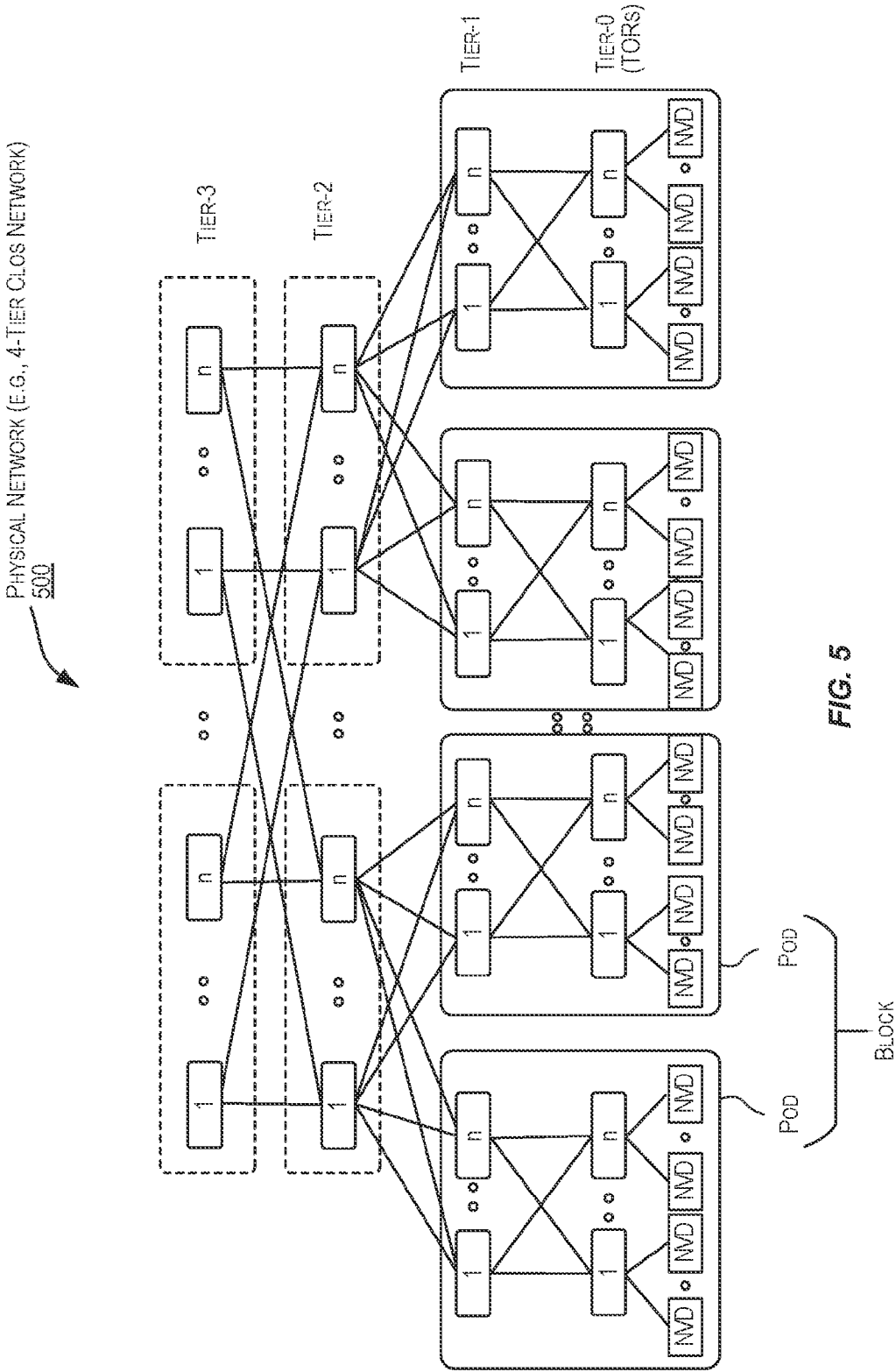


FIG. 5

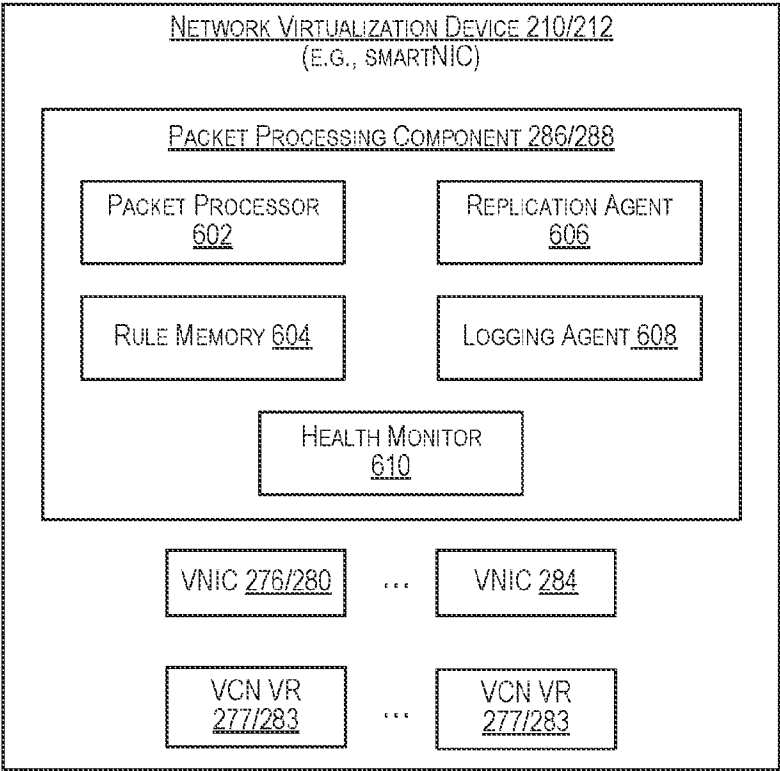


FIG. 6

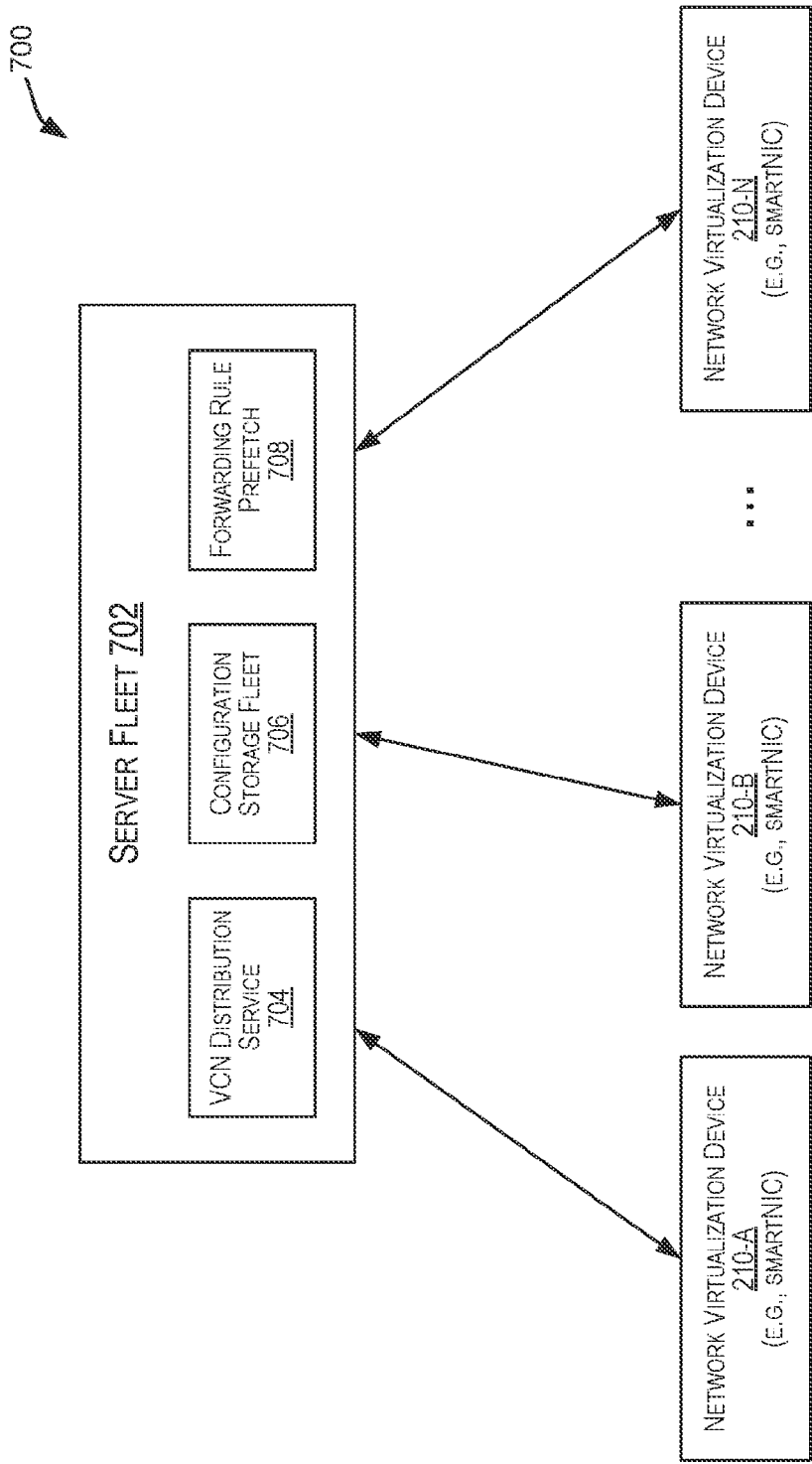
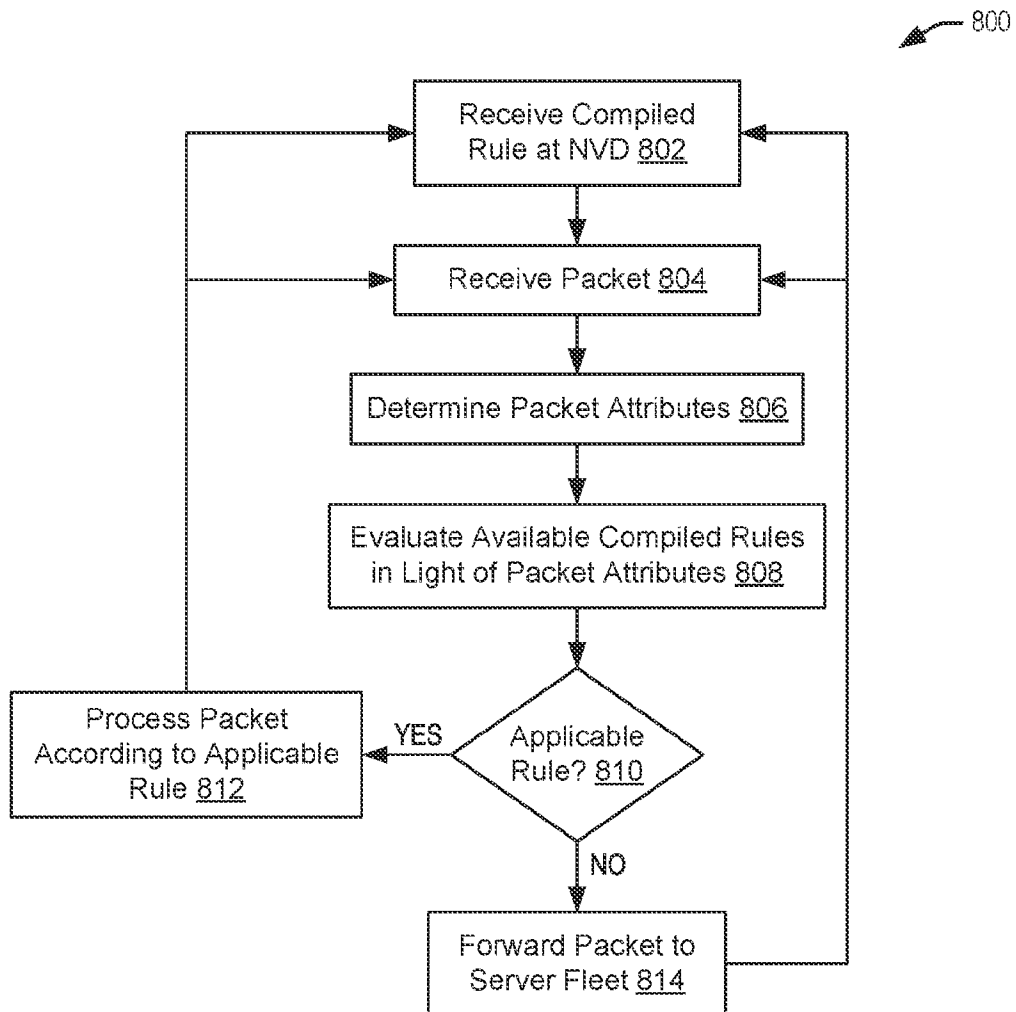
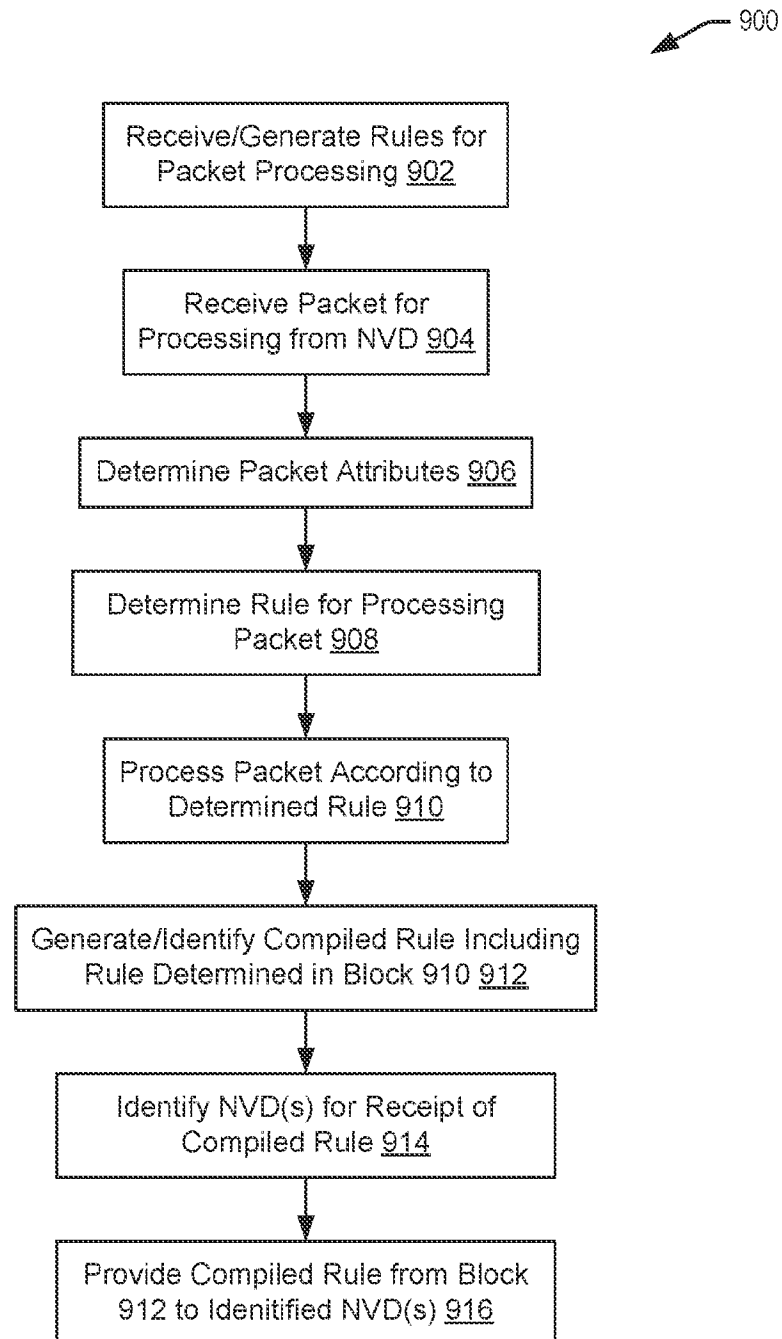


FIG. 7

**FIG. 8**

**FIG. 9**

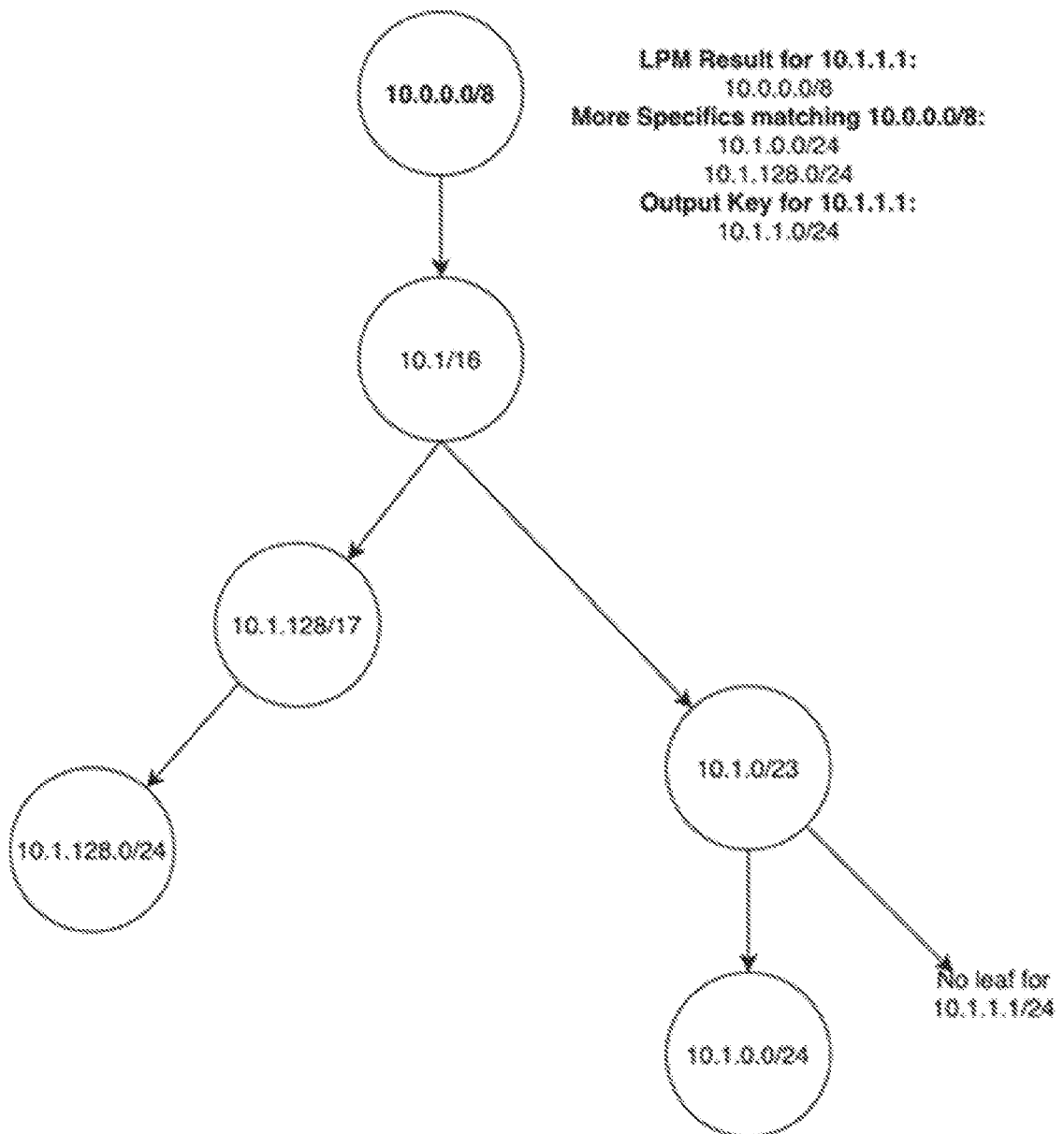


FIG. 10

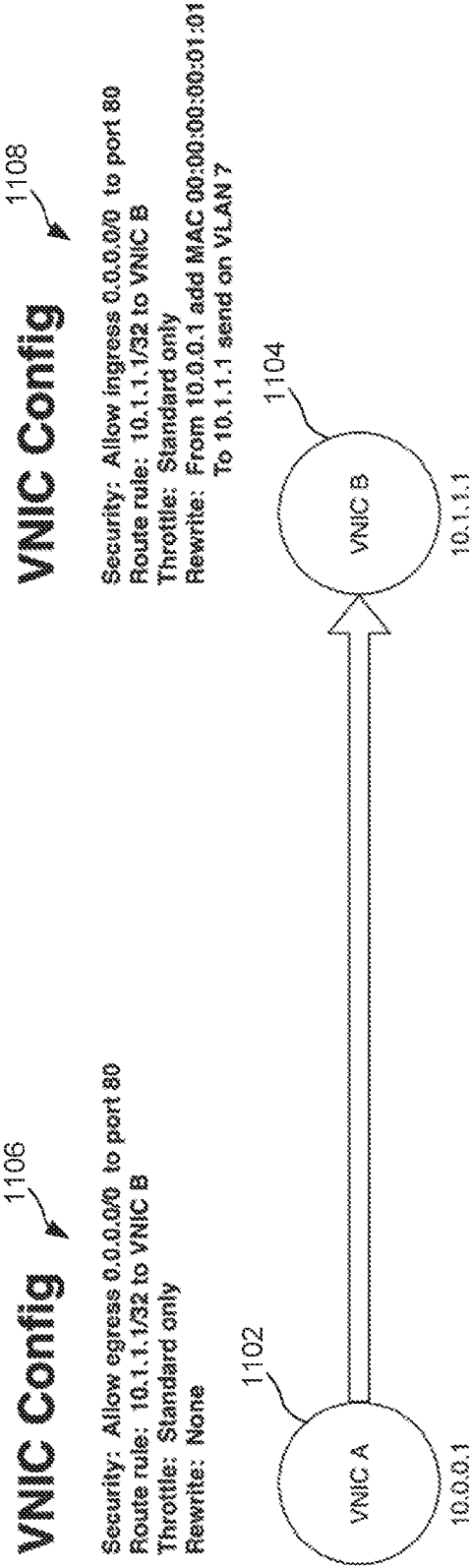
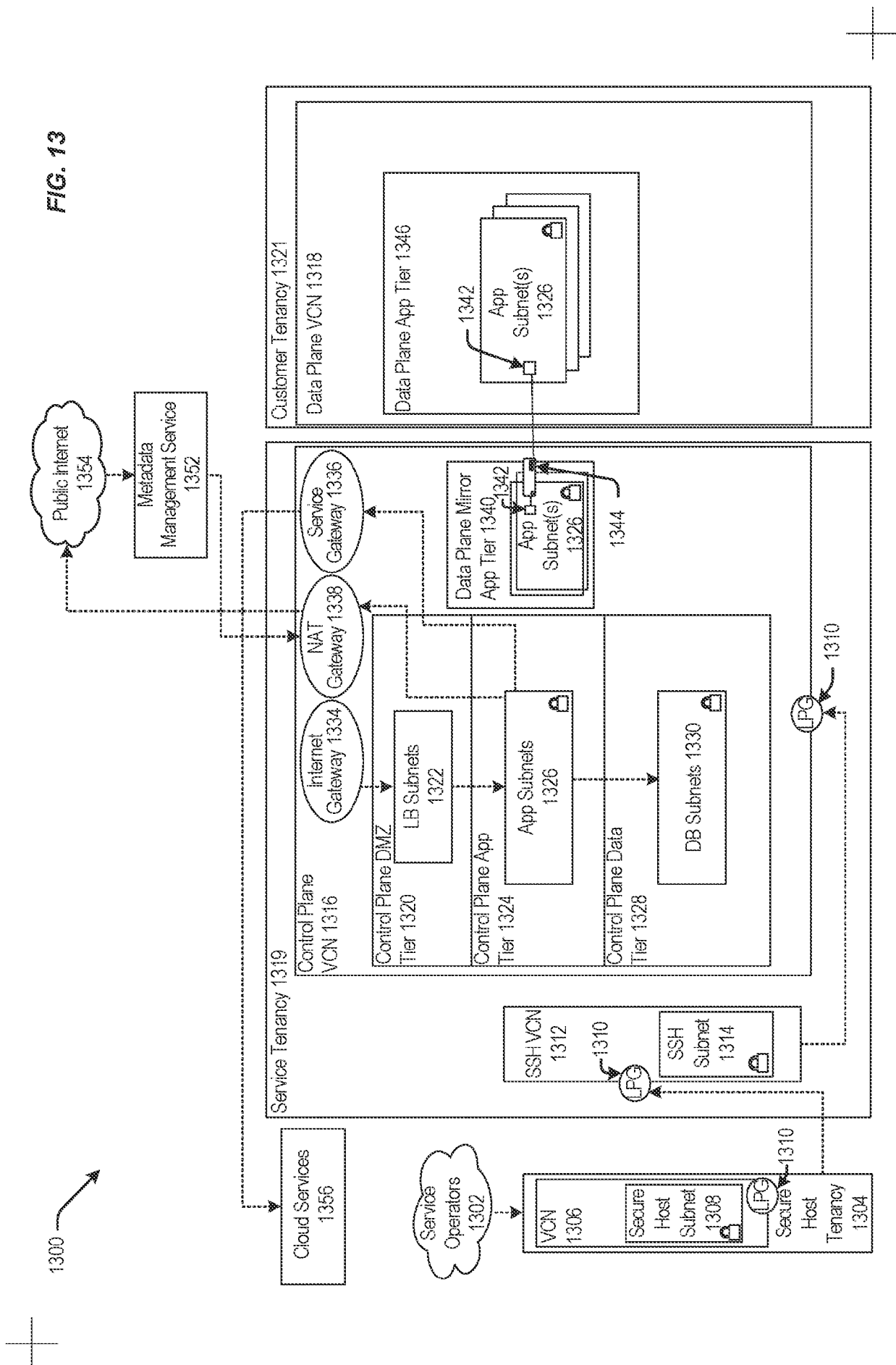
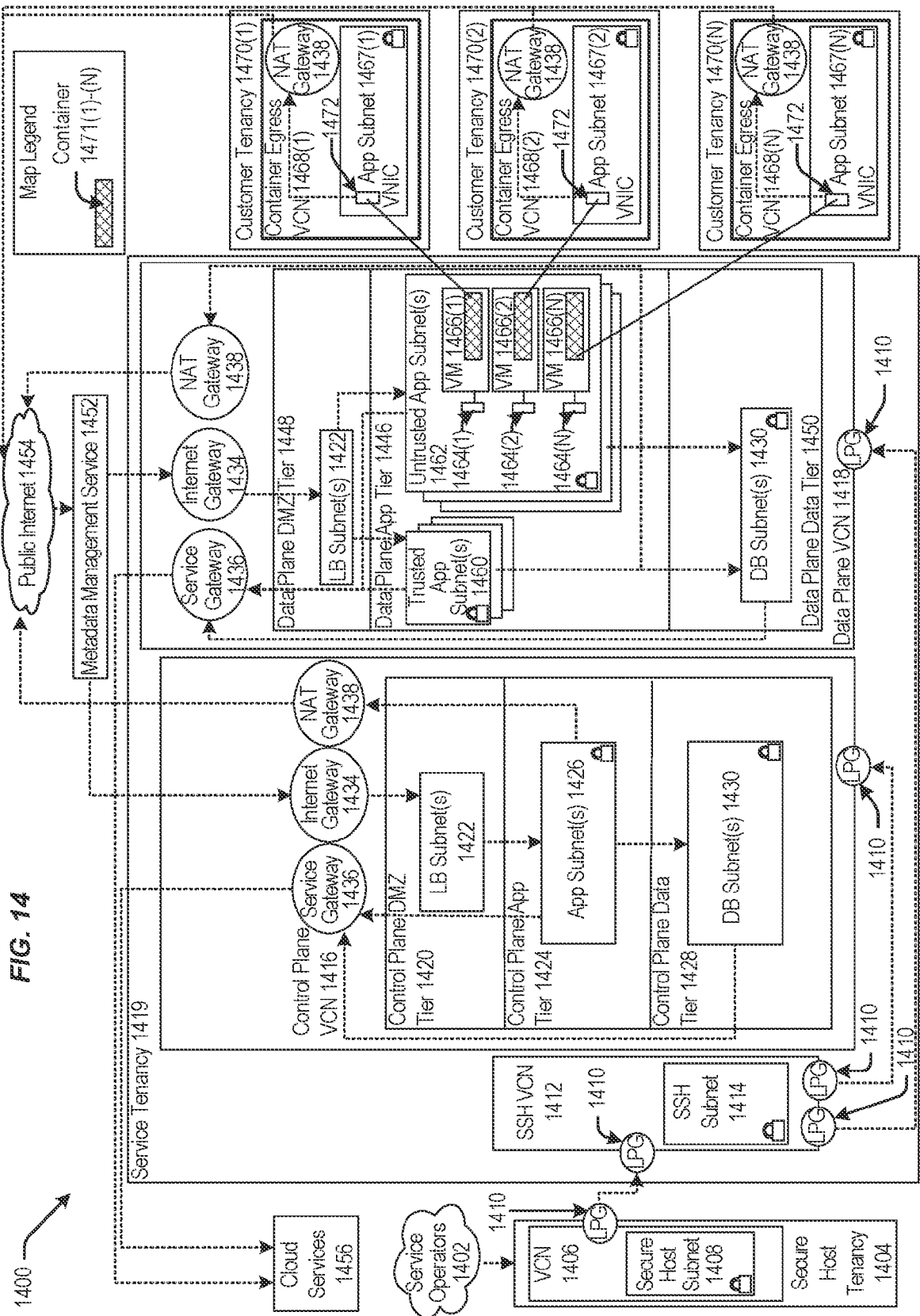


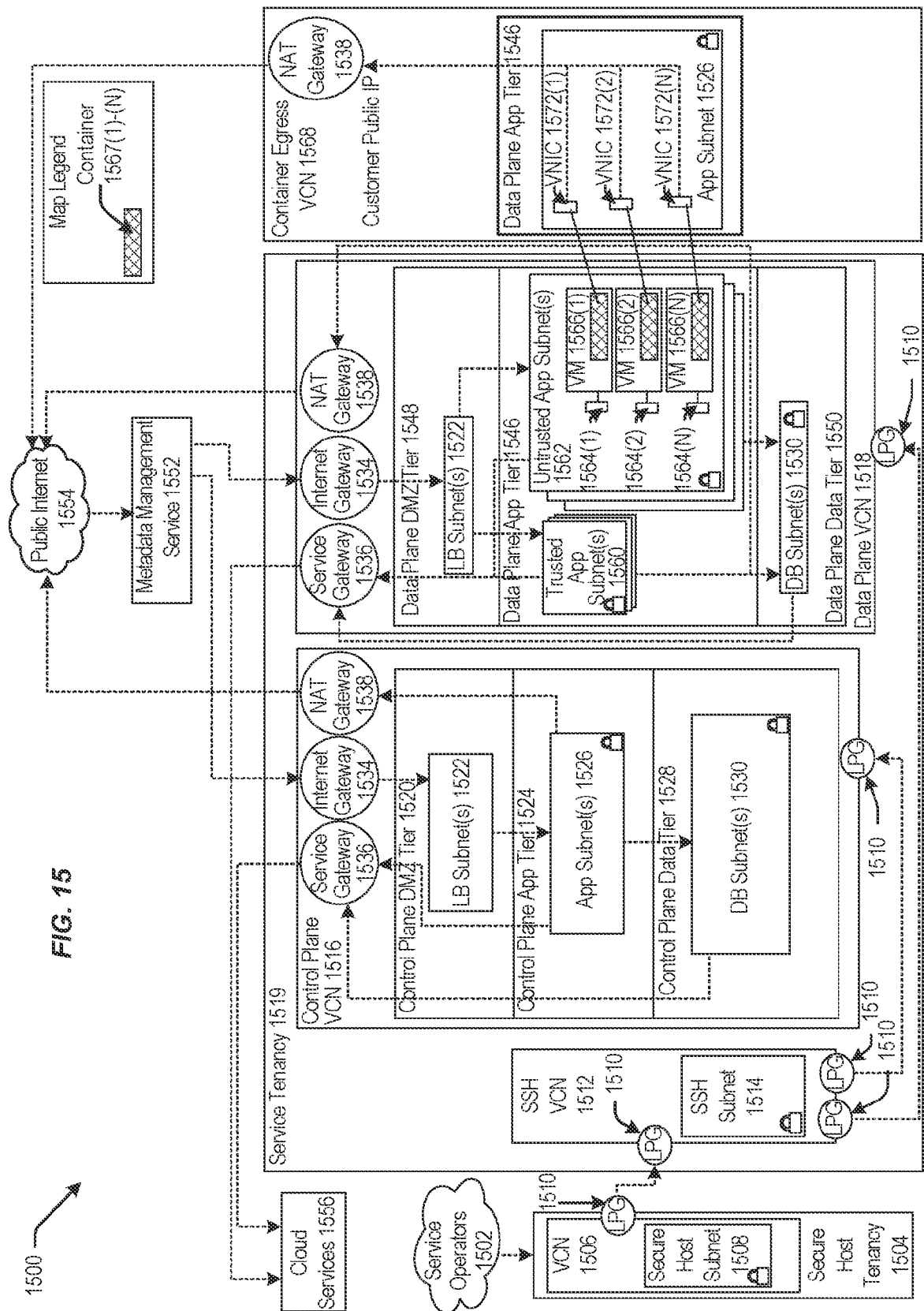
FIG. 11

FIG. 13





15
7
G.
1000
E



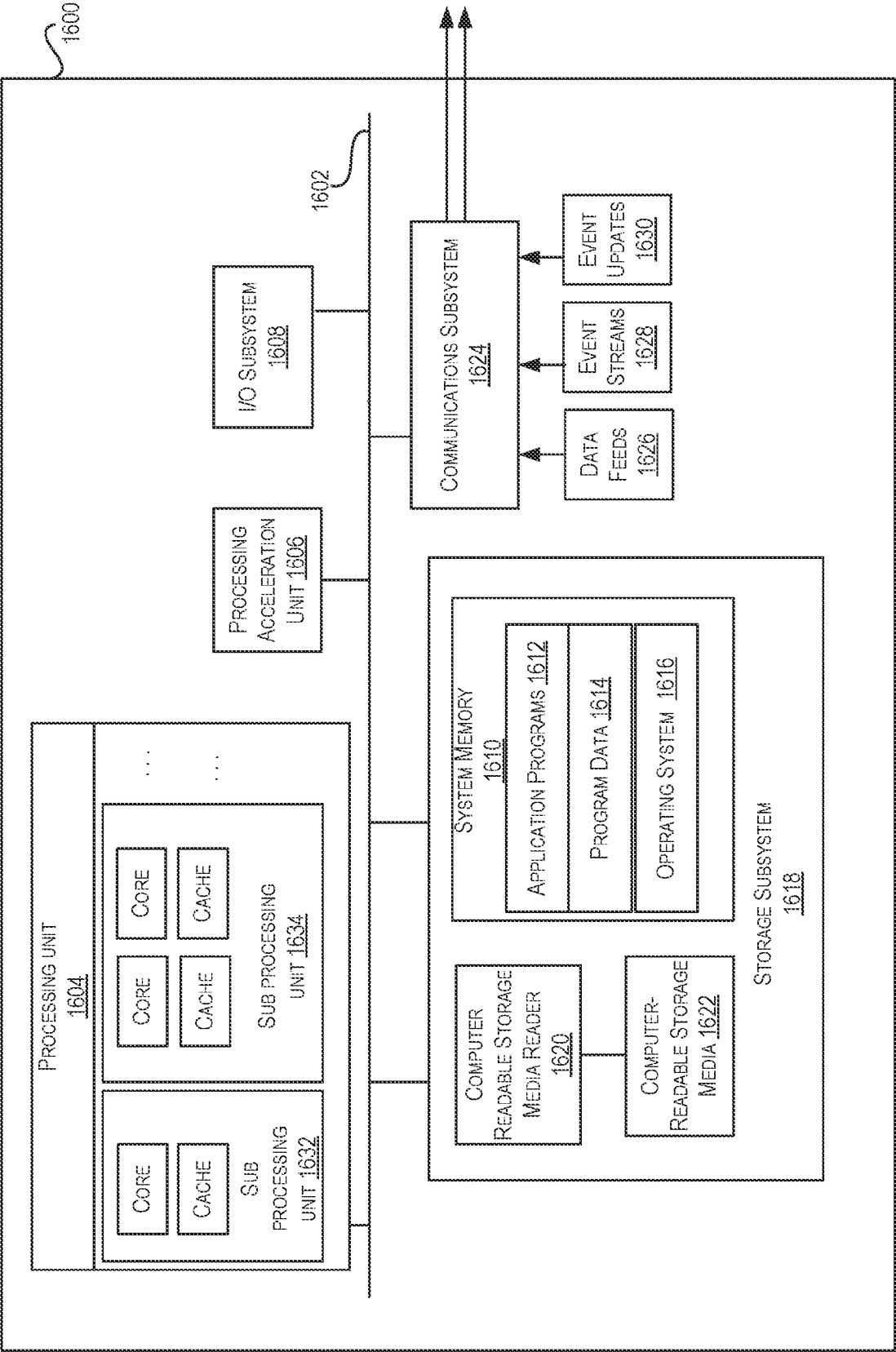


FIG. 16

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2023/024897

A. CLASSIFICATION OF SUBJECT MATTER

INV. H04L9/40 H04L41/044

ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2018/062923 A1 (KATREKAR AKSHAY [US] ET AL) 1 March 2018 (2018-03-01)	1-8, 18, 20
Y	paragraphs [0004] - [0007], [0019],	9-14, 19
A	[0105] - [0106], [0128] - [0130], [0141] - [0142], [0184] - [0187], [0260] - [0269]	15-17

Y	US 2013/151685 A1 (BURSELL MICHAEL [GB]) 13 June 2013 (2013-06-13)	9-14, 19
	paragraphs [0006], [0072], [0121] - [0123]	

A	US 2020/304413 A1 (MACCARTHAIGH COLM [US]) 24 September 2020 (2020-09-24)	1-20
	paragraphs [0033] - [0036]	



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

5 September 2023

Date of mailing of the international search report

12/09/2023

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2

NL - 2280 HV Rijswijk

Tel. (+31-70) 340-2040,

Fax: (+31-70) 340-3016

Authorized officer

Herzog, Till-Henning

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2023/024897

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2018062923 A1	01-03-2018	US 2018062923 A1	01-03-2018
		US 2018063176 A1	01-03-2018
		US 2019306185 A1	03-10-2019

US 2013151685 A1	13-06-2013	CN 104067231 A	24-09-2014
		EP 2788871 A1	15-10-2014
		US 2013151685 A1	13-06-2013
		WO 2013086204 A1	13-06-2013

US 2020304413 A1	24-09-2020	US 10749808 B1	18-08-2020
		US 2020304413 A1	24-09-2020
