(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117099082 A
(43)申请公布日 2023.11.21

(21)申请号 202280026574.9
(22)申请日 2022.09.30
(30)优先权数据
202141044924 2021.10.04 IN
63/364,283 2022.05.06 US
(85)PCT国际申请进入国家阶段日
2023.09.28
(86)PCT国际申请的申请数据
PCT/US2022/077420 2022.09.30
(87)PCT国际申请的公布数据
WO2023/060025 EN 2023.04.13
(71)申请人 瞻博网络公司
地址 美国加利福尼亚州
(72)发明人 扎基·巴赫迈特
伊克拉斯·M·奥塔马利卡
普拉萨德·梅里亚拉
(74)专利代理机构 北京康信知识产权代理有限责任公司 11240
专利代理师 刘彬
(51)Int.Cl.
G06F 9/50(2006.01)

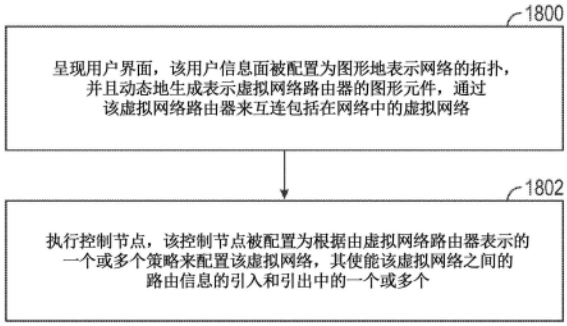
权利要求书3页 说明书57页 附图36页

(54)发明名称

用于云原生软件定义网络架构的用户界面

(57)摘要

总体上,描述了用于经由软件定义网络(SDN)架构呈现的用户界面(UI)来创建虚拟网络路由器的技术。包括存储器和处理电路的网络控制器可以执行这些技术。存储器可以存储UI,处理电路可以呈现UI并执行控制节点。UI可以图形地表示包括第一虚拟网络和第二虚拟网络的网络的拓扑。UI可以动态地生成表示虚拟网络路由器的图形元件,通过该虚拟网络路由器互连第一虚拟网络和第二虚拟网络。虚拟网络路由器可以表示使得第一虚拟网络和第二虚拟网络之间的路由信息的引入和引出中的一个或多个的一个或多个策略的逻辑抽象。控制节点根据一个或多个策略配置第一虚拟网络和第二虚拟网络。



CN 117099082 A

1.一种用于软件定义网络(SDN)架构系统的网络控制器,所述网络控制器包括:

存储器,被配置为存储用户界面;

处理电路,被配置为呈现所述用户界面,并且所述处理电路被配置为执行控制节点,其中,所述用户界面被配置为:

图形地表示由所述软件定义网络(SDN)架构系统支持的网络的拓扑,所述网络包括第一虚拟网络和第二虚拟网络;以及

动态地生成表示虚拟网络路由器的图形元件,其中,由所述虚拟网络路由器互连所述第一虚拟网络和所述第二虚拟网络,所述虚拟网络路由器表示一个或多个策略的逻辑抽象,所述一个或多个策略使得在所述第一虚拟网络和所述第二虚拟网络之间的路由信息的引入和引出中的一者或多者,

其中,所述控制节点根据所述一个或多个策略配置所述第一虚拟网络和所述第二虚拟网络,以使得能够经由所述虚拟网络路由器在所述第一虚拟网络和所述第二虚拟网络之间的路由信息的引入和引出中的一者或多者。

2.根据权利要求1所述的网络控制器,其中,所述用户界面被配置为:在图形地表示所述网络的拓扑时呈现表示所述第一虚拟网络的第一图形元件以及表示所述第二虚拟网络的第二图形元件。

3.根据权利要求1所述的网络控制器,其中,所述用户界面被配置为当图形地表示所述网络的拓扑时:

接收来自用户的指示所述虚拟网络路由器的连通性类型的输入;

选择与所述虚拟网络路由器的连通性类型相关联的图形图标;以及

动态地生成包括与所述虚拟网络路由器的连通性类型相关联的所述图形图标的所述图形元件。

4.根据权利要求3所述的网络控制器,其中,所述连通性类型包括网格连通性和中心辐条式连通性中的一者。

5.根据权利要求3所述的网络控制器,

其中,所述连通性类型包括网格连通性,并且

其中,由网格虚拟网络路由器表示的所述一个或多个策略包括对称的引入和引出策略,所述对称的引入和引出策略使得所述第一虚拟网络与所述第二虚拟网络之间的所述路由信息的引入和引出两者。

6.根据权利要求3所述的网络控制器,

其中,所述连通性类型包括中心辐条式连通性;以及

其中,由中心虚拟网络路由器表示的所述一个或多个策略包括非对称的引入和引出策略,所述非对称的引入和引出策略使得将所述路由信息从第一辐条虚拟网络和第二辐条虚拟网络两者引出到所述虚拟网络路由器,但不在所述第一辐条虚拟网络与所述第二辐条虚拟网络之间引入所述路由信息。

7.根据权利要求1所述的网络控制器,其中,所述用户界面被配置为:在图形地表示所述网络的拓扑时,接收来自用户的定义与所述第一虚拟网络相关联的第一标签以及与所述第二虚拟网络相关联的第二标签的输入,以及

其中,所述处理电路执行配置节点,所述配置节点基于所述第一标签和所述第二标签

识别所述第一虚拟网络和所述第二虚拟网络,以便根据所述一个或多个策略配置与所述虚拟网络路由器对应的路由实例,以使得在所述第一虚拟网络与所述第二虚拟网络之间引入和引出所述路由信息。

8.根据权利要求1所述的网络控制器,

其中,所述第一虚拟网络与第一命名空间相关联,

其中,所述第二虚拟网络与第二命名空间相关联,

其中,所述虚拟网络路由器为第一虚拟网络路由器,

其中,所述一个或多个策略包括第一策略和第二策略,

其中,所述第一虚拟网络路由器表示用于所述第一虚拟网络和第二虚拟网络路由器之间的所述路由信息的引入和引出的所述第一策略,以及

其中,所述第二虚拟网络路由器表示用于所述第二虚拟网络和所述第一虚拟网络路由器之间的所述路由信息的引入和引出的所述第二策略。

9.根据权利要求8所述的网络控制器,其中,所述第一虚拟网络路由器在部署所述第一策略之前必须获得将所述第一虚拟网络路由器互连到所述第二虚拟网络路由器的授权。

10.根据权利要求1所述的网络控制器,其中,所述网络控制器支持容器编排系统。

11.根据权利要求1所述的网络控制器,

其中,所述虚拟网络路由器是用于SDN架构的SDN架构配置的定制资源的实例,

其中,所述处理电路执行定制资源控制器以将所述SDN架构的状态协调至针对所述虚拟网络路由器的预期状态,以及

其中,所述控制节点被配置为部署所述虚拟网络路由器以实现针对所述虚拟网络路由器的所述预期状态。

12.根据权利要求1所述的网络控制器,

其中,使用公共路由实例和公共路由目标实现所述虚拟网络路由器,

其中,使用第一路由实例和第一路由目标实现所述第一虚拟网络,

其中,使用第二路由实例和第二路由目标实现所述第二虚拟网络,以及

其中,所述控制节点被配置为利用所述第一路由目标、所述第二路由目标和所述公共路由目标对所述第一路由实例、所述第二路由实例和所述公共路由实例的引入和引出进行配置,以实现网格连通性。

13.根据权利要求1所述的网络控制器,

其中,使用第一多个计算节点的一个或多个第一虚拟路由器实现所述第一虚拟网络,以及

其中,使用第二多个计算节点的一个或多个第二虚拟路由器实现所述第二虚拟网络。

14.根据权利要求1所述的网络控制器,

其中,所述第一多个计算节点是第一Kubernetes集群,以及其中,所述第二多个计算节点是第二Kubernetes集群。

15.根据权利要求1所述的网络控制器,其中,所述控制节点将所述路由信息解析成针对所述第一虚拟网络和所述第二虚拟网络中的一者或多者的转发信息,并将所述转发信息安装在支持所述第一虚拟网络与所述第二虚拟网络之间的互连性的虚拟路由器中。

16.一种方法,包括:

由用于软件定义网络(SDN)架构系统的网络控制器呈现用户界面,

其中,所述用户界面被配置为:

图形地表示由所述软件定义网络(SDN)架构系统支持的网络的拓扑,所述网络包括第一虚拟网络和第二虚拟网络;以及

动态地生成表示虚拟网络路由器的图形元件,其中,由所述虚拟网络路由器互连所述第一虚拟网络和所述第二虚拟网络,所述虚拟网络路由器表示一个或多个策略的逻辑抽象,所述一个或多个策略使得在所述第一虚拟网络和所述第二虚拟网络之间的路由信息的引入和引出中的一者或多者;以及

由所述网络控制器执行控制节点,其中,所述控制节点根据所述一个或多个策略配置所述第一虚拟网络和所述第二虚拟网络,以使得能够经由所述虚拟网络路由器在所述第一虚拟网络和所述第二虚拟网络之间的路由信息的引入和引出中的一者或多者。

17.根据权利要求16所述的方法,其中,图形地表示所述网络的拓扑包括:呈现表示所述第一虚拟网络的第一图形元件以及表示所述第二虚拟网络的第二图形元件。

18.根据权利要求16所述的方法,其中,图形地表示所述网络的拓扑包括:

接收来自用户的指示所述虚拟网络路由器的连通性类型的输入;

选择与所述虚拟网络路由器的连通性类型相关联的图形图标;以及

动态地生成包括与所述虚拟网络路由器的连通性类型相关联的所述图形图标的所述图形元件。

19.根据权利要求18所述的方法,其中,所述连通性类型包括网格连通性和中心辐条式连通性中的一者。

20.一种非暂时性计算机可读介质,包括用于使用于软件定义网络(SDN)架构系统的网络控制器的处理电路执行以下操作的指令:

呈现用户界面,

其中,所述用户界面被配置为:

图形地表示由所述软件定义网络(SDN)架构系统支持的网络的拓扑,所述网络包括第一虚拟网络和第二虚拟网络;以及

动态地生成表示虚拟网络路由器的图形元件,其中,由所述虚拟网络路由器互连所述第一虚拟网络和所述第二虚拟网络,所述虚拟网络路由器表示一个或多个策略的逻辑抽象,所述一个或多个策略使得在所述第一虚拟网络和所述第二虚拟网络之间的路由信息的引入和引出中的一者或多者;以及

执行控制节点,其中,所述控制节点根据所述一个或多个策略配置所述第一虚拟网络和所述第二虚拟网络,以使得能够经由所述虚拟网络路由器在所述第一虚拟网络和所述第二虚拟网络之间的路由信息的引入和引出中的一者或多者。

用于云原生软件定义网络架构的用户界面

[0001] 相关申请

[0002] 本申请要求于2022年5月6日提交的美国临时申请第63/364,283号和于2021年10月4日提交的印度临时申请第202141044924号的权益,它们中的每一个通过引用整体并入本文。

技术领域

[0003] 本公开涉及虚拟化计算基础设施,并且更具体地,涉及云原生联网。

背景技术

[0004] 在典型的云数据中心环境中,存在提供计算和/或存储容量以运行各种应用的互连服务器的大集合。例如,数据中心可以包括为订户(即,数据中心的客户)托管应用和服务的设施。数据中心可以例如托管所有基础设施装备,诸如联网和存储系统、冗余电源和环境控制。在典型的数据中心中,存储系统和应用服务器的集群经由一层或多层物理网络交换机和路由器提供的高速交换结构互连。更复杂的数据中心利用位于各种物理主机设施中的订户支持装备提供遍布全世界的基础设施。

[0005] 虚拟化数据中心正在成为现代信息技术(IT)基础设施的核心基础。特别地,现代数据中心已经广泛地利用了虚拟化环境,在虚拟化环境中,虚拟主机(在本文也被称为虚拟执行元件,诸如虚拟机或容器)被部署在物理计算设备的底层计算平台上并在其上执行。

[0006] 数据中心或包括一个或多个服务器的任何环境内的虚拟化可以提供若干优点。一个优点是虚拟化可以提供对效率的显著改进。随着每个物理CPU具有大量核的多核微处理器架构的出现,底层物理计算设备(即,服务器)已经变得越来越强大,虚拟化变得更容易和更有效。第二个优点是虚拟化提供了对计算基础设施的显著控制。随着物理计算资源变成可替代资源,诸如在基于云的计算环境中,计算基础设施的供应和管理变得更容易。因此,企业IT人员通常更喜欢数据中心中的虚拟化的计算集群,因为除了虚拟化提供的效率和增加的投资回报(ROI)之外,它们的管理具备优势。

[0007] 容器化是一种基于操作系统级虚拟化的虚拟化方案。容器是用于彼此隔离并且与主机隔离的应用的重量轻且便携的执行元件。因为容器没有紧密耦接到主机硬件计算环境,所以应用可以被绑定到容器图像并且作为单个轻量包在支持底层容器架构的任何主机或虚拟主机上执行。这样,容器解决了如何使软件在不同的计算环境中工作的问题。容器提供了从一个计算环境到另一个虚拟或物理环境一致运行的承诺。

[0008] 利用容器的固有轻量级性质,单个主机通常可以支持比传统虚拟机(VM)更多的容器实例。通常短期的(与大多数VM相比),可以比VM更有效地创建和移动短期容器,并且还可以将它们作为逻辑相关的元件组来管理(例如,对于一些编排平台,有时称为“pod”,例如Kubernetes)。这些容器特性影响对于容器联网解决方案的要求:网络应当是敏捷的和可扩展的。VM、容器和裸金属服务器可能需要在同一计算环境中共存,其中在应用的多样部署之间启用通信。容器网络还应当是不可知的,以与用于部署容器化的应用的多种类型的编排

平台一起工作。

[0009] 管理部署和应用执行的基础设施的计算基础设施可以涉及两个主要角色:(1)编排—用于跨主机集群自动化应用的部署、缩放和操作,并且提供计算基础设施,其可以包括以容器为中心的计算基础设施;和(2)网络管理-用于在网络基础设施中创建虚拟网络以使得能够在诸如容器或VM的虚拟执行环境上运行的应用中以及在传统(例如,物理)环境上运行的应用中进行封包通信。软件定义网络有助于网络管理。

发明内容

[0010] 总体上,描述了用于提供用于与云原生软件定义网络(SDN)架构进行接口的用户界面(例如,图形用户界面)的技术。在一些示例中,SDN架构可以包括在计算节点和诸如路由器或交换机等网络设备中实现的数据平面元件,并且SDN架构还可以包括用于创建和管理虚拟网络的网络控制器。SDN架构配置和控制平面被设计为具有支持服务中升级的基于容器的微服务架构的横向扩展云原生软件。可以实现用于配置平面的配置节点以展露定制资源。用于SDN架构配置的这些定制资源可以包括传统上由网络控制器展露的配置元件,但是配置元件可以与Kubernetes本地/内置资源合并以支持统一意图模型,由聚合的API层展露,其由Kubernetes控制器和工作以协调SDN架构的实际状态与意图状态的定制资源控制器实现。

[0011] 用于SDN架构配置的定制资源包括虚拟网络路由器,该虚拟网络路由器表示用于在由SDN架构实现的虚拟网络之间互连和实现通信的逻辑抽象。虚拟网络路由器定制资源的实例使用在相应虚拟网络的路由实例内配置的引入和引出路由目标来促进路由信息的交换(指的是非对称或对称的引入和引出)。于是,实际上,虚拟网络路由器使得能够在针对SDN架构定义的虚拟网络资源之间实现路由泄漏。因为虚拟网络路由器资源隐藏了用于实现各种虚拟网络资源的VRF中的虚拟路由和转发实例(VRF或“路由实例(routing instance)”)配置的复杂详细信息,所以该技术提供了一种直观的基于意图的模型,用于定义虚拟网络资源和SDN架构中配置的相应虚拟网络之间的互连性,并且甚至对于可能不熟悉边界网关协议和VRF配置的那些人也提高了可用性。

[0012] 用于对控制路由信息的引入和/或引出的策略进行抽象的VNR的添加可简化将网络元件(诸如虚拟网络)拼接在一起以促进此类网络元件之间的互连。本公开所述的技术的各个方面可提供被配置为图形地呈现可包括各种网络元件(诸如虚拟网络)的网络的拓扑的用户界面,而不是求助于复杂的接口来定义涉及可能需要复杂的网络构造知识(诸如路由目标、标识网络元件的标签等)的复杂陈述的VNR。用户可以与用户界面(其可以指图形用户界面(GUI))进行接口,以可视地配置VNR以互连两个或更多个网络元件。

[0013] 用户界面可以接收用户输入(其可以被称为经由GUI输入的输入),诸如用于拖放或以其他方式定义VNR的图形用户界面输入。用户界面可以将这些输入作为配置VNR的请求传递到网络控制器。用户可以通过与GUI的交互来输入,以图形地定义VNR,选择VNR针对其建立互连性(例如,网格连通性和/或中心辐条式连通性)的虚拟网络和其他网络元件。网络控制器可以处理这些输入并更新GUI以呈现提示和其他GUI,通过这些提示和GUI来提示用户定义VNR可能需要的附加输入。网络控制器可将定义VNR的这些输入减少到控制网络元件之间的路由信息的引入和/或引出的各种策略中,并且在适当的网络元件中配置策略以促

进实现网络元件之间的互连性的路由信息的交换。

[0014] 这些技术可以提供附加的一个或多个技术优点。例如,云原生(cloud-native)SDN架构可以解决传统SDN架构中的限制,该限制涉及生命周期管理的复杂性、强制性高资源分析部件、配置管理中的规模限制以及缺少基于命令行界面(CLI)的接口。例如,用于SDN架构的网络控制器是具有简化的安装覆盖区的云原生、轻量级分布式应用。这还便于配置和控制平面的配置节点和控制节点的各种部件微服务的更容易和模块化的升级。

[0015] 该技术还可以实现可选的云原生监视(遥测)和用户界面、使用连接到启用DPDK的pod的基于数据平面转发工具(基于DPDK)的虚拟路由器的用于容器的高性能数据平面,以及在一些情况下利用用于现有编排平台(诸如Kubernetes或Openstack)的配置框架的云原生配置管理。作为云原生架构,网络控制器是可扩展的并且弹性的,以寻址和支持多个集群。在一些情况下,网络控制器还可以支持关键性能指标(KPI)的可扩展性和性能要求。

[0016] 另外,根据本公开所述的技术的各个方面配置的用户界面可以使没有经验的用户(在SDN架构的上下文中)能够在不具有网络构造的广泛知识(诸如以上列出的那些)的情况下定义VNR。这样,用户可以更有效地与用户界面交互以定义VNR来满足企业和/或客户目标。用户界面可以引导用户以减少并可能消除配置VNR中的错误,同时还使得用户能够查看网络的拓扑并获得对网络本身的更好的理解。通过易于使用以及消除错误,用户界面可以减少用户定义复杂配置数据的需要,同时还避免可能使得网络控制器的低效操作(在计算资源方面,诸如处理周期、存储器、存储器总线带宽等,以及相关联的功率)的此类错误。这样,用户界面可以改进网络控制器本身的操作。

[0017] 在一个示例中,本技术的各个方面涉及一种用于软件定义网络(SDN)架构系统的网络控制器,该网络控制器包括:存储器,其被配置为存储用户界面;处理电路,其被配置为呈现该用户界面并执行控制节点,其中该用户界面被配置为:图形地表示由该软件定义网络(SDN)架构系统支持的网络的拓扑,该网络包括第一虚拟网络和第二虚拟网络;以及动态地生成表示虚拟网络路由器的图形元件,通过该虚拟网络路由器来互连该第一虚拟网络和该第二虚拟网络,该虚拟网络路由器表示一个或多个策略的逻辑抽象,该一个或多个策略使得在该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个,其中,该控制节点根据该一个或多个策略来配置该第一虚拟网络和该第二虚拟网络,以实现经由该虚拟网络路由器在该第一虚拟网络和该第二虚拟网络之间进行路由信息的引入和引出中的一个或多个。

[0018] 在另一个示例中,本技术的各个方面涉及一种方法,该方法包括:由用于软件定义网络(SDN)架构系统的网络控制器呈现用户界面,其中该用户界面被配置为:图形地表示由该软件定义网络(SDN)架构系统支持的网络的拓扑,该网络包括第一虚拟网络和第二虚拟网络;以及动态地生成表示虚拟网络路由器的图形元件,通过该虚拟网络路由器来互连该第一虚拟网络和该第二虚拟网络,该虚拟网络路由器表示一个或多个策略的逻辑抽象,该一个或多个策略使得该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个;以及由网络控制器执行控制节点,该控制节点根据一个或多个策略配置第一虚拟网络和第二虚拟网络,以实现经由虚拟网络路由器在第一虚拟网络和第二虚拟网络之间引入和引出路由信息中的一个或多个。

[0019] 在另一个示例中,本技术的各方面涉及一种非暂时性计算机可读介质,该非暂时

性计算机可读介质包括用于使软件定义网络(SDN)架构系统的网络控制器的处理电路执行以下操作的指令:呈现用户界面,其中该用户界面被配置为:图形地表示由该软件定义网络(SDN)架构系统支持的网络的拓扑,该网络包括第一虚拟网络和第二虚拟网络;以及动态地生成表示虚拟网络路由器的图形元件,通过该虚拟网络路由器来互连该第一虚拟网络和该第二虚拟网络,该虚拟网络路由器表示一个或多个策略的逻辑抽象,该一个或多个策略使得在该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个;以及执行控制节点,该控制节点根据该一个或多个策略来配置该第一虚拟网络和该第二虚拟网络,以实现经由该虚拟网络路由器在该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个。

[0020] 在附图和以下描述中阐述本公开的一个或多个示例的详细信息。从说明书和附图以及从权利要求书中,其他特征、目的和优点将是显而易见的。

附图说明

[0021] 图1是图示了其中可实现本文所述的技术的示例的示例性计算基础设施的框图。

[0022] 图2是图示了根据本公开的技术的用于云原生联网的云原生SDN架构的示例的框图。

[0023] 图3是根据本公开的技术,更详细地图示SDN架构的部件的另一视图的框图。

[0024] 图4是图示了根据本公开的技术的SDN架构的示例性部件的框图。

[0025] 图5是根据本公开所述的技术的示例性计算设备的框图。

[0026] 图6是根据本公开的技术的操作为SDN架构系统的一个或多个集群的计算节点的示例性计算设备的框图。

[0027] 图7A是图示了根据本公开的技术的用于使用SDN架构的基础网络和覆盖网络配置的控制/路由平面的框图。

[0028] 图7B是图示了根据本公开的技术的使用底层网络中配置的隧道来连接pod的经配置的虚拟网络的框图。

[0029] 图8是图示了根据本公开的技术的用于SDN架构配置的定制资源的定制控制器的实例的框图。

[0030] 图9是图示了在对不同的定制资源类型具有依赖性的定制资源类型之间的创建、观察和协调的示例性流程的框图。

[0031] 图10A至10M是图示了提供网络的拓扑的图形表示的图形用户界面的示例的示图,其中虚拟网络路由器可被图形地定义以允许不同网络元件之间的连通性。

[0032] 图11至17是图示了根据本公开所述的技术的各方面的虚拟网络路由器可被配置为实现虚拟网络之间的互连性的各种实例的示图。

[0033] 图18A至18D是图示了根据本公开所述的技术的各方面在SDN架构内建立一个或多个虚拟网络路由器时所执行的操作的流程图。

[0034] 图19是图示了图1的示例中所示的计算机架构在执行本文所述的技术的各个方面时的操作的另一流程图。

[0035] 在整个说明书和附图中,相同的附图标记表示相同的元件。

具体实施方式

[0036] 图1是图示了其中可实现本文所述的技术的示例的示例性计算基础设施8的框图。由于例如生命周期管理的复杂性、强制性高资源分析部件、配置模块中的规模限制以及没有基于命令行界面(CLI)(类Kubectl)的接口,用于虚拟网络的软件定义网络(SDN)架构的当前实现对云原生采用提出了挑战。计算基础设施8包括本文所述的云原生SDN架构系统,其解决了这些挑战并为Telco云原生时代(telco cloud-native era)现代化。云原生SDN架构的示例用例包括5G移动网络以及云和企业云原生用例。SDN架构可以包括在计算节点(例如,服务器12)和诸如路由器或交换机的网络设备中实现的数据平面元件,并且SDN架构还可以包括用于创建和管理虚拟网络的SDN控制器(例如,网络控制器24)。SDN架构配置和控制平面被设计为具有支持服务中升级的基于容器的微服务架构的横向扩展云原生软件。

[0037] 结果,SDN架构部件是微服务,并且与现有网络控制器相比,SDN架构假设基本容器编排平台来管理SDN架构部件的生命周期。利用容器编排平台搭建SDN架构部件;SDN架构使用云原生监视工具,该云原生监视工具可以与客户提供的云原生选项集成;SDN架构使用用于SDN架构对象(即,定制资源)的聚合的API来提供资源的声明性方式。SDN架构升级可以遵循云原生模式,并且SDN架构可以利用Kubernetes构造,诸如Multus、认证与授权、集群API、KubeFederation、KubeVirt和Kata容器。SDN架构可以支持数据平面转发工具(DPDK)pod,并且SDN架构可以扩展为支持具有虚拟网络策略和全局安全策略的Kubernetes。

[0038] 对于服务提供商和企业,SDN架构自动化网络资源供应和编排以动态创建高度可扩展的虚拟网络,并且链接虚拟化网络功能(VNF)和物理网络功能(PNF)以按需形成差异化服务链。SDN架构可以与诸如Kubernetes、OpenShift、Mesos、OpenStack、VMware vSphere的编排平台(例如,编排器23)以及与服务提供商运营支持系统/业务支持系统(OSS/BSS)集成。

[0039] 通常,一个或多个数据中心10为客户站11(图示为“客户11”)的应用和服务提供操作环境,该客户站具有通过服务提供商网络7耦接到数据中心的一个或多个客户网络。数据中心10中的每一个可以例如托管基础设施装备,诸如联网和存储系统、冗余电源和环境控制。服务提供商网络7耦接到公共网络15,其可以表示由其他提供商管理的一个或多个网络,并且因此可以形成大规模公共网络基础设施的一部分,诸如互联网。公共网络15可以表示例如局域网(LAN)、广域网(WAN)、互联网、虚拟LAN(VLAN)、企业LAN、层3虚拟专用网(VPN)、由运行服务提供商网络7的服务提供商操作的互联网协议(IP)内联网、企业IP网络或其某种组合。

[0040] 虽然客户站11和公共网络15主要被图示和描述为服务提供商网络7的边缘网络,但是在一些示例中,客户站11和公共网络15中的一个或多个可以是数据中心10中的任何数据中心内的租户网络。例如,数据中心10可以托管多个租户(客户),每个租户与一个或多个虚拟专用网络(VPN)相关联,每个VPN可以实现客户站11中的一个。

[0041] 服务提供商网络7提供到附接的客户站11、数据中心10和公共网络15的基于数据包的连通性。服务提供商网络7可以表示由服务提供商拥有和运营以互连多个网络的网络。服务提供商网络7可以实现多协议标签交换(MPLS)转发,并且在此类实例中可以被称为MPLS网络或MPLS骨干。在一些实例中,服务提供商网络7表示提供来自一个或多个服务提供商的服务的多个互连的自治系统,诸如互联网。

[0042] 在一些示例中,数据中心10中的每一个可以表示地理上分布的许多网络数据中心中的一个,其可以经由服务提供商网络7、专用网络链路、暗光纤或其他连接而彼此连接。如图1的示例所示,数据中心10可以包括为客户提供网络服务的设施。服务提供商的客户可以是集体实体,诸如企业和政府或个人。例如,网络数据中心可以为若干企业和终端用户托管web服务。其他示例性服务可包括数据存储、虚拟专用网络、流量工程、文件服务、数据挖掘、科学或超级计算等。虽然被图示为服务提供商网络7的单独边缘网络,但是数据中心10的元件,诸如一个或多个物理网络功能(PNF)或虚拟化网络功能(VNF)可以被包括在服务提供商网络7核心内。

[0043] 在该示例中,数据中心10包括经由由一层或多层物理网络交换机和路由器提供的交换机结构14互连的存储和/或计算服务器(或“节点”),其中服务器12A至12X(本文为“服务器12”)被描绘为耦接到架顶交换机16A至16N。服务器12是计算设备,并且在本文中还可以被称为“计算节点”、“主机”或“主机设备”。虽然在图1中仅详细示出了耦接到TOR交换机16A的服务器12A,但是数据中心10可以包括耦接到数据中心10的其他TOR交换机16的许多附加服务器。

[0044] 所示示例中的交换结构14包括互连的架顶(TOR)(或其他“叶”)交换机16A至16N(统称为“TOR交换机16”),其耦接到机箱(或“主干”或“核心”)交换机18A至18M(统称为“机箱交换机18”)的分布层。尽管未示出,但是数据中心10还可以包括例如一个或多个非边缘交换机、路由器、中心、网关、诸如防火墙的安全设备、入侵检测和/或入侵防御设备、服务器、计算机终端、膝上型计算机、打印机、数据库、诸如蜂窝电话或个人数字助理的无线移动设备、无线接入点、网桥、线缆调制解调器、应用加速器或其他网络设备。数据中心10还可以包括一个或多个物理网络功能(PNF),诸如物理防火墙、负载均衡器、路由器、路由反射器、宽带网络网关(BNG)、移动核心网络元件和其他PNF。

[0045] 在该示例中,TOR交换机16和机箱交换机18向服务器12提供到IP结构20和服务提供商网络7的冗余(多宿主)连接。机箱交换机18聚合业务流并提供TOR交换机16之间的连接。TOR交换机16可以是提供层2(MAC)和/或层3(例如,IP)路由和/或交换功能的网络设备。TOR交换机16和机箱交换机18可以各自包括一个或多个处理器和存储器,并且可以执行一个或多个软件过程。机箱交换机18耦接到IP结构20,其可以执行层3路由以通过服务提供商网络7在数据中心10和客户站11之间路由网络流量。数据中心10的交换架构仅是示例。例如,其他交换架构可以具有更多或更少的交换层。IP结构20可以包括一个或多个网关路由器。

[0046] 术语“数据包流”、“业务流”或简单地“流”是指源自特定源设备或端点并被发送到特定目的地设备或端点的一组数据包。单个数据包流可以由5元组来标识:<源网络地址、目的地网络地址、源端口、目的端口、协议>。该5元组通常标识接收到的数据包所对应的数据包流。n元组是指从5元组提取的任何n个项。例如,数据包的2元组可以指数据包的<源网络地址、目的地网络地址>或<源网络地址、源端口>的组合。

[0047] 服务器12可以各自表示计算服务器或存储服务器。例如,每个服务器12可以表示被配置为根据本文所述的技术操作的计算设备,诸如基于x86处理器的服务器。服务器12可为网络功能虚拟化基础设施(NFVI)架构提供NFVI。

[0048] 服务器12中的任何服务器都可以配置有虚拟执行元件,诸如pod或虚拟机,通过虚

拟化服务器的资源以提供在服务器上执行的一个或多个处理(应用)之间的隔离的某种措施。“基于管理程序的”或“硬件级”或“平台”虚拟化是指创建虚拟机,每个虚拟机包括用于执行一个或多个处理的访客操作系统。通常,虚拟机提供用于在隔离的虚拟环境中执行应用的虚拟化/访客操作系统。因为虚拟机从主机服务器的物理硬件虚拟化,所以执行的应用与其他虚拟机和主机的硬件隔离。每个虚拟机可以配置有一个或多个虚拟网络接口,用于在对应的虚拟网络上通信。

[0049] 虚拟网络是在物理网络之上实现的逻辑构造。虚拟网络可用于替代基于VLAN的隔离,并在虚拟化数据中心,诸如数据中心10中提供多租户。每个租户或应用可以具有一个或多个虚拟网络。除非安全策略明确允许,否则每个虚拟网络可以与所有其他虚拟网络隔离。

[0050] 虚拟网络可以使用数据中心10网关路由器(图1中未示出)连接到物理多协议标签交换(MPLS)层3虚拟专用网络(L3VPN)和以太网虚拟专用网络(EVPN)网络并在其上扩展。虚拟网络也可用于实现网络功能虚拟化(NFV)和服务链。

[0051] 可以使用各种机制来实现虚拟网络。例如,每个虚拟网络可以被实现为虚拟局域网(VLAN)、虚拟专用网(VPN)等。虚拟网络也可以使用两个网络来实现,这两个网络是由IP结构20和交换结构14组成的物理底层网络以及虚拟覆盖网络。物理底层网络的作用是提供“IP结构”,其提供从任何物理设备(服务器、存储设备、路由器或交换机)到任何其他物理设备的单播IP连接。底层网络可以提供从网络中的任何点到网络中的任何其他点的统一的低延迟、非阻塞、高带宽连通性。

[0052] 如下面关于虚拟路由器21(被示出并且在本文也被称为“vRouter 21”)进一步所述的,在服务器12中运行的虚拟路由器使用它们之间的动态“隧道”的网格在物理底层网络之上创建虚拟覆盖网络。这些覆盖隧道可以是例如通过GRE/UDP隧道、或者VXLAN隧道,或者NVGRE隧道的MPLS。底层物理路由器和交换机可以不存储虚拟机或其他虚拟执行元件的任何每租户状态,诸如任何媒体访问控制(MAC)地址、IP地址或策略。底层物理路由器和交换机的转发表例如可以仅包含物理服务器12的IP前缀或MAC地址。(将虚拟网络连接到物理网络的网关路由器或交换机是例外,并且可以包含租户MAC或IP地址)

[0053] 服务器12的虚拟路由器21通常包含每租户状态。例如,它们可以包含每个虚拟网络的单独的转发表(路由实例)。该转发表包含虚拟机或其他虚拟执行元件(例如,容器的pod)的IP前缀(在层3覆盖的情况下)或MAC地址(在层2覆盖的情况下)。没有单个虚拟路由器21需要包含整个数据中心中的所有虚拟机的所有IP前缀或所有MAC地址。给定的虚拟路由器21仅需要包含本地存在于服务器12上的那些路由实例(即,其具有至少一个存在于服务器12上的虚拟执行元件)。

[0054] “基于容器”或“操作系统”虚拟化是指操作系统的虚拟化,以在单个机器(虚拟或物理)上运行多个隔离系统。此类隔离系统代表容器,诸如由开源DOCKER容器应用或由CoreOS Rkt(“Rocket”)提供的那些容器。与虚拟机一样,每个容器被虚拟化并且可以保持与主机和其他容器隔离。然而,与虚拟机不同,每个容器可以省略单独的操作系统,而是提供应用套件和应用专用库。通常,容器由主机作为隔离的用户空间实例来执行,并且可以与主机上执行的其他容器共享操作系统和公共库。因此,容器可能比虚拟机(“VM”)需要更少的处理能力、存储和网络资源。一个或多个容器的组可以被配置为共享一个或多个虚拟网络接口,以便在对应的虚拟网络上通信。

[0055] 在一些示例中,容器由它们的主机内核管理以允许资源(CPU、存储器、块I/O、网络等)的限制和优先化而不需要启动任何虚拟机,在一些情况下使用允许完全隔离应用(例如,给定容器)的操作环境视图的命名空间隔离功能,包括进程树、联网、用户标识符和安装的文件系统。在一些示例中,可以根据Linux容器(LXC)部署容器,Linux容器是用于使用单个Linux内核在控制主机上运行多个隔离的Linux系统(容器)的操作系统级虚拟化方法。

[0056] 服务器12托管在本文由IP结构20和交换结构14表示的物理网络上操作的一个或多个虚拟网络的虚拟网络端点。尽管主要关于基于数据中心的交换网络进行描述,但是诸如服务提供商网络7的其他物理网络可以承载一个或多个虚拟网络。

[0057] 每个服务器12可以托管一个或多个虚拟执行元件,每个虚拟执行元件具有用于在物理网络中配置的一个或多个虚拟网络的至少一个虚拟网络端点。虚拟网络的虚拟网络端点可表示共享该虚拟网络的虚拟网络接口的一个或多个虚拟执行元件。例如,虚拟网络端点可以是虚拟机、一个或多个容器(例如,pod)的集合、或另一虚拟执行元件,诸如用于虚拟网络的层3端点。术语“虚拟执行元件”涵盖虚拟机、容器和为应用提供至少部分独立的执行环境的其他虚拟化计算资源。术语“虚拟执行元件”也可以包含一个或多个容器的pod。虚拟执行元件可以表示应用工作负载。如图1所示,服务器12A以具有一个或多个容器的pod 22的形式托管一个虚拟网络端点。然而,给定服务器12的硬件资源限制,服务器12可执行与实际情况一样多的虚拟执行元件。每个虚拟网络端点可以使用一个或多个虚拟网络接口来执行数据包I/O或以其他方式处理数据包。例如,虚拟网络端点可以使用由NIC 13A启用的一个虚拟硬件部件(例如,SR-IOV虚拟功能)来在与TOR交换机16A的一个或多个通信链路上执行数据包I/O和接收/发送数据包。下面描述虚拟网络接口的其他示例。

[0058] 服务器12各自包括至少一个网络接口卡(NIC)13,其各自包括至少一个接口以通过通信链路与TOR交换机16交换数据包。例如,服务器12A包括NIC 13A。NIC 13中的任一个可提供用于虚拟化输入/输出(I/O)的一个或多个虚拟硬件部件21。用于I/O的虚拟硬件部件可以是物理NIC(“物理功能”)的虚拟化。例如,在单根I/O虚拟化(SR-IOV)中,其在外围设备接口特殊兴趣组SR-IOV规范中描述,网络接口卡(或“网络适配器”)的PCIe物理功能被虚拟化以将一个或多个虚拟网络接口呈现为“虚拟功能”,以供在服务器12上执行的相应端点使用。这样,虚拟网络端点可以共享相同的PCIe物理硬件资源,并且虚拟功能是虚拟硬件部件21的示例。作为另一个示例,一个或多个服务器12可以实现虚拟化(Virtio),例如对于Linux操作系统可用的半虚拟化框架,其提供仿真的NIC功能作为一种类型的虚拟硬件部件以向虚拟网络端点提供虚拟网络接口。作为另一个示例,一个或多个服务器12可实现开放vSwitch以在所托管的虚拟机的一个或多个虚拟NIC(vNIC)之间执行分布式虚拟多层交换,其中此类vNIC还可表示向虚拟网络端点提供虚拟网络接口的一种类型的虚拟硬件部件。

[0059] 在一些实例中,虚拟硬件部件是虚拟I/O(例如,NIC)部件。在一些实例中,虚拟硬件部件是SR-IOV虚拟功能。在一些示例中,服务器12中的任何服务器可以实现Linux网桥,该Linux网桥仿真硬件网桥并且在服务器的虚拟网络接口之间或在服务器的虚拟网络接口与服务器的物理网络接口之间转发数据包。对于由服务器、Linux网桥或在服务器上执行的其他操作系统网桥托管的容器的Docker实现,在容器之间交换数据包可以被称为“Docker网桥”。本文使用的术语“虚拟路由器”可以包括Contrail或Tungsten结构虚拟路由器、开放vSwitch(OVS)、OVS网桥、Linux网桥、Docker网桥或位于主机设备上并且在一个或多个虚拟

网络的虚拟网络端点之间执行交换、桥接或路由数据包的其他设备和/或软件,其中虚拟网络端点由一个或多个服务器12托管。

[0060] NIC 13中的任一个可包括内部设备交换机以在与NIC相关联的虚拟硬件部件之间交换数据。例如,对于支持SR-IOV的NIC,内部设备交换机可以是虚拟以太网网桥(VEB),以在SR-IOV虚拟功能之间进行交换,并且相应地,在被配置为使用SR-IOV虚拟功能的端点之间进行交换,其中每个端点可以包括访客操作系统。内部设备交换机可以可替代地称为NIC交换机,或者对于SR-IOV实现,称为SR-IOV NIC交换机。与NIC 13A相关联的虚拟硬件部件可以与层2目的地地址相关联,该目的地地址可以由NIC 13A或负责配置NIC 13A的软件进程分配。物理硬件部件(或SR-IOV实现的“物理功能”)也与层2目的地地址相关联。

[0061] 一个或多个服务器12中的每一个可以包括虚拟路由器21,其执行用于数据中心10内的对应虚拟网络的一个或多个路由实例,以提供虚拟网络接口并在虚拟网络端点之间路由数据包。每个路由实例可以与网络转发表相关联。每个路由实例可以表示用于互联网协议-虚拟专用网(IP-VPN)的虚拟路由转发实例(VRF)。由服务器12A的虚拟路由器21例如从数据中心10的底层物理网络结构(即,IP结构20和交换结构14)接收的数据包可以包括外报头,以允许物理网络结构将有效载荷或“内数据包”隧道传输到执行虚拟路由器的服务器12A的网络接口卡13A的物理网络地址。外报头不仅可以包括服务器的网络接口卡13A的物理网络地址,还可以包括虚拟网络标识符,诸如VxLAN标签或多协议标签交换(MPLS)标签,其标识虚拟网络中的一个以及由虚拟路由器21执行的相应路由实例。内数据包包括具有目的地网络地址的内部报头,该目的地网络地址符合由虚拟网络标识符所标识的虚拟网络的虚拟网络寻址空间。

[0062] 虚拟路由器21终止虚拟网络覆盖隧道,并基于数据包的隧道封装报头来确定所接收数据包的虚拟网络,并将数据包转发到数据包的适当目的地虚拟网络端点。对于服务器12A,例如,对于从由服务器12A托管的虚拟网络端点(例如,pod 22)出站的每个数据包,虚拟路由器21附加指示数据包的虚拟网络的隧道封装头部以生成封装的或“隧道”数据包,并且虚拟路由器21经由虚拟网络的覆盖隧道将封装的数据包输出到物理目的地计算设备,诸如服务器12中的另一个。如本文所使用的,虚拟路由器21可以执行隧道技术端点的操作,以封装由虚拟网络端点源起的内数据包,以生成隧道数据包,并且解封装隧道数据包,以获得用于路由到其他虚拟网络端点的内数据包。

[0063] 在一些示例中,虚拟路由器21可以是基于内核的,并且作为服务器12A的操作系统的内核的一部分来执行。

[0064] 在一些示例中,虚拟路由器21可以是使能数据平面转发工具(Data Plane Development Kit,DPDK)的虚拟路由器。在这样的示例中,虚拟路由器21使用DPDK作为数据平面。在该模式下,虚拟路由器21作为链接到DPDK库(未示出)的用户空间应用而运行。这是虚拟路由器的性能版本,并且通常由telco公司使用,其中VNF通常是基于DPDK的应用。虚拟路由器21作为DPDK虚拟路由器的性能可以实现比作为基于内核的虚拟路由器操作的虚拟路由器高十倍的吞吐量。物理接口由DPDK的轮询模式驱动器(PMD)代替Linux内核的基于中断的驱动器使用。

[0065] 诸如vfio或uio_pci_generic的用户-I/O(UIO)内核模块可用于将物理网络接口的寄存器展露到用户空间中,以使得它们可由DPDK PMD访问。当NIC 13A被绑定到UIO驱动

器时,它从Linux内核空间移动到用户空间,并且因此不再由Linux OS管理或可见。因此,是DPDK应用(即,在该示例中为虚拟路由器21A)完全管理NIC 13。这包括数据包轮询、数据包处理和数据包转发。用户数据包处理步骤可以由虚拟路由器21DPDK数据平面执行,而由内核(其中内核未在图1中示出)参与是有限的或者没有参与。该“轮询模式”的本质使得虚拟路由器21DPDK数据平面数据包处理/转发与中断模式相比高效得多,尤其是在数据包速率高时。在数据包I/O期间存在有限中断和上下文切换或不存在中断和上下文切换。可以在以下找到DPDK vRouter的示例的附加详细信息:“DAY ONE:CONTRAIL DPDK vROUTER,”2021,Kiran KN等,Juniper Networks公司,其通过引用整体并入本文。

[0066] 计算基础设施8实现用于跨服务器12自动化虚拟执行元件的部署、缩放和操作的自动化平台,以提供用于执行应用工作负载和服务的虚拟化基础设施。在一些示例中,平台可以是容器编排系统,其提供以容器为中心的基础设施,用于使容器的部署、缩放和操作自动化以提供以容器为中心的基础设施。在虚拟化计算基础设施的上下文中的“编排”一般是指向编排平台可用的主机服务器供应、调度和管理虚拟执行元件和/或在此类虚拟执行元件上执行的应用和服务。容器编排可以促进容器协调,并且涉及例如由容器编排平台将容器部署、管理、缩放和配置到主机服务器。编排平台的示例性实例包括Kubernetes(容器编排系统)、Docker Swarm、Mesos/Marathon、OpenShift、OpenStack、VMware和Amazon ECS。

[0067] 计算基础设施8的自动化平台的元件至少包括服务器12、编排器23和网络控制器24。可以使用基于集群的框架将容器部署到虚拟化环境,其中集群的集群主节点管理将容器部署到集群的一个或多个集群工作节点和对集群工作节点的操作。本文使用的术语“主节点”和“工作节点”包含用于类似装置的不同编排平台术语,其在集群的主要管理元件和集群的主要容器托管装置之间进行区分。例如,Kubernetes平台使用术语“集群主机”和“工作节点”,而Docker Swarm平台是指集群管理器和集群节点。

[0068] 编排器23和网络控制器24可以在分开的计算设备上执行,在同一计算设备上执行。编排器23和网络控制器24中的每一个可以是在一个或多个计算设备上执行的分布式应用。编排器23和网络控制器24可以实现用于一个或多个集群的相应主节点,每个集群具有由相应服务器12实现的一个或多个工作节点(也称为“计算节点”)。

[0069] 通常,网络控制器24控制数据中心10结构的网络配置,以例如建立一个或多个虚拟网络,用于虚拟网络端点之间的封包通信。网络控制器24提供逻辑上和在一些情况下物理上集中式的控制器,以便于数据中心10内的一个或多个虚拟网络的操作。在一些示例中,网络控制器24可以响应于从编排器23和/或管理员/操作者接收的配置输入而操作。在2013年6月5日提交的命名为“PHYSICAL PATH DETERMINATION FOR VIRTUAL NETWORK PACKET FLOWS”的国际申请号PCT/US2013/044378和2014年3月26日提交的名称为“TUNNELED PACKET AGGREGATION FOR VIRTUAL NETWORKS”的美国专利申请第14/226,509号中找到关于结合数据中心10的其他设备或其他软件定义网络操作的关于网络控制器24的示例性操作的附加信息,其每一个通过引用并入本文,如同在本文完全阐述一样。

[0070] 通常,编排器23控制跨服务器12的集群的容器的部署、缩放和操作以及计算基础设施的提供,该计算基础设施可以包括以容器为中心的计算基础设施。编排器23以及在一些情况下的网络控制器24可以实现用于一个或多个Kubernetes集群的相应集群主机。作为示例,Kubernetes是提供跨公共和私有云的可移植性的容器管理平台,公共和私有云中的

每一个都可以向容器管理平台提供虚拟化基础设施。Kubernetes编排系统的示例性部件在下面参考图3描述。

[0071] 在一个示例中,pod 22是Kubernetes pod和虚拟网络端点的示例。pod是一个或多个逻辑上相关的容器(图1中未示出)、容器的共享存储以及如何运行容器的选项的组。在被例示以便执行时,pod可以可替代地被称为“pod副本”。pod 22的每个容器是虚拟执行单元的一个示例。pod的容器总是共同位于单个服务器上、共同调度、以及在共享的上下文中运行。pod的共享上下文可以是一组Linux命名空间、cgroups和隔离的其他方面。

[0072] 在pod的上下文中,单独的应用可以应用另外的子隔离。通常,pod内的容器具有公共IP地址和端口空间,并且能够经由本地主机来检测彼此。因为它们具有共享的上下文,所以pod内的容器也使用进程间通信(IPC)彼此通信。IPC的示例包括SystemV信号量或POSIX共享存储器。通常,作为不同pod的成员的容器具有不同的IP地址,并且在缺少用于实现该特征的配置的情况下不能通过IPC进行通信。作为不同pod的成员的容器通常改为经由pod IP地址相互通信。

[0073] 服务器12A包括用于运行容器化应用(诸如pod 22的容器化应用)的容器平台19。容器平台19从编排器23接收请求以在服务器12A中获得和托管容器。容器平台19获得并执行容器。

[0074] 容器网络接口(CNI)17为虚拟网络端点配置虚拟网络接口。编排器23和容器平台19使用CNI 17来管理包括pod 22的pod的联网。例如,CNI 17创建虚拟网络接口以将pod连接到虚拟路由器21,并且使得这种pod的容器能够经由虚拟网络接口通过虚拟网络与其他虚拟网络端点通信。CNI 17可以例如将用于虚拟网络的虚拟网络接口插入到用于pod 22中的容器的网络命名空间中,并且在虚拟路由器21中配置(或请求配置)用于虚拟网络的虚拟网络接口,使得虚拟路由器21被配置为经由虚拟网络接口将从虚拟网络接收的数据包发送到pod 22的容器,并且发送经由虚拟网络接口从虚拟网络上的pod 22的容器接收的数据包。CNI 17A可以分配网络地址(例如,用于虚拟网络的虚拟IP地址)并且可以为虚拟网络接口建立路由。

[0075] 在Kubernetes中,默认地,所有的pod可以与所有其他pod通信,而不使用网络地址转换(NAT)。在一些情况下,编排器23和网络控制器24创建由所有命名空间共享的服务虚拟网络和pod虚拟网络,从其分别分配服务和pod网络地址。在一些情况下,在Kubernetes集群中引起的所有命名空间中的所有pod可能能够彼此通信,并且所有pod的网络地址可从编排器23指定的pod子网分配。当用户为pod创建隔离的命名空间时,编排器23和网络控制器24可以为新的隔离的命名空间创建新的pod虚拟网络和新的共享服务虚拟网络。在Kubernetes集群中引起的隔离的命名空间中的pod从新pod虚拟网络中提取网络地址,并且用于此类pod的对应服务从新服务虚拟网络中提取网络地址。

[0076] CNI 17可以表示用于服务器12A的库、插件、模块、运行时间或其他可执行代码。CNI 17A可至少部分地符合容器网络接口(CNI)规范或rkt联网提议。CNI 17可表示Contrail、OpenContrail、Multus、Calico、cRPD或其他CNI。CNI 17可以替代地被称为网络插件或CNI插件或CNI实例。例如,可以由Multus CNI调用分开的CNI,以便为pod 22建立不同的虚拟网络接口。

[0077] CNI 17可以由编排器23调用。为了CNI规范的目的,容器可以被认为与Linux网络

命名空间同义。对应于什么单元取决于特定的容器运行时间实现:例如,在诸如rkt的应用容器规范的实现中,每个pod在唯一的网络命名空间中运行。然而,在Docker中,网络命名空间通常对于每个单独的Docker容器而存在。为了CNI规范的目的,网络是指一组可唯一寻址并且可以彼此通信的实体。这可以是单独的容器、机器/服务器(真实的或虚拟的)或一些其他网络设备(例如,路由器)。容器可以在概念上添加到一个或多个网络或从一个或多个网络移除。CNI规范指定了用于符合插件(“CNI插件”)的多个考虑。

[0078] pod 22包括一个或多个容器。在一些示例中,pod 22包括容器化的DPDK工作负载,其被设计为使用DPDK来加速数据包处理,例如通过使用DPDK库与其他部件交换数据。在一些示例中,虚拟路由器21可以作为容器化的DPDK工作负载来执行。

[0079] pod 22配置有用于与虚拟路由器21发送和接收数据包的虚拟网络接口26。虚拟网络接口26可以是pod 22的默认接口。pod 22可以将虚拟网络接口26实现为以太网接口(例如,命名为“eth0”),而虚拟路由器21可以将虚拟网络接口26实现为tap接口、virtio-用户界面或其他类型的接口。

[0080] pod 22和虚拟路由器21使用虚拟网络接口26交换数据数据包。虚拟网络接口26可以是DPDK接口。pod 22和虚拟路由器21可以使用vhost建立虚拟网络接口26。pod 22可以根据聚合模型操作。pod 22可以使用虚拟设备,诸如具有vhost用户适配器的virtio设备,用于虚拟网络接口26的用户空间容器进程间通信。

[0081] CNI 17可以结合图1所示的一个或多个其他部件为pod 22配置虚拟网络接口26。pod 22的任何容器都可以利用(即,共享)pod 22的虚拟网络接口26。

[0082] 虚拟网络接口26可以表示虚拟以太网(“veth”)对,其中该对的每一端是单独的设备(例如,Linux/Unix设备),该对的一端被分配给pod 22,而该对的一端被分配给虚拟路由器21。veth对或veth对的端部有时被称为“端口”。虚拟网络接口可以表示具有分配给pod 22和虚拟路由器21的媒体访问控制(MAC)地址的macvlan网络,用于pod 22和虚拟路由器21的容器之间的通信。虚拟网络接口可以替代地被称为例如虚拟机接口(VMI)、pod接口、容器网络接口、tap接口、veth接口或简单地称为网络接口(在特定上下文中)。

[0083] 在图1的示例性服务器12A中,pod 22是一个或多个虚拟网络中的虚拟网络端点。编排器23可以存储或以其他方式管理用于应用部署的配置数据,该配置数据指定虚拟网络并且指定pod 22(或其中的一个或多个容器)是虚拟网络的虚拟网络端点。编排器23可从例如用户、操作员/管理员或其他计算系统接收配置数据。

[0084] 作为创建pod 22的过程的一部分,编排器23请求网络控制器24为一个或多个虚拟网络(在配置数据中指示)创建相应的虚拟网络接口。对于pod 22所属的每个虚拟网络,pod 22可以具有一不同的虚拟网络接口。例如,虚拟网络接口26可以是用于特定虚拟网络的虚拟网络接口。可以为其他虚拟网络配置附加的虚拟网络接口(未示出)。

[0085] 网络控制器24处理该请求以生成用于pod 22的虚拟网络接口的接口配置数据。接口配置数据可以包括容器或pod唯一标识符和为每个虚拟网络接口指定用于配置虚拟网络接口的网络配置数据的列表或其他数据结构。虚拟网络接口的网络配置数据可以包括网络名称、分配的虚拟网络地址、MAC地址和/或域名服务器值。下面是JavaScript Object Notation(JSON)格式的接口配置数据的示例。

[0086] 网络控制器24将接口配置数据发送到服务器12A,更具体地说,在某些情况下,发

送到虚拟路由器21。为了配置pod 22的虚拟网络接口,编排器23可以调用CNI 17。CNI 17从虚拟路由器21获取接口配置数据并对其进行处理。CNI 17A创建接口配置数据中指定的每个虚拟网络接口。例如,CNI 17可以将实现管理接口26的veth对的一端附接到虚拟路由器21,并且可以将同一veth对的另一端附接到可以使用virtio-用户来实现它的pod 22。

[0087] 以下是虚拟网络接口26的pod 22的示例性接口配置数据。

[0088] [{

[0089] //virtual network interface 26

[0090] "id":"fe4bab62-a716-11e8-abd5-0cc47a698428",

[0091] "instance-id":"fe3edca5-a716-11e8-822c-0cc47a698428",

[0092] "ip-address":"10.47.255.250",

[0093] "plen":12,

[0094] "vn-id":"56dda39c-5e99-4a28-855e-6ce378982888",

[0095] "vm-project-id":"00000000-0000-0000-0000-000000000000",

[0096] "mac-address":"02:fe:4b:ab:62:a7",

[0097] "system-name":"tapeth0fe3edca",

[0098] "rx-vlan-id":65535,

[0099] "tx-vlan-id":65535,

[0100] "vhostuser-mode":0,

[0101] “v6-ip-address”:“::“,

[0102] “v6-plen”:,

[0103] “v6-dns-server”:“::”,

[0104] “v6-gateway”:“::”,

[0105] "dns-server":"10.47.255.253",

[0106] "gateway":"10.47.255.254",

[0107] "author":"/usr/bin/contrail-vrouter-agent",

[0108] "time":"426404:56:19.863169"

[0109] }]

[0110] 常规CNI插件由容器平台/运行时间调用,从容器平台接收Add命令以将容器添加到单个虚拟网络,并且此类插件随后可以被调用以从容器/运行时间接收Del(ete)命令并且从虚拟网络移除容器。术语“调用”可以指将存储器中的软件部件或模块例示为可执行代码以供处理电路执行。

[0111] 根据本公开所述的技术,网络控制器24是用于软件定义网络(software-defined networking,SDN)的云原生分布式网络控制器,该云其使用一个或多个配置节点30和一个或多个控制节点32来实施。每个配置节点30本身可以使用一个或多个云原生部件微服务来实现。每个控制节点32本身可以使用一个或多个云原生部件微服务来实现。

[0112] 在一些示例中,配置节点30可以通过扩展本地编排平台来实现以支持用于软件定义网络的编排平台的定制资源,并且更具体地,通过例如配置用于虚拟执行元件的虚拟网络接口、配置连接服务器12的底层网络、配置包括用于虚拟网络的覆盖隧道和用于多播层2和层3的覆盖树的覆盖路由功能,来提供到编排平台的北向接口以支持虚拟网络的意图驱

动/声明性创建和管理。

[0113] 作为图1所示的SDN架构的一部分,网络控制器24可以是多租户感知的,并且支持用于编排平台的多租户。例如,网络控制器24可以支持Kubernetes基于角色的访问控制(RBAC)构造、本地身份访问管理(IAM)和外部IAM集成。网络控制器24还可以支持Kubernetes定义的联网构造和高级联网特征,如虚拟联网、BGPaaS、联网策略、服务链和其他telco特征。网络控制器24可以使用虚拟网络结构来支持网络隔离,并且支持层3联网。

[0114] 为了互连多个虚拟网络,网络控制器24可以使用(并在底层和/或虚拟路由器21中配置)使用虚拟网络路由器(VNR)资源定义的引入和引出策略。虚拟网络路由器资源可以用于通过配置用于实现SDN架构中的虚拟网络的相应路由实例之间的路由信息的引入和引出来定义虚拟网络之间的连通性。单个网络控制器24可支持多个Kubernetes集群,并且VNR因此允许连接命名空间中的多个虚拟网络、不同命名空间中的虚拟网络、Kubernetes集群以及跨Kubernetes集群。VNR也可以扩展为支持网络控制器24的多个实例之间的虚拟网络连通性。VNR在本文也可以可替代地被称为虚拟网络策略(VNP)或虚拟网络拓扑。

[0115] 如图1的示例所示,网络控制器24可维护表示虚拟网络(VN)50A至50N(“VN 50”)的配置数据(例如,config.30),该虚拟网络表示用于通过物理底层网络和/或虚拟路由器(诸如虚拟路由器21(“vRouter 21”))在数据中心10内建立VN 50的策略和其他配置数据。网络控制器24还可维护表示虚拟网络路由器(VNR)52A至52N(“VNR 52”)的配置数据(例如,config.30),该虚拟网络路由器可至少部分地使用策略和其他配置数据来实现,以便建立VN 50之间的互连性。

[0116] 诸如管理员的用户可以与网络控制器24的UI 60交互以定义VN 50和VNR 52。在一些情况下,UI 60表示便于输入定义VN 50和VNR 52的配置数据的图形用户界面(GUI)。在其他实例中,UI 60可表示命令行界面(CLI)或其他类型的界面。假设UI 60表示图形用户界面,则管理员可以通过将表示诸如pod 22等不同pod的图形元件布置为将pod与VN 50相关联来定义VN 50,其中VN 50中的任一个使能分配给VN的一个或多个pod之间的通信。

[0117] 在这方面,管理员可以理解Kubernetes或其他编排平台,但不能完全理解支持VN 50的底层基础设施。一些控制器架构,诸如Contrail,可以基于与传统物理网络中的路由协议类似(如果不是基本上类似的话)的网络协议来配置VN 50。例如,Contrail可以利用来自边界网关协议(BGP)的概念,边界网关协议是用于在所谓的自治系统(AS)内并且有时在AS之间传送路由信息的路由协议。

[0118] 存在BGP的不同版本,诸如用于在AS内传送路由信息的内部BGP(iBGP)和用于在AS之间传送路由信息的外部BGP(eBGP)。ASE可能涉及Contrail内的项目的概念,其也类似于Kubernetes中的命名空间。在AS、项目和命名空间的每个实例中,类似项目和命名空间的AS可以表示一个或多个网络(例如,VN 50中的一个或多个)的集合,其可以共享路由信息并由此促进网络(或者,在该实例中,VN 50)之间的互连性。

[0119] 在最简单的形式中,VNR 52表示在Kubernetes的上下文中设置的路由器的逻辑抽象,其中VNR 52可以被定义为便于VN 50之间的互连性的定制资源。假定Kubernetes管理员可能不完全理解根据复杂的路由协议(诸如BGP)的路由信息的复杂散布,云原生联网技术的各个方面可促进作为VNR 52的底层路由协议(或Contrail或其他控制器架构的补充过程)的抽象。

[0120] 即,管理员可定义一个或多个VNR 52以互连VN 50,而不必手动开发和部署广泛的策略和/或路由实例配置以实现这样的VN 50之间的路由信息的交换,而不是采取定义在两个或更多个VN 50之间如何发生路由的手段。相反,管理员(可能对路由协议几乎没有理解)可使用熟悉的Kubernetes语法/语义(或甚至仅通过拖动图形元件并指定表示例如VNR 52A的该图形元件与表示例如VN 50A和50N的图形元件之间的互连)来定义定制资源(例如,VNR 52中的一个或多个)。

[0121] 在这方面,管理员可使用图1的示例中所示的逻辑抽象容易地将VN 50互连为VNR 50,于是网络控制器24可将VNR 50转换为底层路由目标,以自动(意味着很少或可能没有任何人为干预)使VN 50A和50N的路由信息被交换,并使VN 50A和50N之间能够通信(意味着数据包或其他数据的交换)。

[0122] 假定管理员可以采用熟悉的Kubernetes语法/语义来配置VNR 50,而不是配置符合路由协议语法/语义的复杂配置数据,网络控制器24可有助于更好的用户体验,同时也促进数据中心8本身的更有效的运行。即,让管理员输入这样的管理员不熟悉的配置数据可使得浪费数据中心8的底层资源(在处理周期、存储器、总线带宽等以及相关联的功率方面)的错误配置,同时还延迟网络拓扑的适当实现(这可阻止VN 50之间的数据包和其他数据的成功路由)。这种延迟不仅会使管理员沮丧,而且还会使与VN 50相关联的客户失望,这可能需要迅速操作VN 50以实现各种商业目标。通过使管理员能够使用被示出为VNR 50的逻辑抽象来容易地有助于VN 50之间的通信,数据中心8本身可以经历更高效的操作(就上述计算资源而言,包括处理器周期、存储器、总线带宽和相关联的功率),同时为管理员和客户提供更好的用户体验。

[0123] 网络控制器24,代表数据中心10的SDN架构系统,包括处理电路以实现配置节点和控制节点(如关于图3的示例更详细地描述的)。网络控制器24可被配置为互连在由数据中心10表示的SDN架构系统内操作的第一虚拟网络(例如,VN 50A)和第二虚拟网络(例如,VN 50N)。网络控制器24可被配置为定义一个或多个策略的逻辑抽象,以便经由一个或多个VNR 52,例如VNR 52A执行这样的互连。

[0124] 策略可包括关于虚拟网络(在此示例中,可指VN 50A和50N)所维护的路由信息的引入和引出策略。即,Kubernetes可通过代表VNR 52A的定制资源被扩展,以将VNR 52A转换成一个或多个引入和引出策略,该引入和引出策略关于VN 50A和VN 50N被部署,以便经由VN 50A和VN 50N之间的路由信息分发来配置互通。一旦被配置,VN 50A可将路由信息(例如,表示VN 50A的路由)引出到VN 50N,并将路由信息(例如,表示VN 50N的路由)引入到VN 50A。同样地,VN 50N可将路由信息(例如,表示VN 50N的路由)引出到VN 50A,并将路由信息(例如,表示VN 50A的路由)引入到VN 50N。

[0125] 抽象可隐藏底层路由配置以实现这种路由泄漏,诸如定义路由信息到用于实现VN 50A和VN 50N的路由实例的引入和引出的路由目标。相反,网络控制器24可将VNR 52A转换为公共路由目标,并配置用于实现VN 50A和VN 50N(在该示例中)的路由实例的经由公共路由目标的路由信息的通信。

[0126] 为了实现网格连通性,网络控制器24可以利用与VN 50A、VN 50N和VNR 52A相关联的路由目标来配置VN 50A、VN 50N和VNR 52A的路由实例的引入和引出。为了实现中心辐条式连通性,网络控制器24可配置与VN 50A和VN 50N相关联的路由实例的引出,以将路由信

息引出到与VNR 52A(充当中心)相关联的路由实例,以及配置VNR 52A的路由实例,以将路由信息引入到与VN 50A和VN 50N相关联的路由实例。在该中心辐条式连通性中,VN 50A和VN 50N可以不彼此直接通信。关于VNR的更多信息可以在2022年7月29日提交的名称为“VIRTUAL NETWORK ROUTERS FOR CLOUD NATIVE SOFTWARE-DEFINED NETWORK ARCHITECTURES”的美国申请第17/809,659号中找到,其全部内容通过引用并入本文。

[0127] 虽然VNR可以在将VNR抽象自动转换为底层路由配置方面降低复杂度,但是仍然难以可视化VNR将如何适配可以包括一个或多个VN(诸如VN 50A和VN 50N)的网络拓扑以便于VN 50A和VN 50N之间的连通性。这样的困难可能使得在配置诸如VNR 52A的VNR时的用户错误,这可能使得网络控制器24和由网络控制器24管理的底层网络的有效操作。

[0128] 根据本公开所述的技术的各个方面,网络控制器24可以呈现UI 60(例如,图形用户界面,其可以被表示出为GUI 60),通过该UI 60来可视化网络的拓扑,并且通过与GUI 60的各种交互来交互地定义诸如VNR 52A的VNR。添加用于抽象控制路由信息的引入和/或引出的策略的VNR 52可简化将网络元件(诸如VN 50A和50N)拼接在一起以促进此类网络元件之间的互连。本公开所述的技术的各方面可以提供GUI 60,该GUI 60被配置为图形地呈现可以包括各种网络元件(诸如VN 50A和50N)的网络拓扑,而不是借助于复杂的接口来定义涉及可能需要对网络构造(诸如路由目标、标识网络元件的标签等)的复杂声明的VNR 52。用户可以与GUI 60接口连接以可视地配置VNR 52A以互连两个或更多个网络元件。

[0129] GUI 60可以接收用户输入(其可以被称为经由GUI 60输入的输入),诸如拖放或以其他方式定义VNR 52A的图形用户输入。GUI 60可以将这些输入作为配置VNR 52A的请求传递给网络控制器24。用户可通过与GUI 60的交互来输入,以图形地定义VNR 52A,选择VN 50A和50N以及VNR 52A针对其建立互连性(例如,网格连通性和/或中心辐条式连通性)的其他网络元件。网络控制器24可处理这些输入并更新GUI 60以呈现提示和其他GUI 60,通过该GUI 60来提示用户定义VNR 52A所需的附加输入。网络控制器24可以将定义VNR 52A的这些输入减少到控制VN 50A和50N之间的路由信息的引入和/或引出的各种策略中,并且在适当的网络元件中配置策略以便于实现VN 50A和50N之间的互连性的路由信息的交换。

[0130] 在操作中,网络控制器24可以本地地(经由本地显示器)呈现GUI 60。在一些示例中,GUI 60可以表示基于web的GUI,网络控制器24可以经由客户端设备(诸如远程计算机或其他终端)远程托管该基于web的GUI以用于显示。不管如何呈现GUI 60,GUI 60可被配置为图形地表示由软件定义网络(SDN)架构系统8支持的网络的拓扑。出于说明的目的,假设网络包括VN 50A和VN 50N。然而,网络可以包括任何数量的VN 50或其他网络元件。

[0131] 网络控制器24可以维护定义网络的拓扑的配置数据(在图1的示例中未示出),网络控制器24可以将其变换成网络的拓扑的图形表示。网络拓扑的图形表示可以便于查看网络的一部分或全部,其中GUI 60可以提供过滤和其他操作,以使得用户能够以各种粒度级别查看网络拓扑的图形表示,如下面更详细描述的。

[0132] GUI 60还可以被配置为动态地生成表示VNR 52A的图形元件,通过其互连VN 50A和VN 50N。GUI 60可以接收来自用户的标识VNR 52A和VN 50A和50N的输入,诸如指示代表VNR的图形元件(例如,图形图标)已被拖到可能接近、邻近和/或在VN 50A和50N中的一个或多个之上的网络拓扑上的输入。GUI 60可以将这些输入提供回网络控制器24,该网络控制器24可以响应于这些输入、更新GUI 60以包括关于VNR 52A和VN 50A和50N的信息的附加提

示。

[0133] GUI 60(经由网络控制器24)可以以这种方式迭代,直到VNR 52A已经以实现VN 50A和50N之间的连通性的方式被成功定义。网络控制器24可以执行配置节点30以在调用控制节点32来配置VN 50A和50N之前验证VNR 52A。一旦被成功验证,控制节点32就根据一个或多个策略来配置VN 50A和50N,以实现经由VNR 52A在VN 50A和VN 50N之间引入和引出路由信息中的一个或多个。

[0134] 这样,根据本公开所述的技术的各个方面配置的GUI 60可使得无经验的用户(在SDN架构的上下文中)能够定义VNR 52,而不需要具有网络构造(诸如上面列出的那些)的广泛知识。这样,用户可以更有效地与GUI 60交互以定义VNR 52来满足企业和/或客户目标。GUI 60可引导用户减少并可能消除配置VNR 52中的错误,同时还使用户能够查看网络的拓扑并获得对网络本身的更好理解。通过易于使用并且消除错误,GUI 60可以减少用户定义复杂配置数据的需要,同时还避免可能使得网络控制器的低效操作(在计算资源方面,诸如处理周期、存储器总线带宽等,以及相关联的功率)的此类错误。这样,GUI 60可以改善网络控制器自身的运行。

[0135] 此外,网络控制器24可以使用网络策略来实现多层安全性。Kubernetes默认行为是pod彼此通信。为了应用网络安全策略,由网络控制器24和虚拟路由器21实现的SDN架构可以通过CNI 17作为Kubernetes的CNI来操作。对于层3,隔离发生在网络级,并且虚拟网络在L3操作。虚拟网络通过策略连接。Kubernetes本地网络策略在层4提供安全性。SDN架构可以支持Kubernetes网络策略。Kubernetes网络策略在Kubernetes命名空间边界处操作。SDN架构可以添加用于增强网络策略的定制资源。SDN架构可以支持基于应用的安全性。(在某些情况下,这些安全策略可以基于元标记,以便以可扩展的方式应用粒度安全策略)。对于层4+,SDN架构在一些示例中可以支持与容器式安全设备和/或Istio的集成,并且可以提供加密支持。

[0136] 作为图1所示的SDN架构的一部分,网络控制器24可以支持多集群部署,这对于telco云和高端企业使用情况是重要的。例如,SDN架构可以支持多个Kubernetes集群。集群API可用于支持Kubernetes集群的生命周期管理。KubefedV2可用于跨Kubernetes集群的配置节点32联合。集群API和KubefedV2是用于对支持多个Kubernetes集群的网络控制器24的单个实例进行支持的可选部件。

[0137] SDN架构可以使用web用户界面和遥测部件来提供对基础设施、集群和应用的洞察。遥测节点可以是云原生的并且包括微服务以支持洞察。

[0138] 作为上述特征和将在本文别处所述的其他特征的结果,计算基础设施8实现作为云原生的SDN架构,并且可以呈现以下技术优点中的一个或多个。例如,网络控制器24是具有简化的安装覆盖区的云原生、轻量分布式应用。这还便于配置节点30和控制节点32(以及本公开所述的网络控制器的其他示例的任何其他部件)的各种部件微服务的更容易和模块化的升级。该技术还可以实现可选的云原生监视(遥测)和用户界面、使用连接到启用DPDK的pod的基于DPDK的虚拟路由器的容器的高性能数据平面、以及在一些情况下利用用于现有编排平台的配置框架(诸如Kubernetes或Openstack)的云原生配置管理。作为云原生架构,网络控制器24是用于寻址和支持多个集群的可缩放和弹性架构。在某些情况下,网络控制器24还可以支持关键性能指标(KPI)的可缩放性和性能要求。

[0139] 具有诸如在本文所述的特征和技术优点的SDN架构可以用于实现云原生telco云以支持例如5G移动联网(和后续代)和边缘计算,以及包括例如高性能云原生应用托管的企业Kubernetes平台。telco云应用正迅速向容器化的云原生方法发展。5G固定和移动网络正在驱动将工作负载部署为具有显著分离的微服务的需求,特别是在5G Next-Gen RAN(5GNR)中。5G NextGen Core(5GNC)可能被部署为与3GPP描述的不同部件中的每一个相对应的基于微服务的应用集合。当被视为微服务递送应用的组时,5GNC可能是具有复杂联网、安全性和策略要求的pod的高度复杂组合。对于这种使用情况,可以利用本文所述的云原生SDN架构,其具有用于联网、安全和策略的明确定义的构造。网络控制器24可以提供能够创建这些复杂结构的相关API。

[0140] 同样,5GNC内的用户平面功能(UPF)将是超高性能应用。它可以作为一组高度分布的高性能pod来递送。本文所述的SDN架构能够提供非常高的吞吐量数据平面(按照每部分比特(bps,bits per section)和每秒数据包(pps))。与具有最近性能增强的DPDK虚拟路由器、eBPF以及与SmartNIC的集成将有助于实现所需的吞吐量。基于DPDK的虚拟路由器在第17/649,632号美国申请中有更详细的描述,该申请在2022年2月1日提交,名为“CONTAINERIZED ROUTER WITH VIRTUAL NETWORKING”,通过引用将其整体并入本文。

[0141] 高性能处理可能在GiLAN中也是相关的,因为工作负载从更传统的虚拟化工作负载迁移到容器化微服务。在UPF和GiLAN服务的数据平面中,诸如GiLAN防火墙、入侵检测和预防、虚拟化IP多媒体子系统(vIMS)语音/视频等,吞吐量将是高的,并且在bps和pps方面都是持续的。对于5GNC功能的控制平面功能,诸如接入和移动性管理功能(AMF)、会话管理功能(SMF)等,以及对于一些GiLAN服务(例如,IMS),虽然就bps而言绝对业务量可能是适度的,但是小数据包的优势意味着pps将保持较高。在一些示例中,SDN控制器和数据平面提供每虚拟路由器21每秒数百万个数据包,如在服务器12上实现的。在5G无线电接入网络(RAN)中,为了远离由传统无线电供应商提供的专有垂直集成RAN栈,开放RAN将多个部件中的RAN硬件和软件解耦接,该部件包括非RT无线电智能控制器(RIC)、近实时RIC、集中式单元(CU)控制平面和用户平面(CU-CP和CU-UP)、分布式单元(DU)和无线电单元(RU)。软件部件部署在商品服务器架构上,必要时补充可编程加速器。本文所述的SDN架构可以支持O-RAN规范。

[0142] 边缘计算可能主要针对两种不同的使用情况。第一种将作为对容器化telco基础设施(例如,5G RAN、UPF、安全功能)的支持,第二种将用于来自telco以及来自诸如厂商或企业客户的第三方的容器化服务工作量。在这两种情况下,边缘计算实际上是GiLAN的特殊情况,其中流量被中断以在高度分布的位置处进行特殊处理。在许多情况下,这些位置将具有有限的资源(功率、冷却、空间)。

[0143] 本文所述的SDN架构可以非常适合于支持非常轻量的覆盖区的要求,可以支持远离相关联的控制功能的站中的计算和存储资源,并且可以以部署工作负载和存储的方式知道位置。一些站可以具有少至一个或两个计算节点,其向高度本地化的用户或其他服务集合递送非常特定的服务集合。可能存在站的分级结构,其中中心站密集地与许多路径连接,区域站与二到四个上行链路路径多重连接,并且远程边缘站可以具有到仅一个或两个上游站的连接。

[0144] 这要求SDN架构可以被部署的方式以及覆盖中的隧道化业务被终止并绑定到核心传输网络(SRv6、MPLS等)中的方式(和位置)的极大灵活性。同样地,在托管telco云基础设

施工作负载的站中,本文所述的SDN架构可以支持高性能工作负载所需的专用硬件(GPU、SmartNIC等)。还可能存在需要SR-IOV的工作负载。这样,SDN架构还可以支持在ToR处创建VTEP并且将其链接回覆盖作为VXLAN。

[0145] 预期将存在全分布式Kubernetes微集群的混合,其中每个站运行其自己的主设备,并且SDN架构可以支持类似远程计算的场景。

[0146] 对于涉及企业Kubernetes平台的用例,高性能云原生应用为金融服务平台、在线游戏服务和托管的应用服务提供商提供动力。递送这些应用的云平台必须提供高性能、对故障的弹性、以及高安全性和可见性。在这些平台上托管的应用往往是内部开发的。应用开发者和平台所有者与基础设施团队合作以部署和操作组织的应用的实例。这些应用往往需要高吞吐量(>20Gbps每服务器)和低时延。一些应用还可以使用多播用于信令或有效载荷业务。可以利用附加的硬件和网络基础设施来确保可用性。应用和微服务将利用集群内的命名空间来进行分区。命名空间之间的隔离在高安全性环境中是关键的。尽管默认拒绝策略是零信任应用部署环境中的标准态势,但是使用虚拟路由和转发实例(VRF)的附加网络分段添加了附加的安全层并且允许使用重叠的网络范围。重叠的网络范围是被管理的应用托管环境的关键要求,其倾向于在所有被管理的客户的一组可达端点上标准化。

[0147] 复杂的基于微服务的应用倾向于利用复杂的网络过滤。本文所述的SDN架构可以递送按比例过滤的高性能防火墙。这样的过滤可以表现出一致的转发性能,具有较小的延迟降级,而不管规则集长度或顺序。一些客户可能还具有一些与telcos相同的关于应用分离的管理压力,不仅在网络层,而且在内核中。金融以及其他方面都要求数据平面加密,特别是当在公共云上运行时。在一些示例中,本文所述的SDN架构可以包括用于满足这些要求的特征。

[0148] 在一些示例中,当SDN架构通过应用dev/test/stage/prod连续集成/连续开发(CI/CD)流水线被自动化时,SDN架构可以提供GitOps友好的UX用于每天若干次、甚至每天数百次的生产中进行改变的严格改变管理控制、审计和可靠性。

[0149] 图2是图示了根据本公开的技术的用于云原生联网的云原生SDN架构的示例的框图。SDN架构200以抽象各种部件之间的底层连通性的方式示出。在该示例中,SDN架构200的网络控制器24包括配置节点230A至230N(“配置节点”或“config nodes”,统称为“配置节点230”)和控制节点232A至232K(统称为“控制节点232”)。配置节点230和控制节点232可以分别表示图1的配置节点30和控制节点32的示例性实施方式。配置节点230和控制节点232虽然被图示为与服务器12分离,但是可以作为服务器12上的一个或多个工作负载来执行。

[0150] 配置节点230提供北向、表述性状态转移(REST)接口,以支持SDN架构200的意图驱动配置。可用于将意图推送到配置节点230的示例平台和应用包括虚拟机编排器240(例如,Openstack)、容器编排器242(例如,Kubernetes)、用户界面242或其他一个或多个应用246。在一些示例中,SDN架构200具有Kubernetes作为其基础平台。

[0151] SDN架构200被划分为配置平面、控制平面和数据平面,以及可选的遥测(或分析)平面。配置平面用水平可伸缩配置节点230实现,控制平面用水平可伸缩控制节点232实现,并且数据平面用计算节点实现。

[0152] 在高等级,配置节点230使用配置存储224来管理SDN架构200的配置资源的状态。通常,配置资源(或更简单地“资源”)是命名对象模式,其包括描述定制资源的数据和/或方

法,并且定义应用编程接口(API)以通过API服务器创建和操纵数据。一种是对象模式的命名。配置资源可以包括Kubernetes原生资源,诸如Pod、入口(Ingress)、配置图(Configmap)、服务(Service)、角色(Role)、命名空间(Namespace)、节点(Node)、网络策略(Networkpolicy)或负载均衡器(LoadBalancer)。

[0153] 配置资源还可以包括定制资源,其用于通过定义在Kubernetes平台的默认安装中可能不可用的应用程序接口(API)来扩展Kubernetes平台。在SDN架构200的示例中,定制资源可描述SDN架构200的物理基础设施、虚拟基础设施(例如,VN 50和/或VNR 52)、配置和/或其他资源。作为配置和操作SDN架构200的一部分,可以例示各种定制资源(例如,vRouter 21内的VNR 52)。例示的资源(无论是原生的还是定制的)可以被称为对象或资源的实例,它们是SDN架构200中的持久实体,表示SDN架构200的意图(期望状态)和状况(实际状态)。

[0154] 配置节点230提供用于对配置存储224中的SDN架构200的配置资源执行操作(即,创建、读取、更新和删除)的聚合的API。负载均衡器226表示在配置节点230之间负载均衡配置请求的一个或多个负载均衡器对象。配置存储224可以表示一个或多个etcd数据库。可以使用Nginx来实现配置节点230。

[0155] SDN架构200可以为Openstack和Kubernetes提供联网。Openstack使用插件架构来支持联网。利用作为Openstack的虚拟机编排器240,Openstack联网插件驱动器将Openstack配置对象转换为SDN架构200配置对象(资源)。计算节点运行Openstack nova以产生虚拟机。

[0156] 对于作为Kubernetes的容器编排器242,SDN架构200用作Kubernetes CNI。如上所述,可以支持Kubernetes原生资源(pod、服务、入口、外部负载均衡器等),并且SDN架构200可以支持Kubernetes的定制资源以用于SDN架构200的高级联网和安全性。

[0157] 配置节点230向控制节点232提供REST监视以监视配置资源改变,该控制节点232在计算基础设施内起作用。控制节点232通过监视资源从配置节点230接收配置资源数据,并构建完整的配置图。控制节点232中的给定控制节点消耗与控制节点相关的配置资源数据,并经由到虚拟路由器21的控制平面方面(即,虚拟路由器代理-图1中未示出)的控制接口254将所需的配置分发到计算节点(服务器12)。任何计算节点232可以仅接收作为处理所需的部分图。控制接口254可以是XMPP。部署的配置节点230和控制节点232的数量可以是支持的集群数量的函数。为了支持高可用性,配置平面可以包括2N+1个配置节点230和2N个控制节点232。

[0158] 控制节点232在计算节点之间分配路由。控制节点232使用iBGP在控制节点232之间交换路由,并且控制节点232可以与任何外部BGP支持的网关或其他路由器对等。控制节点232可以使用路由反射器。

[0159] Pod 250和虚拟机252是可由虚拟机编排器240或容器编排器242部署到计算节点并由SDN架构200使用一个或多个虚拟网络互连的工作负载的示例。

[0160] 用户界面244可表示UI 60的示例。如下面更详细描述的,用户界面244可以呈现网络拓扑的图形表示。网络可以包括两个或更多个VN,诸如VN 50A和VN 50N。用户界面244可表示呈现各种界面元件的图形用户界面(并且因此可被称为图形用户界面-GUI-244)的示例,该各种界面元件诸如按钮、滑块、选择器、可拖动界面元件(诸如图形图标)等,通过该各种界面元件定义VNR,诸如VNR 52A,通过该VNR互连两个或更多个VN。

[0161] 图3是根据本公开的技术,更详细地图示了SDN架构200的部件的另一视图的框图。配置节点230、控制节点232和用户界面244被图示为具有它们各自的部件微服务,用于将网络控制器24和SDN架构200实现为云原生SDN架构。可以向计算节点部署部件微服务中的每一个。

[0162] 图3图示了被分成网络控制器24、用户界面244、计算(服务器12)和遥测260特征的单个集群。如上所述,用户界面244可以表示UI 60的示例。配置节点230和控制节点230一起形成网络控制器24。

[0163] 配置节点230可以包括部件微服务API服务器300(或"Kubernetes API服务器300"-对应的控制器406未在图3中示出)、定制API服务器301、定制资源控制器302、以及SDN控制器管理器303(有时被称为"kube-管理器"或"SDN kube-管理器",其中用于网络控制器24的编排平台是Kubernetes)。Contrail-kube-管理器是SDN控制器管理器303的一个示例。配置节点230扩展API服务器300与定制API服务器301接口,以形成支持SDN架构200的数据模型的聚合层。SDN架构200配置意图可以是定制资源,如上所述。

[0164] 控制节点232可以包括部件微服务控制320和核心DNS 322。控制320执行配置分发和路由学习和分发,如以上参考图2所述。

[0165] 计算节点由服务器12表示。每个计算节点包括虚拟路由器代理316和虚拟路由器转发部件(vRouter,v路由器)318。虚拟路由器代理316和vRouter 318中的一个或两个可以是部件微服务。通常,虚拟路由器代理316执行控制相关功能。虚拟路由器代理316从控制节点232接收配置数据,并将该配置数据转换为vRouter 318的转发信息。虚拟路由器代理316还可以执行防火墙规则处理、设置vRouter 318的流、以及与编排插件(用于Kubernetes的CNI和用于Openstack的Nova插件)接口。虚拟路由器代理316在工作负载(pod或VM)被带到计算节点上时生成路由,并且虚拟路由器316与控制节点232交换这样的路由以用于分发到其他计算节点(控制节点232使用BGP在控制节点232之间分发路由)。虚拟路由器代理316还在工作负载终止时撤销路由。vRouter 318可以支持一个或多个转发模式,诸如内核模式、DPDK、SmartNIC卸载等等。在容器架构或虚拟机工作负载的一些示例中,计算节点可以是Kubernetes工作器/工作节点或Openstack nova-计算节点,这取决于使用的特定编排器。

[0166] 一个或多个可选的遥测节点260提供度量、警报、日志和流分析。SDN架构200遥测利用云原生监视服务,诸如Promethyus、Elastic、Fluentd、Kinaba栈(EFK)和Influx TSDB。配置节点230、控制节点232、计算节点、用户界面244和分析节点(未示出)的SDN架构部件微服务可以产生遥测数据。该遥测数据可由遥测节点260的服务消耗。遥测节点260可以为用户展露REST端点,并且可以支持洞察和事件相关性。

[0167] 可选的用户界面244包括web用户界面(UI)306和UI后端308服务。通常,用户界面244为SDN架构部件提供配置、监视、可视化、安全性和故障排除。此外,用户界面244(如上所述,其也可称为GUI 244)可呈现网络拓扑的图形表示。网络可以包括两个或更多个VN,诸如VN 50A和VN 50N。GUI 244可呈现各种界面元件,例如按钮、滑块、选择器、可拖动界面元件(诸如图形图标)等,通过这些界面元件定义VNR,诸如VNR 52A,通过VNR互连两个或更多个VN。

[0168] GUI 244可以接收来自用户的输入,并将输入传递到网络控制器24,其可以处理更新GUI 244的请求。例如,如以下更详细地描述的,GUI 244可更新GUI 244以包括弹出框、子

窗口、附加的框、输入、对话框和其他界面元件,通过它们来提示用户关于VNR 52A的附加配置详细信息。在一些实例中,用户可与网络的拓扑的图形表示对接,以响应于由对GUI 244的更新呈现的提示来选择VN 50A和50N。在任何情况下,GUI 244可使相对没有经验的用户能够以图形和直观的方式配置VNR 52A,而不必指定符合正式配置语言和/或需要路由协议以及其他低级网络概念的广泛知识的复杂配置语句。

[0169] 遥测260、用户界面244、配置节点230、控制节点232和服务器12/计算节点中的每一个可以被认为是SDN架构200节点,因为这些节点中的每一个是实现配置、控制或数据平面的功能性的实体,或者UI和遥测节点的功能性的实体。节点缩放在“建立”期间被配置,并且SDN架构200支持使用编排系统运营商(诸如Kubernetes运营商)的SDN架构200节点的自动缩放。

[0170] 如上所述,SDN架构200配置意图可以是定制资源。一个这样的定制资源可包括VNR 52(图1的示例中所示),通过其以上述方式在两个或更多个VN 50之间建立通信。如上所述,VNR 52可以表示用于配置VNR 50之间的路由信息的引入和引出的策略的逻辑抽象,由此VNR 52可以使用为VNR 52中的每一个建立的公共路由目标来促进路由信息的交换(指的是非对称或对称的引入和引出)。公共路由目标可被定义并与用于实现VN 50的路由实例相关联。

[0171] 管理员(诸如Kubernetes管理员)可与用户界面244(例如,web UI 306)对接,以可能经由具有表示pod、VN 50等的图形元件的图形用户界面来定义VNR 52。为了定义VNR 52,管理员可将VNR 52与分配给VN 50的一个或多个标签相关联。使用这些标签,VNR 52可以建立去往和来自在例示VNR 52时创建的公共路由目标的引入和引出策略。Web UI 306可以与配置控制器230对接以创建公共路由目标,经由控制节点232将公共路由目标安装到一个或多个虚拟路由器318中。Web UI 306还可经由配置控制器230和控制节点232定义用于公共路由目标的路由实例(以将公共路由目标与其他公共路由目标区分),利用该路由域来促进VN 50之间的互连,如将在下面关于多个不同的联网方案更详细地描述的。

[0172] 图4是图示了根据本公开的技术的SDN架构的示例性部件的框图。在该示例中,SDN架构400扩展并使用Kubernetes API服务器用于实现用户对于网络配置的意图的网络配置对象。在Kubernetes术语中,这样的配置对象被称为定制资源,并且当在SDN架构中持久保存时被简单地称为对象。配置对象主要是用户意图(例如,虚拟网络-诸如VN 50、VNR 52、BGPaaS、网络策略、服务链等)。

[0173] SDN架构400配置节点230可以使用Kubernetes API服务器用于配置对象。在kubernetes术语中,这些被称为定制资源。

[0174] Kubernetes提供了向集群添加定制资源的两种方式:

[0175] • 定制资源定义(CRD)是简单的并且可以在没有任何编程的情况下创建。

[0176] • API聚合要求编程但允许对API行为的更多控制,诸如如何存储数据和在API版本之间转换。

[0177] 聚合的API是位于充当代理的主API服务器后面的从属API服务器。这种布置被称为API聚合(API Aggregation,AA)。对于用户,简单地显示Kubernetes API被扩展。CRD允许用户在不添加另一API服务器的情况下创建新类型的资源。不管它们如何安装,新资源被称为定制资源(CR)以将它们与原生Kubernetes资源(例如,Pod)区分开。CRD用于初始Config

原型。该架构可使用API服务器构建器Alpha库来实现聚合的API。API服务器构建器是用于构建原生Kubernetes聚合扩展的库和工具的集合。

[0178] 通常,Kubernetes API中的每个资源需要处理REST请求并管理对象的持久存储的代码。主Kubernetes API服务器300(用API服务器微服务300A至300J实现)处理原生资源,并且还可以通过CRD一般地处理定制资源。聚合的API 402表示扩展Kubernetes API服务器300以允许通过编写和部署定制API服务器301(使用定制API服务器微服务301A至301M)来为定制资源提供专门实施方式的聚合层。主API服务器300将对定制资源的请求委托给定制API服务器301,从而使这些资源对其所有客户端都可用。

[0179] 这样,API服务器300(例如,kube-apiserver)接收Kubernetes配置对象、原生对象(pod、服务)和定制资源。SDN架构400的定制资源可以包括配置对象,当SDN架构400中的配置对象的预期状态被实现时,该配置对象实现SDN架构400的预期网络配置,包括将每个VNR 52实现为一个或多个引入策略和/或一个或多个引出策略以及公共路由目标(和路由实例)。如上所述,在SDN架构400内的VNR 52可导致如下文更详细描述的互连两个或更多个VN 50的引入和/或引出策略。

[0180] 在这方面,定制资源可以对应于传统上为网络配置定义的配置模式,但是根据本公开的技术,定制资源被扩展为可通过聚合的API 402来操作。这种定制资源在本文中可替代地称为“用于SDN架构配置的定制资源”。这些可以包括VN、VNR、bgp-as-a-service(BGPaaS)、子网、虚拟路由器、服务实例、项目、物理接口、逻辑接口、节点、网络ipam、浮动IP、警报、别名IP、访问控制列表、防火墙策略、防火墙规则、网络策略、路由目标、路由实例。用于SDN架构配置的定制资源可以对应于通常由SDN控制器展露的配置对象,但是根据本文所述的技术,配置对象被展露出为定制资源,并且与Kubernetes原生/内置资源一起合并以支持由聚合的API 402展露的统一意图模型,该模型由Kubernetes控制器406A至406N和由定制资源控制器302(在图4中用部件微服务302A至302L示出)实现,该定制资源控制器用于协调包括网络元件的计算基础设施的实际状态与意图状态。

[0181] 给定在展露与Kubernetes原生/内置资源一起合并的定制资源方面的统一性质,Kubernetes管理员(或其他Kubernetes用户)可使用公共Kubernetes语义定义VNR,诸如VNR 52,其然后可被转换成详述路由信息的引入和引出的复杂策略,以促进VN 50的互连,而不需要对BGP和互连VN 50通常所需的其他路由协议的任何理解。因此,本技术的各个方面可以提升更统一的用户体验,这潜在地使得更少的错误配置和试错,这可以改进SDN架构400自身的执行(在利用更少的处理周期、存储器、带宽等以及相关联的功率方面)。

[0182] API服务器300聚合层将API定制资源发送到它们相应的注册的定制API服务器300。可以有多个定制API服务器/定制资源控制器来支持不同类型的定制资源。定制API服务器300处理SDN架构配置的定制资源并写入到可以是etcd的配置存储304。定制API服务器300可以是主机并展露定制资源控制器302可能需要的SDN控制器标识符分配服务。

[0183] 定制资源控制器302开始应用商业逻辑以达到具有用户意图配置的用户意图。业务逻辑被实现为协调环。图8是图示了根据本公开的技术的用于SDN架构配置的定制资源的定制控制器的实例的框图。定制控制器814可以表示定制资源控制器301的示例性实例。在图8所示的示例中,定制控制器814可以与定制资源818相关联。定制资源818可以是针对SDN架构配置的任何定制资源。定制控制器814可以包括协调器816,该协调器816包括用于执行

协调循环的逻辑,其中定制控制器814观察834(例如,监视)定制资源818的当前状态832。响应于确定期望状态836与当前状态832不匹配,协调器816可执行动作以调整838定制资源的状态,使得当前状态832与期望状态836匹配。API服务器300可以接收请求,并且将该请求中继到定制API服务器301,以将定制资源818的当前状态832改变为期望状态836。

[0184] 在API请求301是针对定制资源的创建请求的情况下,协调器816可作用于定制资源的实例数据的创建事件。协调器816可以为请求的定制资源依赖的定制资源创建实例数据。作为示例,边缘节点定制资源可以依赖于虚拟网络定制资源、虚拟接口定制资源和IP地址定制资源。在该示例中,当协调器816接收到边缘节点定制资源的创建事件时,协调器816还可以创建边缘节点定制资源依赖的定制资源,例如虚拟网络定制资源、虚拟接口定制资源和IP地址定制资源。

[0185] 默认地,定制资源控制器302运行主动-被动模式,并且使用主选来实现一致性。当控制器pod开始时,它试图使用指定的关键在Kubernetes中创建ConfigMap资源。如果创建成功,则pod成为主并开始处理协调请求;否则,它会阻止试图在无限循环中创建ConfigMap。

[0186] 定制资源控制器302可以跟踪它创建的定制资源的状态。例如,虚拟网络(Virtual Network,VN)创建路由实例(Routing Instance,RI),该路由实例创建路由目标(Route Target,RT)。如果路由目标的创建失败,则路由实例状态被降级,并且因此虚拟网络状态也被降级。定制资源控制器302因此可以输出指示这些定制资源的状态的定制消息,用于故障排除。同样,VNR创建RI,该RI以与以上关于VN所讨论的方式类似的方式创建RT。图9图示了在对不同的定制资源类型具有依赖性的定制资源类型中的创建、监视和协调的示例性流程。

[0187] 由配置节点230实现的配置平面具有高可用性。配置节点230可以基于Kubernetes,包括Kube-apiserver服务(例如,API服务器300)和存储后端etcd等(例如,配置存储304)。有效地,由配置节点230实现的聚合的API 402作为由控制节点232实现的控制平面的前端运行。API服务器300的主要实施方式是Kube-apiserver,其被设计成通过部署更多实例来水平缩放。如所示,API服务器300的若干实例可运行以使API请求和处理负载均衡。

[0188] 配置存储304可以实现为etcd。Etcd是用作集群数据的Kubernetes后备存储的一致且高度可用的键值存储。

[0189] 在图4的示例中,SDN架构400的服务器12各自包括编排代理420和容器化(或“云原生”)路由协议守护进程324。以下进一步详细描述SDN架构400的这些部件。

[0190] SDN控制器管理器303可以作为Kubernetes核心资源(服务、命名空间、pod、网络策略、网络附件定义)和扩展的SDN架构资源(VirtualNetwork、RoutingInstance等)之间的接口来操作。SDN控制器管理器303监视Kubernetes API以寻找Kubernetes核心和用于SDN架构配置的定制资源两者上的改变,并且因此可以对相关资源执行CRUD操作。

[0191] 在一些示例中,SDN控制器管理器303是一个或多个Kubernetes定制控制器的集合。在一些示例中,在单或多集群部署中,SDN控制器管理器303可以在其管理的Kubernetes集群上运行。

[0192] SDN控制器管理器303针对以下Kubernetes对象监听创建、删除和更新事件:

[0193] • Pod

[0194] • 服务

[0195] • 节点多口

[0196] • 入口

[0197] • 端点

[0198] • 命名空间

[0199] • 部署

[0200] • 网络策略

[0201] 当生成这些事件时,SDN控制器管理器303创建适当的SDN架构对象,其进而被定义为用于SDN架构配置的定制资源。响应于检测到定制资源的实例上的事件,不管是由SDN控制器管理器303例示还是通过定制API服务器301例示,控制节点232获得用于定制资源的实例的配置数据,并配置SDN架构400中的配置对象的对应实例。

[0202] 例如,SDN控制器管理器303监视Pod创建事件,并且作为响应,可以创建以下SDN架构对象:VirtualMachine(工作负载/pod)、VirtualMachineInterface(虚拟网络接口)和InstanceIP(IP地址)。然后,在这种情况下,控制节点232可以在选择的计算节点中例示SDN架构对象。

[0203] 作为示例,基于监视,控制节点232A可检测由客户API服务器301A展露的第一定制资源的实例上的事件,其中第一定制资源用于配置SDN架构系统400的某个方面并且对应于SDN架构系统400的配置对象的类型。例如,配置对象的类型可以是与第一定制资源对应的防火墙规则。响应于该事件,控制节点232A可以获得防火墙规则实例的配置数据(例如,防火墙规则规范)并且在用于服务器12A的虚拟路由器中提供防火墙规则。配置节点230和控制节点232可以利用SDN架构的相应类型的配置对象,诸如虚拟网络、虚拟网络路由器、bgp-as-a-service(BGPaaS)、子网、虚拟路由器、服务实例、项目、物理接口、逻辑接口、节点、网络ipam、浮动ip、警告、别名ip、访问控制列表、防火墙策略、防火墙规则、网络策略、路由目标、路由实例等,对其他定制资源执行类似的操作。

[0204] 图5是根据本公开所述的技术的示例性计算设备的框图。图2的计算设备500可以表示真实或虚拟服务器,并且可以表示任何服务器12的示例性实例,并且可以被称为计算节点、主/工作节点或主机。在该示例中,计算设备500包括耦接计算设备500硬件环境的硬件部件的总线542。总线542将网络接口卡(NIC)530、存储盘546和一个或多个微处理器210(在下文中,“微处理器510”)耦接。NIC 530可以具有SR-IOV能力。在一些情况下,前端总线可以耦接微处理器510和存储器设备524。在一些示例中,总线542可以耦接存储器设备524、微处理器510和NIC 530。总线542可以表示外围部件接口(PCI)express(PCIe)总线。在一些示例中,直接内存访问(DMA)控制器可以控制耦接到总线542的部件之间的DMA传输。在一些示例中,耦接到总线542的部件控制耦接到总线542的部件之间的DMA传输。

[0205] 微处理器510可包括一个或多个处理器,每个处理器包括独立的执行单元以执行符合指令集架构的指令,指令被存储到存储介质。执行单元可以被实现为单独的集成电路(IC),或者可以被组合在一个或多个多核处理器(或“众核”处理器)内,其中每个多核处理器使用单个IC(即,芯片多处理器)来实现。

[0206] 盘546表示计算机可读存储介质,其包括以用于存储诸如处理器可读指令、数据结

构、程序模块或其他数据的信息的任何方法或技术实现的易失性和/或非易失性、可移动和/或不可移动介质。计算机可读存储介质包括但不限于随机存取存储器(RAM)、只读存储器(ROM)、EEPROM、闪存、CD-ROM、数字多功能盘(DVD)或其他光学存储装置、磁带盒、磁带、磁盘存储装置或其他磁性存储设备、或者可用于存储期望信息并且可由微处理器510访问的任何其他介质。

[0207] 主存储器524包括一个或多个计算机可读存储介质,其可以包括随机存取存储器(RAM),诸如各种形式的动态RAM(DRAM),诸如DDR2/DDR3 SDRAM,或者静态RAM(SRAM)、闪存,或者可以用于携带或存储指令或数据结构形式的期望的程序代码和程序数据并且可以由计算机访问的任何其他形式的固定或可移动存储介质。主存储器524提供由可寻址存储器位置组成的物理地址空间。

[0208] 网络接口卡(NIC)530包括被配置为使用底层物理网络的链路来交换数据包的一个或多个接口532。接口532可以包括具有一个或多个网络端口的端口接口卡。NIC 530还可包括卡上存储器,以例如存储数据包数据。NIC 530和耦接到总线542的其他设备之间的直接存储访问传输可以从NIC存储器读取/向NIC存储器写入。

[0209] 存储器524、NIC 530、存储盘546和微处理器510可以为软件栈提供操作环境,该软件栈包括在内核空间中执行的操作系统内核580。内核580可以表示例如Linux、Berkeley软件分布(BSD)、另一Unix变体内核或可从微软公司获得的Windows服务器操作系统内核。在一些实例中,操作系统可以执行管理程序和由管理程序管理的一个或多个虚拟机。示例管理程序包括用于Linux内核的基于内核的虚拟机(KVM)、Xen、可从VMware获得的ESXi、可从Microsoft获得的Windows Hyper-V和其他开放源和专有管理程序。术语管理程序可以包括虚拟机管理器(VMM)。包括内核580的操作系统为用户空间545中的一个或多个进程提供执行环境。

[0210] 内核580包括物理驱动器525以使用网络接口卡530。网络接口卡530还可实现SR-IOV以实现在诸如容器529A或一个或多个虚拟机(图5中未示出)等一个或多个虚拟执行元件之间共享物理网络功能(I/O)。诸如虚拟功能等共享虚拟设备可提供专用资源,以使得每个虚拟执行元件可访问NIC 530的专用资源,因此其对于每个虚拟执行元件表现为专用NIC。虚拟功能可以表示与物理驱动器525所使用的物理功能以及与其他虚拟功能共享物理资源的轻量PCIe功能。对于支持SR-IOV的NIC 530,NIC 530可以具有根据SR-IOV标准的数千个可用的虚拟功能,但是对于I/O密集型应用,所配置的虚拟功能的数量通常小得多。

[0211] 计算设备500可以耦接到物理网络交换结构,该物理网络交换结构包括将交换结构从物理交换机扩展到耦接到交换结构的物理服务器的软件或“虚拟”路由器的覆盖网络,包括虚拟路由器506。虚拟路由器可以是由物理服务器(例如,图1的服务器12)执行的进程或线程或其部件,其动态地创建和管理可用于虚拟网络端点之间的通信的一个或多个虚拟网络。在一个示例中,虚拟路由器使用覆盖网络来实现每个虚拟网络,其提供将端点的虚拟地址从该端点正在其上执行的服务器的物理地址(例如,IP地址)解耦的能力。

[0212] 每个虚拟网络可以使用其自己的寻址和安全方案,并且可以被视为与物理网络及其寻址方案正交。可以使用各种技术来通过物理网络在虚拟网络内和跨虚拟网络传输数据包。本文使用的术语“虚拟路由器”可以包括OpenvSwitch(OVS)、OVS网桥、Linux网桥、Docker网桥或位于主机设备上并且在一个或多个虚拟网络的虚拟网络端点之间执行交换、

桥接或路由数据包的其他设备和/或软件,其中虚拟网络端点由一个或多个服务器12托管。在图5的示例性计算设备500中,虚拟路由器506在用户空间内作为基于DPDK的虚拟路由器执行,但是在各种实现中,虚拟路由器506可以在管理程序、主机操作系统、主机应用或虚拟机内执行。

[0213] 虚拟路由器506可以替换和包含通常用于pod 502的Kubernetes部署的Linux网桥/OVS模块的虚拟路由/桥接功能。虚拟路由器506可以为虚拟网络执行桥接(例如,E-VPN)和路由(例如,L3VPN、IP-VPN)。虚拟路由器506可以执行联网服务,诸如应用安全策略、NAT、多播、镜像和负载均衡。

[0214] 虚拟路由器506可以作为内核模块或作为用户空间DPDK进程来执行(本文在用户空间545中示出了虚拟路由器506)。虚拟路由器代理514也可在用户空间中执行。在示例性计算设备500中,虚拟路由器506在用户空间内作为基于DPDK的虚拟路由器执行,但是在各种实现中,虚拟路由器506可以在管理程序、主机操作系统、主机应用或虚拟机内执行。虚拟路由器代理514具有使用信道到网络控制器24的连接,其用于下载配置和转发信息。虚拟路由器代理514将该转发状态编程到由虚拟路由器506表示的虚拟路由器数据(或"转发")平面。虚拟路由器506和虚拟路由器代理514可以是进程。虚拟路由器506和虚拟路由器代理514是容器化的/云原生的。

[0215] 虚拟路由器506可以替换和包含通常用于pod 502的Kubernetes部署的Linux网桥/OVS模块的虚拟路由/桥接功能。虚拟路由器506可以为虚拟网络执行桥接(例如,E-VPN)和路由(例如,L3VPN、IP-VPN)。虚拟路由器506可以执行联网服务,诸如应用安全策略、NAT、多播、镜像和负载均衡。

[0216] 虚拟路由器506可以是多线程的,并在一个或多个处理器核上执行。虚拟路由器506可以包括多个队列。虚拟路由器506可以实现数据包处理流水线。虚拟路由器代理514可以根据要应用于数据包的操作,以从最简单到最复杂的方式来拼接流水线。虚拟路由器506可以维护多个转发基础实例。虚拟路由器506可以使用读取复制更新(Read Copy Update,RCU)锁来访问和更新表。

[0217] 为了向其他计算节点或交换机发送数据包,虚拟路由器506使用一个或多个物理接口532。通常,虚拟路由器506与诸如VM或pod 502之类的工作负载交换覆盖数据包。虚拟路由器506具有多个虚拟网络接口(例如,vif)。这些接口可以包括用于与主机操作系统交换数据包的内核接口vhost0;与虚拟路由器代理514的接口,pkt0,用于从网络控制器获得转发状态并发送上去异常数据包。可以存在与一个或多个物理网络接口532相对应的一个或多个虚拟网络接口。虚拟路由器506的其他虚拟网络接口用于与工作负载交换数据包。

[0218] 在虚拟路由器506(未示出)的基于内核的部署中,虚拟路由器506被安装为操作系统内部的内核模块。虚拟路由器506向TCP/IP栈注册其自身,以从其想要的任何期望的操作系统接口接收数据包。接口可以是捆绑、物理、tap(用于VM)、veth(用于容器)等。在该模式中,虚拟路由器506依赖于操作系统来发送和接收来自不同接口的数据包。例如,操作系统可以展露由vhost-net驱动器支持的tap接口以与VM通信。一旦虚拟路由器506注册了来自该tap接口的数据包,TCP/IP栈就向其发送所有数据包。虚拟路由器506经由操作系统接口发送数据包。此外,NIC队列(物理的或虚拟的)由操作系统处理。数据包处理可以在中断模式下操作,这生成中断并且可能使得频繁的上下文切换。当存在高数据包速率时,伴随频繁

中断和上下文切换的开销可能淹没操作系统并导致差的性能。

[0219] 在虚拟路由器506的基于DPDK的部署中(图5中示出),虚拟路由器506被安装为链接到DPDK库的用户空间545应用。这可以导致比基于内核的部署更快的性能,特别是在存在高数据包速率的情况下。物理接口532由DPDK的轮询模式驱动器(PMD)而不是内核的基于中断的驱动器使用。物理接口532的寄存器可被展露到用户空间545,以便PMD可访问;以此类方式绑定的物理接口532不再由主机操作系统管理或对主机操作系统可见,并且基于DPDK的虚拟路由器506管理物理接口532。这包括数据包轮询、数据包处理和数据包转发。换句话说,用户数据包处理步骤由虚拟路由器506DPDK数据平面执行。当数据包速率高时,这种"轮询模式"的本质使得虚拟路由器506DPDK数据平面数据包处理/转发与中断模式相比更高效。与内核模式的虚拟路由器506相比,在数据包I/O期间存在相对较少的中断和上下文切换,并且在某些情况下可以完全避免在数据包I/O期间的中断和上下文切换。

[0220] 通常,每个pod 502A至502B可以被分配一个或多个虚拟网络地址以在相应的虚拟网络内使用,其中每个虚拟网络可以与由虚拟路由器506提供的一不同虚拟子网相关联。pod 502B可以被分配其自己的虚拟层三(L3)IP地址,例如用于发送和接收通信,但是可能不知道pod 502B在其上执行的计算设备500的IP地址。虚拟网络地址因此可以不同于底层物理计算机系统(例如,计算设备500)的逻辑地址。

[0221] 计算设备500包括虚拟路由器代理514,其控制计算设备500的虚拟网络的覆盖,并且协调计算设备500内的数据数据包的路由。通常,虚拟路由器代理514与虚拟化基础设施的网络控制器24通信,其生成命令以创建虚拟网络并配置网络虚拟化端点,诸如计算设备500,并且更具体地,虚拟路由器506,以及虚拟网络接口212。通过基于从网络控制器24接收到的信息来配置虚拟路由器506,虚拟路由器代理514可以支持配置网络隔离、基于策略的安全、网关、源网络地址转换(SNAT)、负载均衡器以及用于编排的服务链能力。

[0222] 在一个示例中,由虚拟网络域内的容器529A至529B生成或消费的网络数据包,诸如层三(L3)IP数据包或层二(L2)以太网数据包,可以被封装在由物理网络传送的另一个数据包(例如,另一个IP或以太网数据包)中。在虚拟网络中传输的数据包在本文可以被称为"内数据包",而物理网络数据包在本文可以被称为"外数据包"或"隧道数据包"。虚拟网络数据包在物理网络数据包内的封装和/或解封装可以由虚拟路由器506执行。该功能在本文被称为隧道技术传送,并且可以用于创建一个或多个覆盖网络。除了IPinIP,可以使用的其他示例隧道技术协议包括通用路由封装(GRE)上的IP、VxLAN、GRE上的多协议标签交换(MPLS)、用户数据报上的MPLS协议(UDP)等。虚拟路由器506对源于/去往pod 502的任何容器的数据包执行隧道封装/解封装,并且虚拟路由器506经由总线542和/或NIC 530的网桥与pod 502交换数据包。

[0223] 如上所述,网络控制器24可以提供逻辑集中控制器,以便于一个或多个虚拟网络的操作。网络控制器24可以例如维护路由信息库,诸如存储物理网络以及一个或多个覆盖网络的路由信息的一个或多个路由表。虚拟路由器506实现诸如VRF 222A的一个或多个虚拟路由和转发实例(VRF),用于虚拟路由器506作为相应隧道端点操作的相应虚拟网络。通常,每个VRF存储用于相应虚拟网络的转发信息,并且识别数据包将被转发到何处以及数据包是否将被封装在隧道技术协议中,诸如利用可以包括用于虚拟网络协议栈的不同层的一个或多个报头的隧道技术报头。每个VRF可以包括存储虚拟网络的路由和转发信息的网络

转发表。

[0224] NIC 530可接收隧道数据包。虚拟路由器506处理隧道数据包,以根据隧道封装报头确定内数据包的源和目的地端点的虚拟网络。虚拟路由器506可以剥离层2报头和隧道封装报头,以在内部仅转发内数据包。隧道封装报头可以包括虚拟网络标识符,诸如VxLAN标记或MPLS标签,其指示虚拟网络,例如与VRF 222A相对应的虚拟网络。VRF 222A可包括用于内数据包的转发信息。例如,VRF 222A可将内数据包的目的地层3地址映射到虚拟网络接口212。作为响应,VRF 222A经由虚拟网络接口212将内数据包转发到pod 502A。

[0225] 容器529A还可以将内数据包作为源虚拟网络端点。容器529A例如可生成以由另一计算设备(即,非计算设备500)执行的目的地虚拟网络端点或容器中的另一个为目的地的层3内数据包。容器529A可以经由附接到VRF 222A的虚拟网络接口将层3内数据包发送到虚拟路由器506。

[0226] 虚拟路由器506接收内数据包和层2报头,并为内数据包确定虚拟网络。虚拟路由器506可以使用上述虚拟网络接口实现技术(例如,macvlan、veth等)中的任何一个来确定虚拟网络。虚拟路由器506使用与用于内数据包的虚拟网络相对应的VRF 222A来生成用于内数据包的外报头,该外报头包括用于覆盖隧道的外IP报头和标识虚拟网络的隧道封装报头。虚拟路由器506封装具有外报头的内数据包。虚拟路由器506可以封装具有新的层2报头的隧道数据包,该报头具有与计算设备500外部的设备(例如,TOR交换机16或服务器12中的一个)相关联的目的层2地址。如果在计算设备500外部,则虚拟路由器506使用物理功能221将具有新的层2报头的隧道数据包输出到NIC 530。NIC 530在出站接口上输出数据包。如果目的地是在计算设备500上执行的另一虚拟网络端点,则虚拟路由器506将数据包路由到虚拟网络接口212、213中的适当的一个。

[0227] 在一些示例中,计算设备500的控制器(例如,图1的网络控制器24)配置每个pod 502中的默认路由,以使虚拟机224使用虚拟路由器506作为出站数据包的初始下一跳。在一些示例中,NIC 530配置有一个或多个转发规则以使得从虚拟机224接收的所有数据包被交换到虚拟路由器506。

[0228] Pod 502A包括一个或多个应用容器529A。Pod 502B包括容器化路由协议守护进程(containerized routing protocol daemon,cRPD)560的实例。容器平台588包括容器运行时间590、编排代理592、服务代理593和CNI 570。

[0229] 容器引擎590包括可由微处理器510执行的代码。容器运行时间590可以是一个或多个计算机进程。容器引擎590运行容器529A至529B形式的容器化的应用。容器引擎590可以表示Dockert、rkt或用于管理容器的其他容器引擎。通常,容器引擎590接收请求并管理诸如图像、容器、网络和卷之类的对象。图像是具有用于创建容器的指令的模板。容器是图像的可执行实例。基于来自控制器代理592的指示,容器引擎590可获得图像并将它们例示为pod 502A至502B中的可执行容器。

[0230] 服务代理593包括可由微处理器510执行的代码。服务代理593可以是一个或多个计算机进程。服务代理593监视服务和端点对象的添加和移除,并且它维护计算设备500的网络配置以确保pod和容器之间的通信,诸如使用服务。服务代理593还可以管理ip表以捕获到服务的虚拟IP地址和端口的流量,并且将流量重定向到代理备份pod的代理端口。服务代理593可以表示Kubernetes集群的工作节点的kube-代理。在一些示例中,容器平台588不

包括服务代理593,或者服务代理593被禁用以有利于CNI 570对虚拟路由器506和pod 502的配置。

[0231] 编排代理592包括可由微处理器510执行的代码。编排代理592可以是一个或多个计算机进程。编排代理592可以表示Kubernetes集群的工作节点的kubelet。编排代理592是编排器(例如,图1的编排器23)的代理,其接收容器的容器规范数据并确保计算设备500执行容器。容器规范数据可以是从编排器23发送到编排代理592,或者是经由命令行界面、HTTP端点或HTTP服务器间接接收的清单文件的形式。容器规范数据可以是容器的pod 502中的一个的容器规范(例如,PodSpec-描述pod的又一标记语言(Yet Another Markup Language,YAML)或JSON对象)。基于容器规范数据,编排代理592指导容器引擎590获得并例示容器529的容器图像,以便由计算设备500执行容器529。

[0232] 编排代理592例示或以其他方式调用CNI 570以为每个pod 502配置一个或多个虚拟网络接口。例如,编排代理592接收用于pod 502A的容器规范数据,并且指导容器引擎590基于用于pod 502A的容器规范数据创建具有容器529A的pod 502A。编排代理592还调用CNI 570来为pod 502A配置对应于VRF 222A的虚拟网络的虚拟网络接口。在该示例中,pod 502A是对应于VRF 222A的虚拟网络的虚拟网络端点。

[0233] CNI 570可以获得用于配置pod 502的虚拟网络接口的接口配置数据。虚拟路由器代理514作为虚拟网络控制平面模块运行,用于使网络控制器24能够配置虚拟路由器506。与管理虚拟执行元件的提供、调度和管理的编排控制平面(包括用于工作节点的容器平台588和主节点,诸如编排器23)不同,虚拟网络控制平面(包括用于工作节点的网络控制器24和虚拟路由器代理514)管理部分地由工作节点的虚拟路由器506在数据平面中实现的虚拟网络的配置。虚拟路由器代理514向CNI 570通信用于虚拟网络接口的接口配置数据,以使得编排控制平面元件(即,CNI 570)能够根据网络控制器24确定的配置状态来配置虚拟网络接口,从而桥接编排控制平面与虚拟网络控制平面之间的间隙。另外,这可以使得CNI 570能够获得pod的多个虚拟网络接口的接口配置数据,并且配置多个虚拟网络接口,这可以减少调用单独的CNI 570来配置每个虚拟网络接口所固有的通信和资源开销。

[0234] 在2022年2月1日提交的美国申请第17/649,632号中描述了容器化路由协议守护进程,其通过引用整体并入本文。

[0235] 此外,CNI 570(可能结合虚拟路由器代理514)可以配置虚拟路由器506以实现VNR 52。VNR 52可导致经由用于与VNR 52的公共路由目标交换路由信息的一个或多个引入策略和/或一个或多个引出策略来配置路由平面。这些策略导致为VN 50维护的路由信息与VNR 52公共路由目标交换(或者换句话说,在其间泄漏),其进而被解析为转发信息。CNI 570可以从网络控制器24获得用于安装该转发信息的配置数据,其与虚拟路由器代理514接口以安装用于转发来自VN 50的数据包的转发信息。

[0236] 换言之,创建该公共路由目标可以使得能够将路由信息从VN 50中的一个(例如,VN 50A)引入和引出到通过VNR 52中的一个或多个提供的公共路由目标,以及从公共路由目标引入和引出到VN 50中的另一个(例如,VN 50N)。网络控制器24可以将该路由信息解析为上述转发信息,并将其安装在虚拟路由器506内,以实现VN 50A和VN 50N之间的数据包转发(在一些配置中,诸如上述网格配置)。以这种方式,虚拟路由器506可以建立用于在VN 50之间引入和引出路由信息的方式,然后VN 50可以使用其来在VN 50中的每一个其他VN之间

传送数据包。

[0237] 图6是根据本公开的技术的操作为SDN架构系统的一个或多个集群的计算节点的示例性计算设备的框图。计算设备1300可以表示一个或多个真实或虚拟服务器。在一些示例中,计算设备1300可以实现用于相应集群或用于多个集群的一个或多个主节点。

[0238] 调度器1322、API服务器300A、控制器406A、定制API服务器301A、定制资源控制器302A、控制器管理器1326、SDN控制器管理器1325、控制节点232A和配置存储1328虽然被图示和描述为由单个计算设备1300执行,但是可以被分布在构成计算系统或硬件/服务器集群的多个计算设备之间。换句话说,多个计算设备中的每一个可以为调度器1322、API服务器300A、控制器406A、定制API服务器301A、定制资源控制器302A、网络控制器管理器1326、网络控制器1324、SDN控制器管理器1325、控制节点232A或配置存储1328中的任何一个或多个的一个或多个实例提供硬件操作环境。

[0239] 在该示例中,计算设备1300包括耦接计算设备1300硬件环境的硬件部件的总线1342。总线1342将网络接口卡(NIC)1330、存储盘1346和一个或多个微处理器1310(在下文中,“微处理器1310”)耦接。在一些情况下,前端总线可以耦接微处理器1310和存储器设备1344。在一些示例中,总线1342可以耦接存储器设备1344、微处理器1310和NIC 1330。总线1342可以表示外围部件接口(PCI)express(PCIe)总线。在一些示例中,直接内存访问(DMA)控制器可以控制耦接到总线242的部件之间的DMA传输。在一些示例中,耦接到总线1342的部件控制耦接到总线1342的部件之间的DMA传输。

[0240] 微处理器1310可包括一个或多个处理器,每个处理器包括独立的执行单元以执行符合指令集架构的指令,指令被存储到存储介质。执行单元可以被实现为单独的集成电路(IC),或者可以被组合在一个或多个多核处理器(或“众核”处理器)内,其中每个多核处理器使用单个IC(即,芯片多处理器)来实现。

[0241] 盘1346表示计算机可读存储介质,其包括以用于存储诸如处理器可读指令、数据结构、程序模块或其他数据的信息的任何方法或技术实现的易失性和/或非易失性、可移动和/或不可移动介质。计算机可读存储介质包括但不限于随机存取存储器(RAM)、只读存储器(ROM)、EEPROM、闪存、CD-ROM、数字多功能盘(DVD)或其他光学存储装置、磁带盒、磁带、磁盘存储装置或其他磁性存储设备、或者可用于存储期望信息并且可由微处理器1310访问的任何其他介质。

[0242] 主存储器1344包括一个或多个计算机可读存储介质,其可以包括随机存取存储器(RAM),诸如各种形式的动态RAM(DRAM),诸如DDR2/DDR3 SDRAM,或者静态RAM(SRAM)、闪存,或者可以用于携带或存储指令或数据结构形式的期望的程序代码和程序数据并且可以由计算机访问的任何其他形式的固定或可移动存储介质。主存储器1344提供由可寻址存储器位置组成的物理地址空间。

[0243] 网络接口卡(NIC)1330包括被配置为使用底层物理网络的链路来交换数据包的一个或多个接口3132。接口3132可以包括具有一个或多个网络端口的端口接口卡。NIC 1330还可包括卡上存储器,以例如存储数据包数据。NIC 1330和耦接到总线1342的其他设备之间的直接内存访问传输可以从NIC存储器读取/向NIC存储器写入。

[0244] 存储器1344、NIC 1330、存储盘1346和微处理器1310可以为软件栈提供操作环境,该软件栈包括在内核空间中执行的操作系统内核1314。内核1314可以表示例如Linux、

Berkeley软件分布(BSD)、另一Unix变体内核或可从微软公司获得的Windows服务器操作系统内核。在一些实例中,操作系统可以执行管理程序和由管理程序管理的一个或多个虚拟机。示例管理程序包括用于Linux内核的基于内核的虚拟机(KVM)、Xen、可从VMware获得的ESXi、可从Microsoft获得的Windows Hyper-V和其他开放源和专有管理程序。术语管理程序可以包括虚拟机管理器(VMM)。包括内核1314的操作系统为用户空间1345中的一个或多个进程提供执行环境。内核1314包括物理驱动器1327以使用网络接口卡230。

[0245] 计算设备1300可以耦接到物理网络交换结构,该物理网络交换结构包括将交换结构从物理交换机扩展到耦接到交换结构的物理服务器的软件或虚拟路由器(诸如虚拟路由器21)的覆盖网络。计算设备1300可以使用一个或多个专用虚拟网络来配置集群的工作节点。

[0246] API服务器300A、调度器1322、控制器406A、定制API服务器301A、定制资源控制器302A、控制器管理器1326和配置存储1328可以实现集群的主节点,并且可以可替代地被称为“主部件”。该集群可以是Kubernetes集群,并且主节点可以是Kubernetes主节点,在该情况下,主部件是Kubernetes主部件。

[0247] API服务器300A、控制器406A、定制API服务器301A和定制资源控制器302A中的每一个都包括可由微处理器1310执行的代码。定制API服务器301A验证并配置用于SDN架构配置(诸如VN 50和VNR 52)的定制资源的数据。服务可以是定义pod的逻辑集合和用于访问pod的策略的抽象。基于服务定义来选择实现服务的pod集合。服务可以部分地实现为负载均衡器,或者以其他方式包括负载均衡器。API服务器300A和定制API服务器301A可以实现代表性状态传输(Representational State Transfer,REST)接口以处理REST操作,并且作为SDN架构的配置平面的一部分,将前端提供给存储到配置存储1328的对应集群的共享状态。API服务器300A可以表示Kubernetes API服务器。

[0248] 配置存储1328是所有集群数据的后备存储。集群数据可以包括集群状态和配置数据。配置数据还可以提供用于服务发现的后端和/或提供锁定服务。配置存储1328可以实现为键值存储。配置存储1328可以是中央数据库或者分布式数据库。配置存储1328可以代表etcd存储。配置存储1328可以表示Kubernetes配置存储。

[0249] 调度器1322包括可由微处理器1310执行的代码。调度器1322可以是一个或多个计算机进程。调度器1322监视新创建或请求的虚拟执行元件(例如,容器的pod)并且选择虚拟执行元件将在其上运行的工作节点。调度器1322可以基于资源要求、硬件约束、软件约束、策略约束、位置等来选择工作节点。调度器1322可以表示Kubernetes调度器。

[0250] 通常,API服务器1320可以调用调度器1322来调度pod。调度器1322可以选择工作节点,并将选择的工作节点的标识符返回给API服务器1320,该API服务器可以将该标识符与pod相关联地写入配置存储1328。API服务器1320可调用所选择的工作节点的编排代理310,这可使选择的工作节点的容器引擎208从存储服务器获得pod并在工作节点上创建虚拟执行单元。用于选择的工作节点的编排代理310可将pod的状态更新到API服务器1320,该API服务器将此新状态持久存储到配置存储1328。这样,计算设备1300例示了计算基础结构8中的新pod。

[0251] 控制器管理器1326包括可由微处理器1310执行的代码。控制器管理器1326可以是一个或多个计算机进程。控制器管理器1326可以嵌入核心控制回路,通过从API服务器1320

获得通知来监视集群的共享状态。控制器管理器1326可以尝试将集群的状态移向期望状态。示例性控制器406A和定制资源控制器302A可由控制器管理器1326管理。其他控制器可以包括复制控制器、端点控制器、命名空间控制器和服务帐户控制器。控制器管理器1326可以执行生命周期功能,诸如命名空间创建和生命周期、事件垃圾收集、终止的pod垃圾收集、级联删除垃圾收集、节点垃圾收集等。控制器管理器1326可以表示用于Kubernetes集群的Kubernetes控制器管理器。

[0252] 本文所述的SDN架构的网络控制器可以为在网络基础设施上运行的计算架构提供云联网。云联网可以包括用于企业或服务提供商的私有云、基础设施即服务(IaaS)和用于云服务提供商(CSP)的虚拟私有云(VPC)。私有云、VPC和IaaS用例可以涉及多租户虚拟化数据中心,诸如关于图1所描述的。在此类情况下,数据中心中的多个租户共享相同的物理资源(物理服务器、物理存储、物理网络)。每个租户被分配其自己的逻辑资源(虚拟机、容器或其他形式的虚拟执行元件;虚拟存储;虚拟网络)。除非安全策略特别允许,否则这些逻辑资源彼此隔离。数据中心中的虚拟网络也可以互连到物理IP VPN或L2 VPN。

[0253] 网络控制器(或"网络控制器")可以向诸如商业边缘网络、宽带订户管理边缘网络和移动边缘网络的网络提供网络功能虚拟化(NFV)。NFV涉及在虚拟机、容器或其他虚拟执行元件中而不是在物理硬件设备上编排和管理联网功能,诸如防火墙、入侵检测或预防系统(IDS/IPS)、深度数据包检查(DPI)、高速缓存、广域网(WAN)优化等。

[0254] SDN控制器管理器1325包括可由微处理器1310执行的代码。SDN控制器管理器1325可以是一个或多个计算机进程。SDN控制器管理器1325用作面向编排的元件(例如,调度器1322、API服务器300A和定制API服务器301A、控制器管理器1326和配置存储1328)之间的接口。通常,SDN控制器管理器1325针对新的Kubernetes原生对象(例如,pod和服务)监视集群。SDN控制器管理器1325可以隔离虚拟网络中的pod,并且将pod与服务连接以及使用所谓的虚拟网络路由器(其不应与虚拟路由器混淆,该虚拟路由器以对上述公共路由目标的各种引入和引出策略的形式实现虚拟网络路由器以促进虚拟网络之间的互连)来互连虚拟网络。

[0255] SDN控制器管理器1325可以作为集群的主节点的容器来执行。在一些情况下,使用SDN控制器管理器1325使得能够禁用工作节点的服务代理(例如,Kubernetes kube代理),使得使用虚拟路由器来实现所有pod连通性,如本文所述。

[0256] 网络控制器24的部件可以作为Kubernetes的CNI操作,并且可以支持多种部署模式。CNI 17、CNI750是用于管理Kubernetes的联网的该整体CNI架构的计算节点接口。部署模式可分为两类:(1)作为集成到工作负载Kubernetes集群中的CNI的SDN架构集群,以及(2)作为与工作负载Kubernetes集群分离的CNI的SDN架构集群。

[0257] 与工作负载Kubernetes集群集成

[0258] 网络控制器24的部件(例如,定制API服务器301、定制资源控制器302、SDN控制器管理器1325和控制节点232)在靠近Kubernetes控制器部件的主节点上的受管理的Kubernetes集群中运行。在这种模式下,网络控制器24的部件实际上是与工作负载相同的Kubernetes集群的一部分。

[0259] 与工作负载Kubernetes集群分开

[0260] 网络控制器24的部件将由与工作负载Kubernetes集群分开的Kubernetes集群执

行。

[0261] SDN控制器管理器1325可以使用用于编排平台的控制器框架来监听(或以其他方式监视)在Kubernetes原生API中定义的对象中的改变,并且向这些对象中的一些添加注释。注释可以是指定对象的属性的标签或其他标识符(例如,“虚拟网络绿色”)。SDN控制器管理器1325是SDN架构的部件,其监听Kubernetes核心资源(诸如Pod、NetworkPolicy、Service等)事件,并根据需要将这些事件转换为用于SDN架构配置的定制资源。CNI插件(例如,CNI 17、570)是支持Kubernetes联网插件标准的SDN架构部件:容器网络接口。

[0262] SDN控制器管理器1325可以使用由聚合的API 402展露的REST接口来创建用于应用的网络解决方案,以定义诸如虚拟网络、虚拟网络接口和访问控制策略之类的网络对象。网络控制器24部件可以通过例如配置虚拟路由器中的一个或多个虚拟网络和虚拟网络接口来实现计算基础设施中的网络解决方案。(这仅仅是SDN配置的一个示例。)

[0263] 用于该应用的以下示例部署配置包括pod和用于该pod的虚拟网络信息:

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-net-pod
  annotations:
    networks: '[
      { "name": "red-network" },
      { "name": "blue-network" },
      { "name": "default/extns-network" }
    ]'
```

[0264]

```
spec:
  containers:
  - image: busybox
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
    name: busybox
    stdin: true
    tty: true
restartPolicy: Always
```

[0265] 该元数据信息可以被复制到由控制器管理器1326创建的每个pod副本。当SDN控制器管理器1325被通知这些pod时,SDN控制器管理器1325可以创建如注释中列出的虚拟网络(以上示例中的“red-network”、“blue-network”和“default/extns-network”),并且针对

每个虚拟网络创建具有来自虚拟网络的集群范围地址块(例如,10.0/16)的唯一私有虚拟网络地址的每pod副本(例如,pod 202A)虚拟网络接口。

[0266] Contrail是示例性网络控制器架构。Contrail CNI可以是为Contrail开发的CNI。云原生Contrail控制器可以是本公开所述的网络控制器(诸如网络控制器24)的示例。

[0267] 图7A是图示了根据本公开的技术的用于使用SDN架构的底层网络和覆盖网络配置的控制/路由平面的框图。图7B是图示了根据本公开的技术的使用底层网络中配置的隧道来连接pod的经配置的虚拟网络的框图。

[0268] 用于SDN架构的网络控制器24可以使用分布式或集中式路由平面架构。SDN架构可以使用容器化路由协议守护进程(进程)。

[0269] 从网络信令的角度来看,路由平面可以根据分布式模型工作,其中cRPD在集群中的每个计算节点上运行。这本质上意味着智能被构建到计算节点中并且涉及每个节点处的复杂配置。该模型中的路由反射器(RR)可以不进行智能路由决策,而是用作中继来反映节点之间的路由。分布式容器路由协议守护进程(cRPD)是可以使用的路由协议进程,其中每个计算节点运行其自己的路由守护进程实例。同时,集中式cRPD主实例可以充当RR以在计算节点之间中继路由信息。路由和配置智能分布在中心位置具有RR的节点上。

[0270] 可替代地,路由平面可以根据更集中的模型工作,其中网络控制器的部件集中运行并吸收处理配置信息、构造网络拓扑以及将转发平面编程到虚拟路由器中所需的智能。虚拟路由器代理是处理由网络控制器编程的信息的本地代理。这种设计有助于在计算节点处所需的更有限的智能,并且倾向于使得更简单的配置状态。

[0271] 集中控制平面提供以下内容:

[0272] • 允许代理路由框架更简单和更轻。BGP的复杂性和限制对于代理是隐藏的。代理不需要理解例如路由-区分器、路由-目标等概念。代理只交换前缀并相应地建立其转发信息。

[0273] • 控制节点可以做的不仅仅是路由。它们建立在虚拟网络概念上,并且可以使用路由复制和重新源起(例如,以支持例如服务链和VN间路由的特征,以及其他使用情况)来生成新的路由。

[0274] • 建立用于最佳广播和多播转发的BUM树。

[0275] 注意,控制平面对于某些方面具有分布式特性。作为支持分布式功能的控制平面,它允许每个本地虚拟路由器代理发布其本地路由,并基于需要知道的基础来预订配置。

[0276] 于是,从工具POV考虑控制平面设计并且在它们最适合的地方适当地使用手头的工具是有意义的。考虑contrail-bgp和cRPD的利弊集合。

[0277] 以下功能可以由cRPD或网络控制器24的控制节点提供。

[0278] 路由守护进程/处理

[0279] 控制节点和cRPD都可以充当实现不同协议并具有在转发平面中编程路由信息的能力的路由守护进程。

[0280] cRPD实现具有丰富路由栈的路由协议,该丰富路由栈包括内部网关协议(IGP)(例如,中间系统到中间系统(IS-IS))、BGP-LU、BGP-CT、SR-MPLS/SRv6、双向转发检测(BFD)、路径计算元件协议(PCEP)等。它也可以被部署来提供仅控制平面的服务,诸如路由反射器,并且由于这些能力而在因特网路由用例中是流行的。

[0281] 控制节点232也实现路由协议,但主要基于BGP。控制节点232理解覆盖联网。控制节点232在覆盖虚拟化中提供丰富的特征集合,并且满足SDN用例。诸如虚拟化(使用虚拟网络的抽象)和服务链的覆盖特征在telco和云提供商中非常流行。cRPD在某些情况下可以不包括对这种覆盖功能的支持。然而,CRPD的丰富特征集为底层网络提供了强有力的支持。

[0282] 网络编排/自动化

[0283] 路由功能只是控制节点232的一部分。覆盖联网的整体部分是编排。除了提供覆盖路由之外,控制节点232还有助于对编排功能进行建模并提供网络自动化。控制节点232的编排能力的中心是使用基于虚拟网络(和相关对象)的抽象(包括上述VNR)来对网络虚拟化进行建模的能力。控制节点232与配置节点230接口,以将配置信息中继到控制平面和数据平面。控制节点232还帮助建立多播层2和层3的覆盖树。例如,控制节点可以建立它用来实现这一点的集群的虚拟拓扑。cRPD通常不包括这种编排能力。

[0284] 高可用性和水平可缩放性

[0285] 控制节点设计更集中,而cRPD更分散。存在运行在每个计算节点上的cRPD工作器节点。另一方面,控制节点232不运行在计算上,并且甚至可以运行在远程集群(即,与工作负载集群分离并且在一些情况下地理上远离工作负载集群)上。控制节点232还为HA提供水平可伸缩性并在活动-活动模式下运行。计算负载在控制节点232之间共享。另一方面,cRPD通常不提供水平可伸缩性。控制节点232和cRPD都可以向HA提供平滑重启,并且可以允许无头模式下的数据平面操作-其中,即使控制平面重启,虚拟路由器也可以运行。

[0286] 控制平面应该不仅仅是路由守护进程。它应当支持覆盖路由和网络编排/自动化,而cRPD作为管理底层路由的路由协议表现良好。然而,cRPD通常缺乏网络编排能力,并且不能为覆盖路由提供强有力的支持。

[0287] 因此,在一些示例中,SDN架构可以在计算节点上具有cRPD,如图7A至7B所示。图7A图示了SDN架构700,其可以表示示例性实施方式SDN架构200或400。在SDN架构700中,cRPD 324在计算节点上运行,并且在运行提供编排和覆盖服务的控制节点232的集中(并且水平可扩展)集合的同时向转发平面提供底层路由。在一些示例中,可以使用默认网关,而不是在计算节点上运行cRPD 324。

[0288] 计算节点上的cRPD 324通过使用接口540(其可以是gRPC接口)与虚拟路由器代理514交互来提供到转发平面的丰富底层路由。虚拟路由器代理接口可以允许对路由进行编程、配置用于覆盖的虚拟网络接口、以及以其他方式配置虚拟路由器506。这在美国申请第17/649,632号中有更详细的描述。同时,一个或多个控制节点232作为提供覆盖服务的单独的pod运行。SDN架构700因此可以获得由控制节点232提供的丰富的覆盖和编排以及由cRPD 324在计算节点上进行的现代底层路由以补充控制节点232。单独的cRPD控制器720可用于配置cRPD 324。cRPD控制器720可以是设备/元件管理系统、网络管理系统、编排器、用户界面/CLI或其他控制器。cRPD 324运行路由协议并与包括其他cRPD 324的路由器交换路由协议消息。每个cRPD 324可以是容器化的路由协议过程,并且有效地作为路由器控制平面的仅软件版本来操作。

[0289] 由cRPD 324提供的增强的底层路由可替换转发平面处的默认网关,并为可支持的用例提供丰富的路由栈。在不使用cRPD 324的一些示例中,虚拟路由器506将依赖于用于底层路由的默认网关。在一些示例中,作为底层路由过程的cRPD 324将被限制为仅对具有控

制平面路由信息的默认inet(6).0结构进行编程。在这样的示例中,非默认覆盖VRF可以由控制节点232来编程。

[0290] 在这种情况下,控制节点232可获得定义VNR 52的定制资源,并例示单独的路由实例(具有相应的公共路由目标)。控制节点232可为公共路由目标创建引入和/或引出策略,其使得从各种虚拟网络(例如,VN 50A和50N)到公共路由目标的路由的引入和引出。控制节点232可以解析公共路由目标以获得转发信息,该转发信息然后可以经由虚拟路由代理514被下推到虚拟路由器506。使用该转发信息,虚拟路由器506可在VN 50A和50N之间转发数据包(在一些互连性方案中,诸如上面提到的网格互连性方案)。

[0291] 图7A至7B图示了上述的双路由/控制平面解决方案。在图7A中,cRPD 324向虚拟路由器代理514提供底层路由/转发信息,在某些方面类似于路由器控制平面如何对路由器转发/数据平面进行编程。

[0292] 如图7B所示,cRPD 324交换可用于为VRF创建通过底层网络702的隧道的路由信息。隧道710是示例并且连接服务器12A的虚拟路由器506和服务器12X的虚拟路由器506。隧道710可以表示分段路由(SR)或SRv6隧道、通用路由封装(Generic Route Encapsulation,GRE)隧道、IP-in-IP隧道、LSP或其他隧道。控制节点232利用隧道710来创建连接服务器12A的pod 22和服务器12X的pod 22的虚拟网络712,其中服务器12A和服务器12X附接到虚拟网络的VRF。

[0293] 如上所述,cRPD 324和虚拟路由器代理514可以使用gRPC接口交换路由信息,并且虚拟路由器代理5145可以使用gRPC接口用配置对虚拟路由器506进行编程。还注意到,控制节点232可用于覆盖和编排,而cRPD 324可用于管理底层路由协议。虚拟路由器代理514可以使用与cRPD 324的gRPC接口,同时使用XMPP来与控制节点和域名服务(DNS)通信。

[0294] gRPC模型对于cRPD 324工作良好,因为可能存在在每个计算节点上运行的工作器,并且虚拟路由器代理314充当gRPC服务器,其向客户端(cRPD 324)展露服务以用于对路由和配置信息(针对底层)进行编程。因此,当与XMPP相比时,gRPC作为解决方案是有吸引力的。特别地,它将数据作为二进制流传送,并且在要通过它发送的编码/解码数据中没有添加的开销。

[0295] 在一些示例中,控制节点232可以使用XMPP与虚拟路由器代理514接口连接。在虚拟路由器代理514充当gRPC服务器的情况下,cRPD 324充当gRPC客户端。这意味着客户端(cRPD)需要发起到服务器(vRouter代理)的连接。在SDN架构700中,虚拟路由器代理514选择订阅的控制节点232的集合(因为存在多个控制节点)。在该方面,控制节点232充当服务器,并且虚拟路由器代理514作为客户端连接并订阅更新。

[0296] 关于gRPC,控制节点232将需要挑选其需要连接的虚拟路由器代理514,然后作为客户端订阅。由于控制节点232不在每个计算节点上运行,因此这将需要实现算法来选择它可以预订的虚拟路由器代理514。此外,控制节点232需要彼此同步该信息。这也使重启发生时的情况复杂化,并且需要控制节点232之间的同步来挑选它们所服务的代理。诸如平滑重启动(Graceful Restart,GR)和快速收敛(Fast Convergence)的特征已经在XMPP之上实现。XMPP已经是轻质和有效的。因此,对于控制节点232到虚拟路由器代理514的通信,XMPP可能优于gRPC。

[0297] 对控制节点232的附加增强及其使用如下。HA和具有三个控制节点的水平可缩放

性。与任何路由平台类似,仅具有两个控制节点232就足以满足HA需求。在许多情况下,这是有利的。(然而,可以使用一个或多个控制节点232。)例如,它提供了更确定的基础设施并且与标准路由最佳实践一致。每个虚拟路由器代理514附接到唯一的一对控制节点232以避免随机性。利用两个控制节点232,调试可以更简单。此外,用于构造多播/广播树的边缘复制可以仅用两个控制批注232来简化。当前,由于vRouter代理314只连接到三个控制节点中的两个,所以所有控制节点在一段时间内可能没有完整的树的图片,并且依赖BGP来同步它们之间的状态。由于虚拟路由器代理314可以随机选择两个,所以这在三个控制节点232的情况下更加严重。如果只有两个控制节点232,则每个虚拟路由器代理314将连接到相同的控制节点。这又意味着控制节点232不需要依赖BGP来同步状态,并且将具有多播树的相同图片。

[0298] SDN架构200可以提供入口复制作为边缘复制的替代,并向用户提供选项。入口复制可以被视为一般覆盖多播树的特殊退化情况。然而,实际上,入口复制树的信令比一般的覆盖多播树的信令简单得多。通过入口复制,每个虚拟路由器21以树结束,该树以其自身作为根并且以每个其他vrouter作为叶。虚拟路由器21下降理论上不应导致重建树。注意,入口复制的性能随着更大的集群而恶化。然而,它对于较小的集群工作良好。此外,多播对于许多客户来说不是普遍的和流行的要求。它主要限于传输广播BUM流量,这仅在最初发生。

[0299] 配置处理模块增强

[0300] 在常规SDN架构中,网络控制器处理所有使用情况的编排。配置节点基于数据模型将意图转换成配置对象,并将它们写入数据库(例如,Cassandra)。在一些情况下,同时,例如经由RabbitMQ向等待该配置的所有客户端发送通知。

[0301] 控制节点不仅充当BGP讲者,而且具有配置处理模块,该配置处理模块以下面的方式从数据库读取配置对象。首先,当控制节点出现(或重新启动)时,它连接到数据库并直接从数据库读取所有配置。第二,控制节点也可以是消息客户端。当存在对配置对象的更新时,控制节点接收列出已被更新的对象的消息通知。这又使配置处理模块从数据库读取对象。

[0302] 配置处理模块读取控制平面(BGP相关配置)和vRouter转发平面的配置对象。该配置可以存储为具有对象作为节点和关系作为链接的图。然后可以将该图下载到客户端(BGP/cRPD和/或vRouter代理)。

[0303] 在一些实例中,常规配置API服务器和消息传递服务在一些实例中由Kubernetes中的etcd的Kube api服务器(API服务器300和定制API服务器301)和先前的Cassandra数据库代替。通过这种改变,对配置对象感兴趣的客户端可以直接观察etcd数据库以获得更新,而不是依赖于RabbitMQ通知。

[0304] 用于CRPD的控制器编排

[0305] BGP配置可以被提供给cRPD 324。在一些示例中,cRPD控制器720可以是Kubernetes控制器,该Kubernetes控制器适于开发其自己的控制器,该控制器适于Kubernetes空间并实现编排和供应cRPD 324所需的CRD。

[0306] 分布式配置处理

[0307] 如本节前面所述,配置处理模块可以是控制节点232的一部分。它直接从数据库读取配置、将数据转换成JSON格式,并将其作为对象作为节点,对象之间的关系作为链接的图

形存储在其本地IFMAP数据库中。然后,该图经由XMPP被下载到计算节点上的感兴趣的虚拟路由器代理514。虚拟路由器代理514本地地构建基于IFMAP的依赖图,以及存储这些对象。

[0308] 作为中间模块的IFMAP和存储依赖图的需要可以通过使虚拟路由器代理514直接监视API服务器300中的etcd服务器来避免。在计算节点上运行的cRPD 324可以使用相同的模型。这将避免对IFMAP-XMPP config信道的需要。Kubernetes配置客户端(用于控制节点232)可用作该配置的一部分。该客户端也可以由虚拟路由器代理使用。

[0309] 然而,这可增加从etcd服务器读取配置的客户端的数量,尤其是在具有数百个计算节点的集群中。添加更多的观察器最终使得写入速率下降并且事件速率达不到理想状态。etcd的gRPC代理从一个服务器观察器向多个客户端观察器重播。gRPC代理将同一键或范围上的多个客户端观察器(c-观察器)合并到连接到etcd服务器的单个观察器(s-观察器)中。代理将所有事件从s-观察器广播到其c-观察器。假设N个客户端观看相同的键,一个gRPC代理可以将etcd服务器上的观看负载从N减少到1。用户可以部署多个gRPC代理以进一步分配服务器负载。这些客户端共享一个服务器观察器;代理有效地从核心群集卸载资源压力。通过增加代理,etcd每秒可以服务一百万个事件。

[0310] SDN架构中的DNS/命名

[0311] 在先前的架构中,DNS服务由共同工作以向网络中的VM提供DNS服务的contrail-dns和contrail-named的进程提供。其命名作为提供BIND协议的实现的DNS服务器。contrail-dns从vrouter-代理接收更新并将这些记录推送至命名。

[0312] 系统中支持四种DNS模式,IPAM配置可以选择所需的DNS模式。

[0313] 1.没有-无对VM的DNS支持。

[0314] 2.默认DNS服务器-VM的DNS解析基于服务器基础结构中的命名服务器配置来完成。当VM得到DHCP响应时,子网默认网关被配置为VM的DNS服务器。VM发送到该默认网关的DNS请求经由配置在相应计算节点上的(结构)命名服务器来解析,并且响应被发送回VM。

[0315] 3.租户DNS服务器-租户可以使用该模式使用其自己的DNS服务器。可以在IPAM中配置服务器列表,然后在对于作为DNS服务器的VM的DHCP响应中发送。VM发送的DNS请求基于可用路由信息被路由为任何其他数据数据包。

[0316] 4.虚拟DNS服务器-在该模式中,系统支持虚拟DNS服务器,提供解析来自VM的DNS请求的DNS服务器。我们可以在系统中的每个域下定义多个虚拟域名服务器。每个虚拟域名服务器是用于所配置的DNS域的授权服务器。

[0317] 本文所述的SDN架构在其提供的DNS服务方面是高效的。云原生世界中的客户将受益于各种DNS服务。然而,随着向下一代基于Kubernetes的架构的移动,SDN架构可以替代地使用coreDNS用于任何DNS服务。

[0318] 数据平面

[0319] 数据平面由两个部分组成:SDN架构解决方案中的虚拟路由器代理514(aka代理)和虚拟路由器转发平面506(也称为DPDK vRouter/Kernel vRouter)负责管理数据平面部件。代理514与两个控制节点232建立XMPP邻居关系,然后与它们交换路由信息。vRouter代理514还动态地生成流条目并将它们注入到虚拟路由器506中。这向虚拟路由器506给出关于如何转发数据包的指令。

[0320] 代理514的职责包括:与控制节点232接口以获得配置。将接收到的配置转换成数

据路径能够理解的形式(例如,将数据模型从IFMap转换成数据路径所使用的数据模型)。与控制节点232接口连接以管理路由。并从数据路径收集和引出统计数据至监测解决方案。

[0321] 虚拟路由器506实现可允许虚拟网络接口与VRF相关联的数据平面功能。每个VRF具有其自己的转发和流表,而MPLS和VXLAN表在虚拟路由器506内是全局的。转发表可以包含目的地的IP和MAC地址的路由,并且IP-to-MAC关联被用于提供代理ARP能力。当VM/容器接口出现并且仅对虚拟路由器506本地重要时,虚拟路由器选择MPLS表中的标签的值。VXLAN网络标识符在域内的不同虚拟路由器506中的相同虚拟网络的所有VRF上是全局的。

[0322] 在一些示例中,每个虚拟网络具有分配给它的默认网关地址,并且每个VM或容器接口在初始化时接收的DHCP响应中接收该地址。当工作负载向其子网外部的地址发送数据包时,它将针对与网关的IP地址相对应的MAC进行ARP,并且虚拟路由器506以其自己的MAC地址进行响应。因此,虚拟路由器506可以支持用于所有虚拟网络的完全分布式默认网关功能。

[0323] 以下是由虚拟路由器506实现的数据包流转发的示例。

[0324] 数据包在同一子网中的VM/容器接口之间流动。

[0325] 工作器节点可以是VM或容器接口。在一些示例中,数据包处理进行如下:

[0326] • VM1/容器接口需要向VM2发送数据包,因此虚拟路由器506首先针对IP地址查找其自己的DNS高速缓存,但是由于这是第一数据包,因此不存在条目。

[0327] • VM1向在其接口出现时在DHCP响应中提供的DNS服务器地址发送DNS请求。

[0328] • 虚拟路由器506捕获DNS请求并将其转发到在SDN架构控制器中运行的DNS服务器。

[0329] • 控制器中的DNS服务器以VM2的IP地址作出响应

[0330] • 虚拟路由器506将DNS响应发送到VM1

[0331] • VM1需要形成以太网帧,因此需要用于VM2的MAC地址。它检查它自己的ARP高速缓存,但是没有条目,因为这是第一个数据包。

[0332] • VM1发出ARP请求。

[0333] • 虚拟路由器506捕捉ARP请求并在其自己的转发表中查找IP-VM2的MAC地址,并找到控制器将其发送给VM2的L2/L3路由中的关联。

[0334] • 虚拟路由器506向VM1发送具有VM2的MAC地址的ARP应答。

[0335] • 在VM1的网络栈中发生TCP超时

[0336] • VM1的网络栈重试发送数据包,这次在ARP缓存中找到VM2的MAC地址,并且可以形成以太网帧并将其发送出去。

[0337] • 虚拟路由器506查找VM2的MAC地址并找到封装路由。虚拟路由器506建立外报头并将结果数据包发送到服务器S2。

[0338] • 服务器S2上的虚拟路由器506解封装该数据包并查找MPLS标签以标识发送原始以太网帧至其的虚拟接口。以太网帧被发送到该接口并由VM2接收。

[0339] 不同子网中VM之间的数据包流

[0340] 在一些示例中,向不同子网中的目的地发送数据包时的顺序是类似的,除了虚拟路由器506作为默认网关响应。VM1将在具有默认网关的MAC地址的以太网帧中发送该数据包,该默认网关的IP地址是在VM1引导时虚拟路由器506提供的DHCP响应中提供的。当VM1对

网关IP地址进行ARP请求时,虚拟路由器506用它自己的MAC地址进行响应。当VM1使用该网关MAC地址发送以太网帧时,虚拟路由器506使用帧内的数据包的目的地IP地址来查找VRF中的转发表以找到路由,其将经由封装隧道到达目的地正在其上运行的主机。

[0341] 图10A至10M是图示了提供网络的拓扑的图形表示的图形用户界面的示例的示图,其中虚拟网络路由器可被图形地定义以允许不同网络元件之间的连通性。在图10A的示例中,GUI 2000A可以表示GUI 60/244的一个示例,其呈现网络拓扑的图形表示2002A。图形表示2002A可将网络拓扑呈现为包括表示网络元件(例如,VN、VNR、服务器集群等)的图形节点和表示图形节点之间的连通性的图形边的图形数据结构。

[0342] 每个图形节点可以使用文本和图形图标来指定节点的类型。例如,图形节点2004A包括表示虚拟网络(具有显示标志的云)的图形图标和将图形节点2004A标识为"VN1"的文本。图形表示2002A可包括类似的图形图标2004B至2004N,其中图形图标表示虚拟网络,以及文本将图形节点2004B至2004N识别为相应的"VN2"、"VN3"、"VN4"、…"VN15"。作为另一示例,图形节点2006A包括表示集群的图形图标连同将图形节点2006A标识为"集群2"(其指的是服务器集群)的文本。

[0343] 图形节点2008A至2008C的每一个包括表示被配置为提供网格连通性的VNR的图形图标,以及将图形节点2008A至2008C的每一个标识为VNR的文本,VNR连同表示连通性的类型(例如,"网格")的文本。图形节点2010A包括表示VNR的图形图标,VNR被配置为提供中心连通性,以及将图形节点2010A标识为VNR的文本,还具有表示连通性的类型的文本(例如,支持中心辐条式连通性的"中心")。图形节点2012A包括表示VNR的图形图标,VNR被配置为提供辐条式连通性,以及将图形节点2012A标识为VNR的文本,连同表示连通性类型的文本(例如,支持中心辐条式连通性的"辐条")。假设中心辐条式连通性提供单向连通性,其中辐条可以仅向中心转发数据(而不直接向其他辐条转发数据),则图形边2014A包括指示中心辐条式连通性的单向性质的箭头。

[0344] GUI 2000A包括图标2020A(在该示例中具有表示"拓扑"的web的图形图标),该图标在被选择时使得GUI 2000A呈现拓扑2002A的图形表示。GUI 2000A还包括对应于"收藏夹"(图标2020B)、"性能"(图标2020C)、"状态"(图标2020D)、"配置"(图标2020E)、"分析"(图标2020F)和"设置"(图标2020G)的不同界面的图标2020B至2020G。图标2020A至2020G("图标2020")中的每一个可以使GUI 2000A适于示出拓扑的不同方面(或控制如何查看拓扑的不同方面)。

[0345] GUI 2000A还包括视图控件2030A至2030E。视图控件2030A可以展露标识用于描绘不同类型的图形节点、图形边等的各种图标的图例(如以下关于图10B的示例更详细描述的)。视图控件2030B在被选择时可以使GUI 2000A重置拓扑的图形表示2002A的视图(例如,当GUI 2000A响应于选择拓扑图标2020A而被首次呈现时,重置为原始视图)。视图控件2030C至2030E在被选择时可以使GUI 2000A放大和/或缩小拓扑的图形表示2002A的视图。

[0346] 此外,GUI 2000A包括各种模式控件2040A至2040D。模式控件2040A在被选择时可使GUI 2000A能够添加图形元件(例如,表示VNR、VN等的图形节点和图形边),其使得对由图形表示2002A表示的底层拓扑的配置的添加和/或改变。模式控件2040B在被选择时可以使得GUI 2000A能够编辑现有的图形元件,这可以使得对由图形表示2002A表示的底层拓扑的配置的改变。模式控件2040C在被选择时可使GUI 2000A启用搜索功能。模式控件2040D在被

选择时可以使GUI 2000A启用过滤功能,该过滤功能使用户能够输入过滤标准,通过该过滤标准来限制由图形表示2002A表示的拓扑的视图。

[0347] GUI 2000A还包括拓扑/列表切换控件2050A和2050B。拓扑切换控件2050A在被选择时可以使GUI 2000A转换到网络拓扑的图形表示2002A,而列表切换控件2050B在被选择时可以使GUI 2000A从网络拓扑的图形表示2002A转换到网络拓扑的列表视图。

[0348] 在图10A的示例中,GUI 2000A被配置为呈现“拓扑”视图,并且已经启用了展露过滤控件2060A的过滤功能。过滤控件2060A包括文本框,在该文本框中输入过滤标准(例如,“命名空间=ns1”)或移除过滤标准(例如,通过选择输入的过滤标准旁边的“X”)。过滤控件2060A还包括确认输入的过滤标准的确认复选标记控件、移除过滤标准的删除(“X”)控件、以及保存确认的过滤标准的保存过滤控件。给定“命名空间=ns1”的过滤标准,GUI 2000A已经过滤了网络的拓扑以经由图形表示2002A仅示出与命名空间命名“ns1”相关联的图形节点。过滤可以使用户能够仅关注与实现给定目标和/或功能相关的网络拓扑的特定部分。

[0349] 尽管本文所述为使得GUI 2000A响应于选择各种图标、控件等来执行各种功能,但是应当理解,GUI 2000A可以与网络控制器24对接以响应于选择各种图标、控件等来提供各种输入,于是网络控制器24可以更新GUI 2000A以展露和/或执行归于GUI 2000A(以及本文所述的其他示例GUI)的各种功能。

[0350] 接下来参考图10B的示例,GUI 2000B示出了选择视图控件2030A以示出图例框2070的结果。图例框2070提供了将图标与不同类型的网络元件相关联的图例,诸如“虚拟网络路由器-网格”、“虚拟网络路由器-中心”、“虚拟网络路由器-辐条”、“虚拟网络”和“集群”。图例框2070还识别不同的拓扑状态,诸如“选择的”、“错误”、“单向的”、“连接的”、“连接选择的”、“连接错误”和“连接未决”,将这些不同的拓扑状态链接到每个拓扑状态的图形表示。

[0351] 接下来参考图10C的示例,GUI 2000C示出了选择过滤控件2060A中的“+”以展露过滤标准提示框2080的结果。过滤标准提示框2080可经由下拉框、文本框、单选按钮、复选框等提示用户指定可应用于当前图形表示2002A的过滤标准,以进一步从网络拓扑的图形表示2002A中滤出图形元件(例如,图形节点2004,诸如图形边2014A等的图形边)。

[0352] 在图10D的示例中,GUI 2000D示出了接收对编辑模式控件2040B(在图10A至10C的示例中示出)的选择的结果,该选择使得GUI 2000D展露包括各种图标的编辑模式控件面板2090,这些图标指示(从左到右,以复选标记图标开始并以过滤图标结束)“接受”、“编辑”、“查看配置”、“取消”、“删除”、“搜索”和“过滤”。接收对编辑模式控件2040B的选择还使得GUI 2000D展露具有图标2094A至2094E(其中图标可以表示具有附加图形图像和/或文本的按钮形式)的添加模式控件面板2092。

[0353] 图标2094A在被选择时(通过用户与GUI 2000D通过接口连接)可使GUI 2000D启动一系列提示,通过该提示添加具有网格连通性的VNR。图标2094B在被选择时可使GUI 2000D开始一系列提示,通过这些提示添加具有辐条式连通性的VNR。图标2094C在被选择时可使GUI 2000D发起一系列提示,通过这些提示添加具有中心连通性的VNR。图标2094D在被选择时可以使GUI 2000D发起一系列提示,通过该提示添加VN,而图标2094E在被选择时可以使GUI 2000D发起一系列提示,通过提示添加图形元件(例如,VN和/或VNR)之间的连接(或换句话说,图形边)。

[0354] 接下来参考图10E的示例,GUI 2000E示出了选择图标2094B的结果,通过该图标2094B添加具有辐条式连通性的VNR,其展露了提示2100,通过该提示2100定义配置数据以添加具有辐条式连通性的新VNR。提示2100被示出为具有多个不同图形输入元件(诸如按钮、下拉框、文本框、预览、自适应文本框(例如,具有基于定义网络拓扑的底层配置数据的文本完成,以及便于用户交互的其他自适应、预填充和/或其他动态元件)等的弹出对话框。虽然被描述为弹出框,但是GUI 2000E可以以任何方式呈现提示2100,诸如通过弹出窗口、单独的GUI、附加标签、扩展框等。

[0355] 在任何情况下,图10E的示例中所示的提示100包括模式切换2102A和2102B、命名空间下拉框2104、VNR命名文本框2106、描述文本框2108、VNR类型切换2110A至2110C、可扩展提示部分2112A至2112C、预览部分2114。当切换2102A被选择时(再次由用户选择),可使GUI 2000E呈现如图10E的示例中所示的提示2100,切换2102B当被选择时,可使GUI 2000E呈现YAML文本编辑器,用户可通过该YAML文本编辑器使用YAML(或换言之,符合YAML的代码)定义新VNR的配置数据。

[0356] 命名空间下拉框2104可以允许用户选择先前定义的命名空间,在该命名空间中添加具有辐条式连通性的新VNR。VNR命名文本框2106可以使用户能够针对具有辐条式连通性的新VNR输入VNR命名。描述文本框2108可使用户能够输入具有辐条式连通性的新VNR的描述。VNR类型切换2110A至2100C可以使用户能够改变新VNR的连通性(这也可以使得提示2100的各种元件的更新,诸如可扩展的提示部分2112A至2112C),其中VNR类型切换2110A可以将连通性切换到网格连通性,VNR类型切换2110B可以将连通性切换到辐条式连通性(这是基于选择通过图标2094A将2094C添加哪种类型的VNR的默认情况),VNR类型切换2110C可以将连通性切换到中心连通性。

[0357] 可扩展提示部分2112A至2112C可基于选择了哪个VNR类型切换而被动态填充。在图10E的示例中,VNR类型切换2110B已被选择,这导致可扩展的提示部分2112A至2112C。然而,根据哪个VNR类型切换2110A至2110C已被选择,提示2100可以包括不同的、更多的、或更少的可扩展的特定于经由VNR类型切换2110A至2110C选择的特定VNR类型的提示部分。

[0358] 可扩展提示部分2112A包括各种图形元件,通过这些图形元件,根据VNR键和VNR标签指定VNR标签。可扩展提示部分2112B包括各种图形元件,通过这些图形元件指定用于新VNR的VN。可扩展提示部分2112C包括各种图形元件,通过这些图形元件指定新的辐条VNR应该被配置为与之连接以启用中心辐条式连通性的中心VNR(在潜在的VNR标签和/或命名空间的上下文中)。

[0359] 预览部分2114可以表示通过其预览除VNR连通性类型之外的新VNR的配置的部分。假设用户没有完成提示2100的任何部分,预览部分2114提供具有相应VNR连通性类型(即,图10E的示例中的辐条式连通性)的新VNR(其可以被称为占位符辐条VNR 2116)的占位符图形表示2116。在该示例中,预览部分2114还包括启用预览切换以启用或禁用预览部分2114。

[0360] 图10F是图示图10E的示例中所示的更新提示2100以定义具有命名“VNR11”的新辐条VNR的结果的示图。命名空间下拉框2104已经填充了定义新辐条VNR所属的命名空间为“命名空间1(NAMESPACE 1)”的输入。VNR命名文本框2106已填充指定上述命名“VNR11”的输入。描述文本框2108已填充指定“默认网络的VNR”描述的输入。

[0361] 可扩展提示部分2112A已经填充了将键定义为“站”该站中的标签为“SJC”的输入。

可扩展提示部分2112B已经动态填充(例如,通过GUI 2000F)了受经由过滤控制面板2150输入的过滤标准限制的各种VN(通过命名和命名空间)(其中,这样的过滤标准-“站＝SVL”-指示将返回SVL站处的所有VN)。

[0362] GUI 2000F可以动态更新提示2100的预览部分2114,以反映上述每个输入和/或动态填充事件的条目(通过GUI 2000F)。在图10F的示例中,GUI 2000F可以更新提示2100的预览部分2114,以将占位符辐条VNR 2116替换为具有辐条图标以及反映新VNR的命名(“VNR11”)和新辐条VNR的连通性类型(“辐条”)的文本的辐条VNR 2118。GUI 2000F还可更新提示2100的预览部分2114,以表示与各种匹配VN的潜在连通性(如由虚线边定义的),将匹配VN表示出为具有VN图标、标有“25VN匹配”的文本和进一步识别匹配VN的数量(“25”)的标记的图形节点2120。

[0363] 接下来参考图10G的示例,GUI 2000G示出了根据将先前输入的过滤标准(“Site＝SVL”)与新输入的过滤标准(“Redis In 19,50”)结合,将进一步的过滤标准添加到过滤控制面板2150连同动态过滤辅助提示2152的结果。动态过滤辅助提示2152可以建议布尔运算符,诸如图10G中所示的“AND”和“OR”的示例。虽然建议了示例性布尔运算符,但是动态过滤辅助提示2152可以指定任何类型的过滤标准运算符,诸如数学运算符、关键字、值等。

[0364] 在图10H的示例中,GUI 2000H示出扩展可扩展提示部分2112C的结果,通过其,用户可以选择新的辐条VNR要连接到的中心VNR以配置中心辐条式连通性。可扩展提示部分2112C通知用户新的辐条VNR将连接到与下面选择的VNR标记和命名空间组合匹配的所有中心VNR,然后列出各种中心,允许用户选择(经由复选框)新辐条VNR应连接到与VNR标签和命名空间组合匹配的现有中心VNR中的哪一个。用户还可以与可扩展提示部分2112C接口以添加不同的中心VNR(经由“+”图标),编辑列出的中心VNR(经由铅笔图标),以及删除各种中心VNR(经由垃圾桶图标)。

[0365] 在图10I的示例中示出的GUI 2000I的结果,示出在可扩展提示部分2112C中取消选择中心VNR,这使得GUI 2000I更新提示2100的预览部分2114。预览部分2114更新潜在拓扑的预览,以将潜在连通性添加到三(3)个匹配中心VNR,这由从图形节点2120到代表三个潜在匹配中心VNR的新图形节点2122的虚线边表示。图形节点2122包括与中心VNR相关联的图标、标注“3个中心匹配”的文本,以及强调匹配中心数量(“3”)的标记。

[0366] 在这方面,GUI 2000F至2000I由网络控制器24配置以动态地生成图形元件,该图形元件包括与虚拟网络路由器的连通性类型相关联的图形图标。在动态地生成图形元件(例如,该示例中的图形节点)时,网络控制器24可接收来自用户的指示VNR的连通性类型(例如,辐条式连通性)的输入,并选择与虚拟网络路由器的连通性类型相关联的图形图标。网络控制器24然后可以配置GUI 2000F至2000I中的一个或多个来呈现包括与虚拟网络路由器的连通性类型相关联的图形图标的图形元件。

[0367] 图10J是示出交互的动态流的图,通过该交互,用户可以与GUI 2000J接口以图形地将代表VN的图形节点2004G(其可以被称为“VN8 2004G”)连接到代表网格VNR的图形节点2008C(其可以被称为“网格VNR52008C”)。交互的动态流由以数字一(1)开始并以数字五(5)结束的编号虚线圆圈表示。

[0368] 在动态流步骤1,用户最初可以选择(例如,经由鼠标左击)来展露GUI 2000J可以用VN8 2004G(和/或可能图形节点2004G的特定类型VN)专用的选项来填充的选项提示

2300。在图10J的示例中,用户从选项提示2300中选择“编辑拓扑”而不是“查看详细信息”。响应于接收到选择“编辑拓扑”的输入,GUI 2000J转换到编辑拓扑提示2302(在动态流步骤二),其中用户选择“添加连接”而不是“编辑详细信息”或“删除节点”)。

[0369] 响应于接收到指定经由编辑拓扑提示2302选择了“添加连接”的输入,GUI 2000J可在动态流步骤三呈现提示2304,其提示用户图形地选择你想要连接到VN8 2004G的节点。用户接下来可以选择网格VNR5 2008C(如动态流步骤四所示),于是GUI 2000J可以响应于指定网格VNR52008C被选择的输入,更新图形表示2002A(在动态流步骤五),以反映VN8 2004G和网格VNR5 2008C之间的未决连接。

[0370] 在图10K的示例中,GUI 2000K示出了经由以上参照图10J的示例描述的GUI 2000J完成动态流的结果。响应于接收到指示动态流已完成的输入,GUI 2000K可呈现提示2400,其引导用户查看对VN8 2004G的VN标签添加。提示2400可显示现有VN8标签和向VN8 2004G添加标签,其中每个标签包括复选框以包括(选中)或移除对VN8 2004G的标签添加(包括现有标签以及标签添加)。

[0371] 在图10L和10M的示例中,GUI 2000L/2000M示出了选择列表切换控件2050B的结果。在图10L的示例中示出的GUI 2000L呈现了VNR的当前列表,其中用户可以选择(经由复选框)列表中的一个或多个VNR。GUI 2000M(在图10M的示例中示出)示出了从GUI 2000L中示出的列表中选择VNR 8的结果,于是GUI 2000M呈现指定VNR 8的详细信息的覆盖框2500。

[0372] 图11是图示了根据本公开所述的技术的各方面的其中虚拟网络路由器可被配置为实现虚拟网络之间的网格互连性的第一实例的示图。在图11的示例中,虚拟网络(“VN”)1500A(也被示出为“VN1”)和VN 1500B(也被示出为“VN2”)在Kubernetes(或一些其他编排平台)内被定义和实现为定制资源,其然后被实现为SDN架构(例如,SDN架构700)内的VN。VN 1500A、1500B可以是定制资源的实例,并且具有相应的客户API服务器301和定制资源控制器302。

[0373] 网络控制器24可以允许管理员将VN 1500A定义为向pod 1502A(也示出为“Pod-1”)提供网络连通性的定制资源,并将VN 1500B定义为向pod 1502B(也示出为“Pod-2”)提供网络连通性的定制资源。

[0374] 如图11的示例中进一步所示,VN 1500A和VN 1500N(“VN 1500”)都被定义在相同的命名空间1504(也被示出为“命名空间-1”)内,这指示两个VN 1500都存在于公共网络命名空间内(并且因此彼此不隔离)。至少在Kubernetes的上下文中,命名空间提供用于在单个集群内隔离资源组,例如pod 1502A和pod 1502B(“pod 1502”)和VN 1500的机制。如左上方所示,VN 1500未互连,并且彼此之间不提供网络连通性,如pod-2(来自pod 1502A)的ping状态为“失败”和pod-1(来自pod 1502B)的ping状态为“失败”所示。

[0375] 管理员可以通过在相同的命名空间1504(再次示出为“命名空间-1”)内例示VNR 1506来提供pod 1502之间的网络连通性,将VN 1500互连到另一个。如上所述,VNR 1506可以表示管理员可以用具有网络连通性类型“网格”的示例命名“VNR-web”来例示的定制资源。为了指定要互连哪些VN,管理员可将标签分配给每个VN 1500作为“vn:web”的标签,然后指定具有“vn:web”的选择标签的VN 1500之间的网格连通性。

[0376] 网络控制器24(例如,定制资源控制器302)可处理该定制资源以生成上述引出和引入策略,该引出和引入策略指示经由VNR 1506将VN 1500A的所有路由将被引出到VN

1500B,并且来自VN 1500B的所有路由被引出到VN 1500A,经由VNR 1506,来自VN 1500B的所有路由将被引入到VN 1500A,并且来自VN 1500A的所有路由将被引入VN 1500B。这种引入/引出策略通常被称为对称的,因为每个VN的所有路由在每个其他VN之间交换,从而创建网格。

[0377] VNR 1506可通过在公共路由实例(RI,示出为“vnr-RT”)的上下文内创建公共路由目标(示出为“vnr-RT”,其中RT表示路由目标)“安装”,在公共路由实例中,引入/引出策略被定义为从每个VN 1500引入和引出到vnrRT的所有路由。一旦这些策略被定义并安装在与VN 1500相关联的SDN架构700内(例如,在路由平面中),则路由将被引入并引出到vnr-RT中的每一个的VRF中,并被解析以生成转发信息。vnr-RI和vnr-RT的实例可以是用于SDN架构配置的定制资源的实例。

[0378] 控制节点232可以生成配置数据形式的转发信息并与虚拟路由器代理514接口以将配置数据提供给虚拟路由器代理514,虚拟路由器代理514可以将转发信息安装到虚拟路由器506以实现VN 1500的相应VRF(例如,图5的示例中所示的VRF 222A)。一旦安装,转发信息使得虚拟路由器506和具有用于VN 1500的Pod的其他计算节点中的其他虚拟路由器正确地用于VN间流量。这些虚拟路由器可以使用泄漏的路由信息在pod1502之间正确地转发ping,如图10的示例所示,通过指示ping到pod-2(从pod 1502A)“通过”以及ping到pod-1(从pod 1502B)“通过”。

[0379] 图12是图示了根据本公开所述的技术的各方面的其中虚拟网络路由器可被配置为实现虚拟网络之间的网格互连性的第二实例的示图。图12中所示的示例由图11中所示的示例构建,其中VNR 1506提供VN 1500之间的网格互连性,这允许相同命名空间1504中的pod 1502彼此对称地通信。

[0380] 在图12的示例中,VN 1500C和VN 1500D(也分别示出为“VN3”和“VN4”)被定义在相同的命名空间1504内,并且为相应的pod 1502C和pod 1502D(也示出为“Pod-3”和“Pod-4”)提供联网连通性。为了将VN 1500C和VN 1500D添加到由VNR 1506提供的网格,管理员可将“vn:web”标记添加到VN 1500C和VN 1500D。

[0381] 在添加对应于网格VNR 1506的标签之后,网络控制器24可通过生成引入和引出策略以经由VNR公共路由目标(“vnr-RT”)引出和引入VN 1500C与VN 1500A、VN 1500B和VN 1500D之间的所有路由,并且再次经由VNR公共路由目标(“vnr-RT”)引出和引入VN 1500D与VN 1500A至1500C之间的所有路由,将VN 1500C和VN 1500D添加到网格VNR 1506。即,控制节点232可能仅需要生成将路由从vnr-RT引入VN 1500C和VN 1500D以及将路由从VN 1500C和VN 1500D引出到vnr-RT的策略。通过使用VNR 1506的公共路由目标,在一些情况下,路由可以在所有VN 1500A至1500D之间自动地被引出,而不必更新任何现有策略(诸如现有VN 1500A和VN 1500B的引入/引出策略)。

[0382] 在任何情况下,控制节点232可以将路由解析为转发信息,并与虚拟路由器代理514通过接口连接,以便以与上面参照图10所述的方式类似的方式安装转发信息。在安装转发信息之后,每个pod 1502A至1502D可以具有与每个其他pod 1502A至1502D的网格网络连通性,如通过相对于每个其他pod 1502A至1502D的“通过”ping状态所示(例如,pod 1502A,表示为“Pod-1”,对于每个pod 1502B至1502D,具有“通过”ping状态,对于每个其他的pod 1502B至1502D之一,表示为“Pod-2、Pod-3、Pod-4”等)。图12以这种方式图示了向使用VNR互

连的VN的网格添加附加VN可以通过简单地向VN实例添加适当的标签来实现,这里,“vn:web”。

[0383] 图13A和13B是图示了根据本公开所述的技术的各个方面的第三实例的图,在该第三实例中,虚拟网络路由器可以被配置为实现虚拟网络之间的网格互连性。在该示例中,与图12的示例不同,VN 1500C和VN 1500D被耦接到图13A的示例中的VNR 1506B,以提供VN 1500C和VN 1500D(右侧)之间的网格网络连通性,其中,图12的示例中的VNR 1506被图13A的示例中的类似VNR 1506A代替,以提供VN 1500A和VN 1500B(左侧)之间的网格网络连通性。

[0384] VNR 1506A在命名空间1504的上下文中具有与VNR 1506B的路由实例(示出为“vnr-2-RI”)不同的唯一路由实例(“vnr-RI”)。VNR 1506A提供vnr-RT,而VNR 1506B提供vnr-2-RT(尽管给定不同/唯一路由实例,但RT可以是相同的)。此外,VNR 1506A具有标签选择“vn:web”,其选择VN 1500A和VN 1500B,而VNR 1506B具有标签选择“vn:db”,其为指派给VN 1500C及VN 1500D的标签。这样,pod 1502A和1502DB不具有与pod 1502C和pod 1502D的网络连通性(由“失败”ping状态表示),而pod 1502C和pod 1502D不具有与pod 1502A和pod 1502B的网络连通性(由“失败”ping状态表示)。

[0385] 为了使pod 1502A至1502D具有网格网络连通性(因为VNR 1506A和1506B都是“网格”类型),管理员可以添加标签选择语句来选择其他VNR 1506A和1506B的标签。接下来参考图13B的示例,VNR 1506A现在包括标签选择语句“vnr:db”,网络控制器24将其转换成用于vnr-db-RT的引入和引出策略,从而从/向VNR 1506B的vnr-db-RT引入和引出路由。类似地,VNR 1506B现在包括标签选择语句“vnr:web”,网络控制器24可将其转换成从/向VNR 1506A的vnr-web-RT的引入和引出路由。连接到VNR 1506A(“VNR-web”)和VNR 1506B(“VNR-db”)的VN的所有Pod可以通过使得VNR 1506A、VNR 1506B引入/引出彼此的路由,以此方式使得能够使用VNR和标签彼此通信。

[0386] 再次,网络控制器24(例如,定制资源控制器302)可以在路由平面内部署这些策略,该路由平面然后可以执行路由信息交换。控制节点232可以将交换的路由信息解析成转发信息,并与虚拟路由器代理512通过接口连接,以便以类似于以上参照图11所述的方式安装转发信息虚拟路由器506。因此,pod 1502A至1502D之间的网格连通性被建立,如通过pod1502A至1502D中的彼此的成功“通过”ping状态所证明的。

[0387] 图14A和14B是图示了根据本公开所述的技术的各方面的其中虚拟网络路由器可被配置为实现虚拟网络之间的网格互连性的第四实例的示图。在图14A的示例中,除了VN 1500A和VN 1500B,pod 1502A和pod1502B以及VNR 1506A在第一命名空间1504A(示出为“命名空间-1”)中,而VN 1500C和VN 1500D,pod 1502C和pod 1502D以及VNR 1506B在不同的第二命名空间1504B(示出为“命名空间-2”)中之外,一般网络结构与图13A的示例中所示的相似。

[0388] 假定命名空间1504A和1504B将VN 1500A/1500B、pod 1502A/1502B和VNR 1506A与VN 1500C/1500D、pod 1502C/1502D和VNR 1506B隔离,则必须有跨命名空间1504A和1504B引入路由信息的授权,因为每个命名空间1504A和1504B可由单独的组管理,而没有跨命名空间1504A和1504B的共享管理权限。这样,可以向两个命名空间1504A和1504B的管理员发送请求,以确认允许在VNR 1506A和1506B之间引入和引出路由信息。

[0389] 此外,由于命名空间1504A和1504B相对于其他命名空间所提供的隔离,仅允许VNR 1506A和1506B在命名空间1504A和1504B上引入和引出路由信息。对命名空间间路由信息的此限制强制实施以上授权且避免可能使得交换路由信息的错误配置,该交换允许在命名空间1504A与1504B之间传输具有敏感或其他安全信息的数据包(其可使得违背数据合规、使合同义务无效或以其他方式使得恶意信息违背由命名空间1504A与1504B提供的隔离)。

[0390] 因此,假定由命名空间1504A和1504B的管理员提供授权,则控制节点232可将VNR 1506A和VNR 1506B转换成上面关于图13B的示例描述的引入和引出策略,该引入和引出策略促进生成转发信息以提供pod1502A至pod 1502D之间的网格连通性。图14B示出了策略的这种转换和安装的结果,以及将交换的路由信息解析为可以安装在虚拟路由器506中以实现这种命名空间间网格网络连通性的转发信息。

[0391] 图15是图示了根据本公开所述的技术的各个方面的第五实例的图,其中虚拟网络路由器可以被配置为实现虚拟网络之间的中心辐条式互连性。在图15的示例中,VN 1500A至1500C在命名空间1504内,并为相应的pod 1502A至1502C提供网络连通性。VN 1500A和VN 1500B每个被分配VN标签“vn:web”并且VN 1500C被分配VN标签“vn:db”。

[0392] 另外,管理员可与网络控制器24交互,以例示VNR 1516A和VNR 1516B形式的定制资源。VNR 1516A是“中心”型VNR,它在中心辐条式网络配置中用作中心。中心辐条式网络配置是以下交换,即,中心VNR 1516A引入来自耦接到中心VNR 1516A的任何辐条VNR的所有路由信息,并且耦接到中心VNR 1516A的辐条VNR引入来自中心VNR 1516A的路由信息。VNR 1516B被配置为“辐条”式VNR,它在中心辐条式网络配置中充当辐条。辐条VNR 1516B可能向中心VNR 1516A引出所有的路由信息,但是从中心VNR 1516A引入所有的路由选择信息。然而,辐条VN不从其他辐条VN接收任何路由信息。

[0393] 管理员可以定义中心VNR 1516A以选择标签“vn:db”并将辐条VNR 1516B选择标签“vn:web”。如上所述,标签选择器可以指示VN 1500A至1500C中的哪一个耦接到中心VNR 1516A和辐条VNR 1516B中的哪一个。

[0394] 在图15的示例中,控制节点232可以生成用于中心VNR 1516A(连同辐条-RI或vnr-中心-RI)的中心公共路由目标(“中心-RT”或“vnr-中心-RT”)和用于辐条VNR 1516B(连同中心-RI或vnr-中心-RI)的辐条公共路由目标(“辐条-RT”或“vnr-辐条-RT”(“spoke-RT”或“vnr-spoke-RT”))。控制节点232接下来可生成用于中心VNR 1516A的引出策略,其引导用标签“vn:db”标记的任何VN 1500A至1500C),在该示例中是VN 1500C,以将路由信息引出到vnr-中心-RT。

[0395] 控制节点232还可以为辐条VNR 1516B生成指示辐条VNR 1516B应当将所有路由信息引出到vnr-中心-RT的输出策略。控制节点232可以生成用于中心VNR 1516A的引入策略,该引入策略指示中心VNR 1516A应当引入来自任何辐条VNR的所有路由信息,诸如辐条VNR 1516B的vnr-辐条-RT。控制节点232还可以生成指示VN 1500A和VN 1500B应当将所有路由信息引出到vnr-spoke-RT的引入和引出策略。

[0396] 一旦配置了策略,则控制节点232就可以根据策略交换路由信息,解析路由信息以生成转发信息。经由虚拟路由器代理512的控制节点232可以将转发信息安装在虚拟路由器506中,以实现中心辐条式互连性,其中pod 1502C中的每一个可以与pod 1502A和pod 1502B中的每一个通信(因为中心连接的pod可以到达任何辐条pod),并且pod 1502A和pod

1502B可以与中心pod 1502C通信,但是不能在彼此之间通信,因为辐条pod1502A和pod 1502B不能在每个中心辐条式网络连通性下彼此通信。

[0397] 图16是图示了第六实例的示图,在该第六实例中,虚拟网络路由器可以被配置为根据本公开所述的技术的各方面来实现虚拟网络之间的中心辐条式互连性。图16所示的示例类似于图15所示的示例,除了在该示例中,中心VNR 1516A、VN 1500C和pod 1502C在一不同的命名空间1504B(示出为“命名空间-2”)中,而辐条VNR 1516B、VN 1500A和VN 1500B以及pod 1502A和pod 1502B在第一命名空间1504A(示出为“命名空间-1”)中。

[0398] 在这种情况下,VN 1500A至1500C仅可以与相同命名空间中的VNR 1516A和1516B互连,并且出于上述原因,仅辐条VNR 1516B和中心VNR 1516A可以跨命名空间1504A和1504B通信。假设授权由两个命名空间1504A和1504B的管理员给出,在图16的底部示出的中心辐条式连通性可以以类似于(如果不是基本上类似于)以上关于图15的示例描述的方式来启用。

[0399] 图17是图示了第七实例的示图,在该第七实例中,虚拟网络路由器可以被配置为根据本公开所述的技术的各方面来实现虚拟网络之间的中心辐条式互连性。图17中所示的示例类似于图15中所示的示例,除了提供了允许VN 1500A至1500D之间的修改的中心辐条式互连性的附加网格VNR 1506。

[0400] 在图17的实例中,VN1500A至1500D最初不能彼此通信(每“失败”ping状态),直到中心VNR 1516A和辐条VNR 1516B被例示,被部署为策略,路由信息被交换,然后被解析,并且转发信息被安装在虚拟路由器506中,如上更详细地描述的。这使得VN 1500C和VN 1500D能够向公共中心-RT(“vnr-中心-RT”)引出路由信息,而VN 1500A和VN 1500B向公共辐条-RT(“vnr-辐条-RT”)引出路由信息。辐条VNR 1516B可以向vnr-中心-RT输出所有路由信息,这使中心pod 1502C和1502D能够与辐条pod 1502A和1502B中的每个其他进行通信。辐条pod 1502A和1502B可以与每个其他的中心pod 1502C和1502D通信,但不与辐条pod 1502A和pod 1502B中的另一个通信。

[0401] 为了使中心pod 1502C和1502D能够彼此直接通信,管理员可以例示网格VNR 1506,其使得能够将路由信息引出到与以上关于图10的示例所描述的类似的单独的网格-VNR。以这种方式,中心pod 1502C和1502D可以直接彼此通信,因为所有路由信息都经由网格-RT在VN 1500C和VN 1500D之间交换。

[0402] 图18A至18D是图示了根据本公开所述的技术的各方面在SDN架构内建立一个或多个虚拟网络路由器时所执行的操作的流程图。首先参考图18A的示例,网络控制器24的VNR控制器(例如,定制资源控制器302中的一个,诸如图6的示例中所示的定制资源控制器302A)可初始识别(或换句话说,观看Ectd数据库以识别)具有名称VN-1和vn=web的标签的VN 50A连同具有名称VN-2和vn=web的标签的VN 50N被创建为与网络控制器24相关联的Etcd数据库中的定制资源(例如,图6的示例中所示的配置存储1328),其中网络控制器24的控制器管理器(例如,控制器管理器1326)调用网络控制器24的调度器(例如,调度器1322)以创建VN 50A和50N(例如,1702、1704、1700)。

[0403] 经由网络控制器24的kube-api(其再次可指图6的示例中所示的API服务器300A和/或定制API服务器301A),管理员可例示VNR 52A以互连VN50A和50N,将类型指定为网格以及标签“vn:web”。Kube-API更新Etcd数据库以存储创建VNR 52A的新定制资源,并声明

VNR 52A的创建在Etcd数据库内未决(1706、1708、1710)。

[0404] 网络控制器24的调度器接下来可以从Etcd数据库接收已经定义了新的定制资源的通知,其然后与控制器管理器接口以通知控制器管理器VNR创建请求(1712、1714、1716)。控制器管理器可与VNR控制器接口,以请求VNR协调,其为VNR 52A创建RI,将新的RI提供给kube-API(1718),以便kube-API可更新Etcd,以反映新的RI到VNR 52A的分配(1720)。

[0405] 然后Etcd数据库可以与调度器接口以指示存在VNR-RI创建事件(1722),其进而与控制器管理器接口以向控制器管理器通知VNR-RI创建(1724)。控制器管理器然后与网络控制器24的RI控制器通过接口连接,以请求vnr-RI协调(1726)。

[0406] 接下来参考图18B的示例,网络控制器24的RI控制器可执行VNR-RI的协调,以便为VNR-1-RI创建RT,将VNR-1-RI报告回kube-api(1728)。Kube-api可更新Ectd,以注意VNR-1-RT已创建,这可向Kube-api报告VNR-1-RT未决,然后被成功创建(1730,1732)。Kube-api然后可以与VNR控制器接口以指示VNR-1-RI被创建,这又可以从RI获得VNR-1-RT(1734,1736)。

[0407] VNR控制器接下来可以与Kube-api通过接口连接以识别具有标签vn＝web的VN 50的列表(1740),其中Kube-api可以返回VN 50A和50N命名VN-1和VN-2(1742)。VNR控制器可以再次与Kube-api通过接口连接以获得VN-1和VN-2的RI(1744),其继续与Kube-api通过接口连接以在VN-1-RI、VN-2-RI状态下补丁VNR-1-RT(1746)。Kube-api然后更新VN-RI补丁事件的Ectd(1748)。Ectd然后通知调度器补丁事件(1750),其通知控制器管理器补丁事件(1752)。

[0408] 参考图18C的示例,响应于补丁事件,控制器管理器请求RI控制器执行VN-RI协调(1754),其在请求处于VN-1-RI、VN-2-RI状态的Kube-api补丁VNR-1-RT之前对VN-RI执行协调(1756)。响应于补丁请求,Kube-api更新Ectd中的VN-1-RI以包括用于引入和引出到VNR-1-RT(VNR公共路由目标)的策略,并更新VN-2-RI以包括用于引入和引出到VNR-1-RT(VNR公共路由目标)的策略(1758)。Ectd将补丁的状态报告回Kube-api(1760)。响应VN-1-RI和VN-2-RI的补丁的状态,Kube-api与RI控制器通过接口连接以在VNR-RI补丁VNR-1-RT(1762),其中RI控制器将补丁状态报告回Kube-api(1764)。

[0409] 接下来参考图18D的示例,Kube-api与Etcd通过接口连接以更新补丁事件上的Etcd,其与调度器接口以向调度器通知补丁事件(1766,1768)。调度器然后与控制器管理器通过接口连接以将VNR-RI补丁通知给控制器管理器,其又从VNR控制器请求VNR-RI协调(1770,1772)。RI控制器在VNR-RI处补丁VNR-1-RT,并通知Kube-api补丁(1774,1776)。Kube-api更新补丁上的Etcd(1778)并与RI控制器接口以设置VNR-1状态以指示“成功”),随后Kube-api在VNR-1状态成功时更新Ectd(1780)。

[0410] 以下提供了VNR API模式的详细设计(作为YAML文件):

API Type (Schema)

```
type VirtualNetworkRouterSpec struct {
        // Common spec fields
        CommonSpec `json:",inline" protobuf:"bytes,1,opt,name=commonSpec"`

        // Type of VirtualNetworkRouter. valid types - mesh, spoke, hub
        Type VirtualNetworkRouterType `json:"type,omitempty"
protobuf:"bytes,2,opt,name=type"`

        // Select VirtualNetworks to which this VNR's RT be shared
        VirtualNetworkSelector *metav1.LabelSelector
`json:"virtualNetworkSelector,omitempty"
protobuf:"bytes,3,opt,name=virtualNetworkSelector"`
```

[0411]

```
        // Import Router targets from other virtualnetworkrouters
        Import ImportVirtualNetworkRouter `json:"import,omitempty"
protobuf:"bytes,4,opt,name=import"`
}
type ImportVirtualNetworkRouter struct {
        VirtualNetworkRouters []VirtualNetworkRouterEntry
`json:"virtualNetworkRouters,omitempty"
protobuf:"bytes,1,opt,name=virtualNetworkRouters"`
}

type VirtualNetworkRouterEntry struct {
        VirtualNetworkRouterSelector *metav1.LabelSelector
`json:"virtualNetworkRouterSelector,omitempty"
```

```
protobuf:"bytes,1,opt,name=virtualNetworkRouterSelector"`
        NamespaceSelector *metav1.LabelSelector
`json:"namespaceSelector,omitempty"
protobuf:"bytes,2,opt,name=namespaceSelector"`
}
```

[1] VirtualNetworkRouter Yamls - Mesh VNR

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
        namespace: frontend
        name: vnr-1
        annotations:
                core.juniper.net/display-name: vnr-1
        labels:
[0412]                  vnr: web
                        ns: frontend
        spec:
                type: mesh
                virtualNetworkSelector:
                matchLabels:
                vn: web
        import:
                virtualNetworkRouters:
                        - virtualNetworkRouterSelector:
                                matchLabels:
                                        vnr: db
                          namespaceSelector:
                                matchLabels:
                                        ns: backend
```

[2] VirtualNetworkRouter Yamls - Spoke VNR

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
    namespace: frontend
    name: vnr-1
    annotations:
        core.juniper.net/display-name: vnr-1
    labels:
        vnrgroup: spokes
        ns: frontend
    spec:
        type: spoke
        virtualNetworkSelector:
        matchLabels:
            vngroup: spokes
```

[0413]

```
    import:
        virtualNetworkRouters:
            - virtualNetworkRouterSelector:
                matchLabels:
                    vnrgroup: hubs
              namespaceSelector:
                matchLabels:
                    ns: backend
```

[3] VirtualNetworkRouter Yaml - Hub VNR

```
apiVersion: core.contrail.juniper.net/v1alpha1
kind: VirtualNetworkRouter
metadata:
    namespace: backend
    name: vnr-2
    annotations:
        core.juniper.net/display-name: vnr-2
```

```
labels:
    vnrgroup: hubs
    ns: backend
spec:
    type: hub
    virtualNetworkSelector:
        matchLabels:
            vngroup: hubs
    import:
        virtualNetworkRouters:
            - virtualNetworkRouterSelector:
                matchLabels:
                    vnrgroup: spokes
              namespaceSelector:
                matchLabels:
                    ns: frontend
```

[0414]

[0415] 图19是图示了图1的示例中所示的计算机架构在执行本文所述的技术的各个方面时的操作的另一流程图。如上关于图1的示例所述,网络控制器24可以呈现GUI 60。在一些示例中,GUI 60可以表示基于web的GUI,网络控制器24可以经由客户端设备(诸如远程计算机或其他终端)远程托管该基于web的GUI以用于显示。GUI 60可以被配置为图形地表示由软件定义网络(SDN)架构系统8支持的网络的拓扑。出于说明的目的,假设网络包括VN 50A和VN 50N。然而,网络可以包括任何数量的VN 50或其他网络元件。

[0416] 网络控制器24可以维护定义网络的拓扑的配置数据(在图1的示例中未示出),该网络控制器24可以将其变换成网络的拓扑的图形表示。网络拓扑的图形表示可以便于查看网络的一部分或全部,其中GUI 60可以提供过滤和其他操作,以使得用户能够以各种粒度级别查看网络拓扑的图形表示,如下面更详细描述的。

[0417] GUI 60还可以被配置为动态地生成表示VNR 52A的图形元件,通过该图形元件互连VN 50A和VN 50N。GUI 60可以接收来自用户的标识VNR 52A和VN 50A和50N的输入,诸如指示代表VNR的图形元件(例如,图形图标)已被拖到可能接近、邻近和/或在VN 50A和50N中的一个或多个之上的网络拓扑上的输入。GUI 60可以将这些输入提供回网络控制器24,该网络控制器24可以响应于这些输入,更新GUI 60以包括关于VNR 52A和VN 50A和50N的信息的附加提示。

[0418] GUI 60(经由网络控制器24)可以以这种方式迭代,直到VNR 52A已经以实现VN 50A和50N之间的连通性的方式被成功定义。在这方面,网络控制器24可以呈现UI 60,该UI 60被配置为图形地表示网络的拓扑,并且动态地生成表示虚拟网络路由器的图形元件,通过虚拟网络路由器来互连包括在网络中的VN 50A/50N(1800)。

[0419] 网络控制器24可以执行配置节点30以在调用控制节点32来配置VN 50A和50N之前

验证VNR 52A。一旦被成功验证,控制节点32就根据一个或多个策略来配置VN 50A和50N,以允许经由VNR 52A在VN 50A和VN 50N之间引入和引出路由信息中的一个或多个。换言之,网络控制器24可以执行控制节点,该控制节点被配置为根据由VNR 52A表示的一个或多个策略来配置VN 50A/50N,其使能VN 50A/50N之间的路由信息的引入和引出中的一个或多个(1802)。

[0420] 以此方式,该技术的各个方面可实现以下示例。

[0421] 示例1.一种用于软件定义网络(SDN)架构系统的网络控制器,该网络控制器包括:被配置为存储用户界面的存储器;处理电路,被配置为呈现该用户界面并执行控制节点,其中该用户界面被配置为:图形地表示该软件定义网络(SDN)架构系统支持的网络的拓扑,该网络包括第一虚拟网络和第二虚拟网络;以及动态地生成表示虚拟网络路由器的图形元件,该虚拟网络路由器用于互连该第一虚拟网络和该第二虚拟网络,该虚拟网络路由器表示使得该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个的一个或多个策略的逻辑抽象,其中,该控制节点根据该一个或多个策略配置该第一虚拟网络和该第二虚拟网络,以实现经由该虚拟网络路由器在该第一虚拟网络和该第二虚拟网络之间引入和引出路由信息中的一个或多个。

[0422] 2.根据示例1的网络控制器,其中,该用户界面被配置为在图形地表示该网络的拓扑时呈现表示该第一虚拟网络的第一图形元件和表示该第二虚拟网络的第二图形元件。

[0423] 示例3.根据示例1和示例2的任何组合的网络控制器,其中该用户界面被配置为当图形地表示该网络的拓扑时:从用户接收指示该虚拟网络路由器的连通性类型的输入;选择与该虚拟网络路由器的连通性类型相关联的图形图标;以及动态地生成包括与虚拟网络路由器的连通性类型相关联的图形图标的图形元件。

[0424] 示例4.根据示例3的网络控制器,其中,该连通性类型包括网格连通性和中心辐条式连通性中的一个。

[0425] 示例5.根据示例3的网络控制器,其中,该连通性类型包括网格连通性,并且其中由该网格虚拟网络路由器表示的该一个或多个策略包括使得该第一虚拟网络与该第二虚拟网络之间的该路由信息的该引入和该引出两者的对称的引入和引出策略。

[0426] 示例6.根据示例3的网络控制器,其中该连通性类型包括中心辐条式连通性;并且其中由该中心虚拟网络路由器表示的该一个或多个策略包括非对称的引入和引出策略,该非对称的引入和引出策略使得将该路由信息从该第一辐条式虚拟网络和该第二辐条式虚拟网络两者引出到该虚拟网络路由器,而不在该第一辐条式虚拟网络和该第二辐条式虚拟网络之间引入该路由信息。

[0427] 示例7.根据示例1-6的任意组合的网络控制器,其中,该用户界面被配置为:当图形地表示该网络的拓扑时,从用户接收定义与该第一虚拟网络相关联的第一标签和与该第二虚拟网络相关联的第二标签的输入,并且其中,该处理电路执行配置节点,该配置节点基于该第一标签和该第二标签来识别:该第一虚拟网络和该第二虚拟网络用于根据该一个或多个策略来配置对应于该虚拟网络路由器的路由实例,以使得在该第一虚拟网络和该第二虚拟网络之间引入和引出该路由信息。

[0428] 示例8.根据示例1-7的任意组合的网络控制器,其中,该第一虚拟网络与第一命名空间相关联,其中,该第二虚拟网络与第二命名空间相关联,其中,该虚拟网络路由器是第

一虚拟网络路由器,其中,该一个或多个策略包括第一策略和第二策略,其中,该第一虚拟网络路由器表示用于该第一虚拟网络和第二虚拟网络路由器之间的路由信息的引入和引出的第一策略,并且其中该第二虚拟网络路由器表示用于该第二虚拟网络和该第一虚拟网络路由器之间的路由信息的引入和引出的第二策略。

[0429] 示例9.根据示例8的网络控制器,其中该第一虚拟网络路由器在部署该第一策略之前必须获得授权以将该第一虚拟网络路由器互连到该第二虚拟网络路由器。

[0430] 示例10.根据示例1-9的任意组合的网络控制器,其中,该网络控制器支持容器编排系统。

[0431] 示例11.根据示例1-10的任意组合的网络控制器,其中,该虚拟网络路由器是用于SDN架构的定制资源的实例,该定制资源用于SDN架构配置,并且其中,该处理电路执行定制资源控制器,以协调该SDN架构的状态与该虚拟网络路由器的预期状态,并且其中,该控制节点被配置为部署该虚拟网络路由器,以实现该虚拟网络路由器的该预期状态。

[0432] 示例12.根据示例1-11的任何组合的网络控制器,其中,该虚拟网络路由器是使用公共路由实例和公共路由目标来实现的,其中,该第一虚拟网络使用第一路由实例和第一路由目标来实现,其中,该第二虚拟网络是使用第二路由实例和第二路由目标来实现的,并且其中,该控制节点被配置为将该第一路由实例、该第二路由实例和该公共路由实例的该引入和该引出与该第一路由目标、该第二路由目标和该公共路由目标进行配置,以便实现网格连通性。

[0433] 示例13.根据示例1-12的任意组合的网络控制器,其中,使用第一多个计算节点的一个或多个第一虚拟路由器来实现该第一虚拟网络,并且其中,使用第二多个计算节点的一个或多个第二虚拟路由器来实现该第二虚拟网络。

[0434] 示例14.根据示例13的网络控制器,其中该第一多个计算节点是第一Kubernetes集群,并且其中该第二多个计算节点是第二Kubernetes集群。

[0435] 示例15.根据示例1-14的任意组合的网络控制器,其中,该控制节点将该路由信息解析为该第一虚拟网络和该第二虚拟网络中的一个或多个的转发信息,并将该转发信息安装在支持该第一虚拟网络和该第二虚拟网络之间的互连性的虚拟路由器中。

[0436] 示例16.一种方法,包括:由用于软件定义网络(SDN)架构系统的网络控制器呈现用户界面,其中该用户界面被配置为:图形地表示该软件定义网络(SDN)架构系统支持的网络的拓扑,该网络包括第一虚拟网络和第二虚拟网络;以及动态地生成表示虚拟网络路由器的图形元件,该虚拟网络路由器用于互连该第一虚拟网络和该第二虚拟网络,该虚拟网络路由器表示使得该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个的一个或多个策略的逻辑抽象;以及由该网络控制器执行控制节点,该控制节点根据该一个或多个策略来配置该第一虚拟网络和该第二虚拟网络,以实现经由该虚拟网络路由器在该第一虚拟网络和该第二虚拟网络之间引入和引出路由信息中的一个或多个。

[0437] 示例17.根据示例16的方法,其中图形地表示该网络的拓扑包括呈现表示该第一虚拟网络的第一图形元件和表示该第二虚拟网络的第二图形元件。

[0438] 示例18.根据示例16和17的任何组合的方法,其中图形地表示该网络的拓扑包括:从用户接收指示该虚拟网络路由器的连通性类型的输入;选择与该虚拟网络路由器的连通性类型相关联的图形图标;以及动态地生成包括与虚拟网络路由器的连通性类型相关联的

图形图标的图形元件。

[0439] 示例19.根据示例18的方法,其中,该连通性类型包括网格连通性和中心辐条式连通性中的一个。

[0440] 示例20.一种非暂时性计算机可读介质,该非暂时性计算机可读介质包括用于使软件定义网络(SDN)架构系统的网络控制器的处理电路执行以下操作的指令:呈现用户界面,其中该用户界面被配置为:图形地表示由该软件定义网络(SDN)架构系统支持的网络的拓扑,该网络包括第一虚拟网络和第二虚拟网络;以及动态地生成表示虚拟网络路由器的图形元件,通过该虚拟网络路由器来互连该第一虚拟网络和该第二虚拟网络,该虚拟网络路由器表示一个或多个策略的逻辑抽象,该一个或多个策略使得在该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个;以及执行控制节点,该控制节点根据该一个或多个策略来配置该第一虚拟网络和该第二虚拟网络,以实现经由该虚拟网络路由器在该第一虚拟网络和该第二虚拟网络之间的路由信息的引入和引出中的一个或多个。

[0441] 如果以硬件实现,则本公开可针对一种设备,诸如处理器或集成电路装置,诸如集成电路芯片或芯片组。可替代地或另外,如果以软件或固件实现,那么该技术可至少部分地由包括指令的计算机可读数据存储媒体实现,该指令在执行时致使处理器执行上文所描述的方法中的一个或多者。例如,计算机可读数据存储介质可以存储此类指令以供处理器执行。

[0442] 计算机可读介质可以形成计算机程序产品的一部分,该计算机程序产品可以包括封装材料。计算机可读介质可以包括计算机数据存储介质,诸如随机存取存储器(RAM)、只读存储器(ROM)、非易失性随机存取存储器(NVRAM)、电可擦除可编程只读存储器(EEPROM)、闪存、磁或光数据存储介质等。在一些示例中,制品可以包括一个或多个计算机可读存储介质。

[0443] 在一些示例中,计算机可读存储介质可以包括非暂时性介质。术语“非暂时性”可以指示存储介质不被体现在载波或传播信号中。在某些示例中,非暂时性存储介质可以存储可以随时间变化的数据(例如,在RAM或高速缓存中)。

[0444] 代码或指令可以是由处理电路执行的软件和/或固件,该处理电路包括一个或多个处理器,诸如一个或多个数字信号处理器(DSP)、通用微处理器、专用集成电路(ASIC)、现场可编程门阵列(FPGA)或其他等效的集成或离散逻辑电路。因此,如本文中所使用的术语“处理器”可指代前述结构中的任一个或适合于实现本文所述的技术的任何其他结构。另外,在一些方面中,本公开所述的功能性可提供于软件模块或硬件模块内。

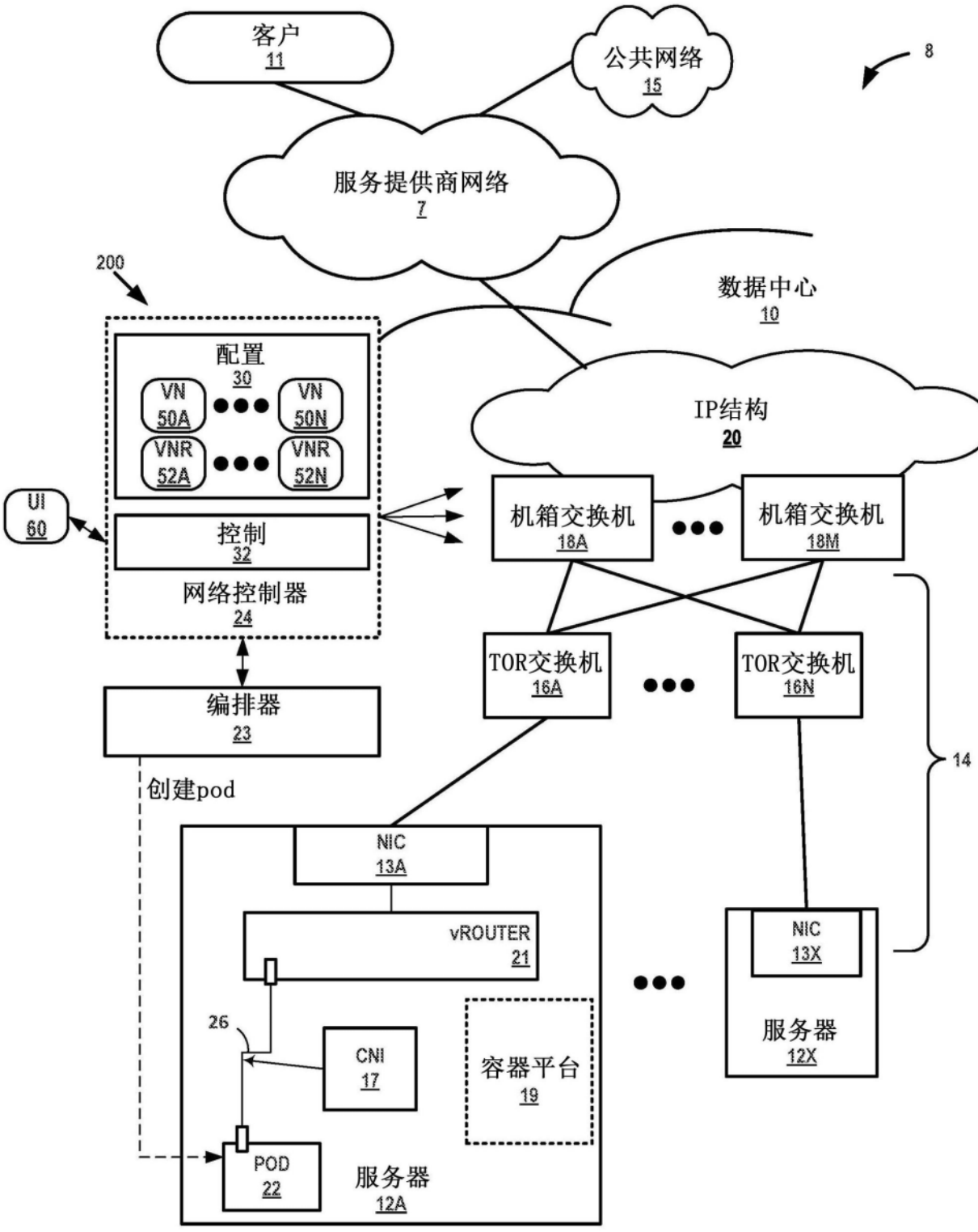

客户
11
公共网络
15
8
服务提供商网络
7
200
数据中心
10
配置
30
VN
50A
VN
50N
VNR
52A
VNR
52N
IP结构
20
UI
60
控制
32
网络控制器
24
机箱交换机
18A
机箱交换机
18M
TOR交换机
16A
TOR交换机
16N
编排器
23
14
创建pod
NIC
13A
vROUTER
21
NIC
13X
26
CNI
17
容器平台
19
服务器
12X
POD
22
服务器
12A

图1

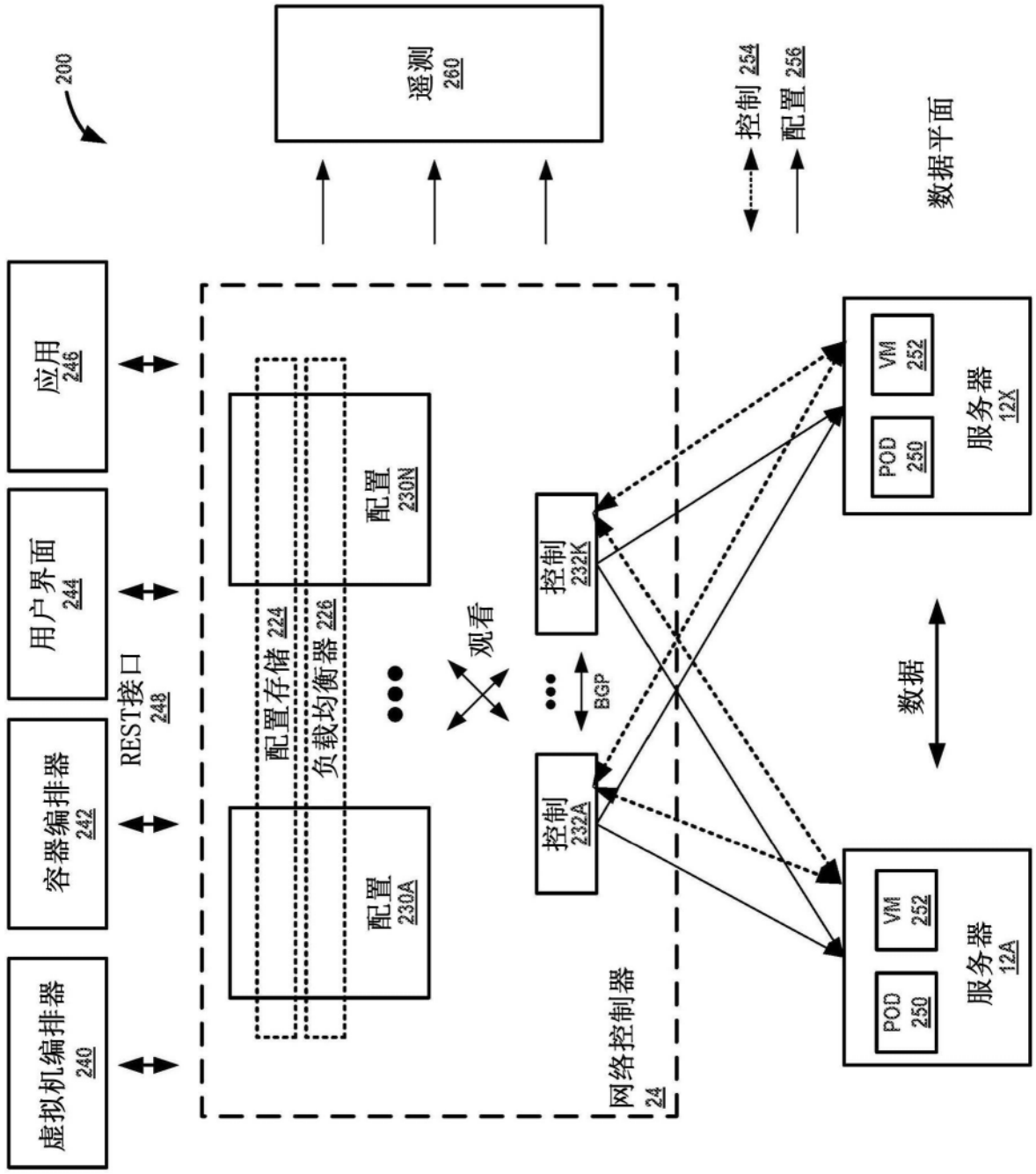

200
遥测
260
控制 254
配置 256
数据平面
应用
246
用户界面
244
容器编排器
242
虚拟机编排器
240
REST接口
248
配置存储 224
负载均衡器 226
配置
230N
配置
230A
观看
控制
232K
控制
232A
BGP
网络控制器
24
数据
VM
252
POD
250
服务器
12X
VM
252
POD
250
服务器
12A

图2

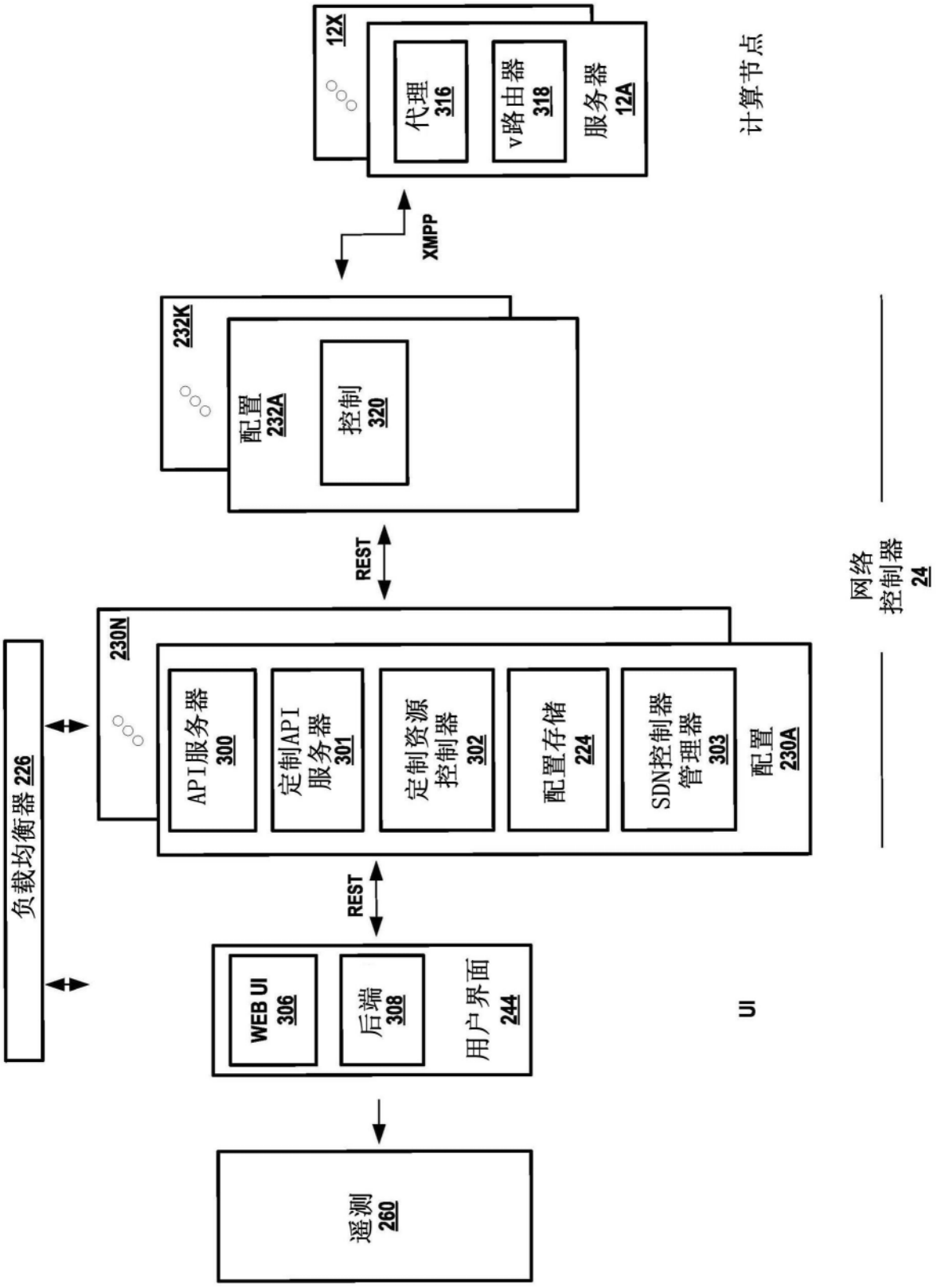

12X
代理 316
v路由器 318
服务器 12A
计算节点
XMPP
232K
配置 232A
控制 320
REST
网络控制器 24
230N
API服务器 300
定制API服务器 301
定制资源控制器 302
配置存储 224
SDN控制器管理器 303
配置 230A
负载均衡器 226
REST
WEB UI 306
后端 308
用户界面 244
UI
遥测 260

图3

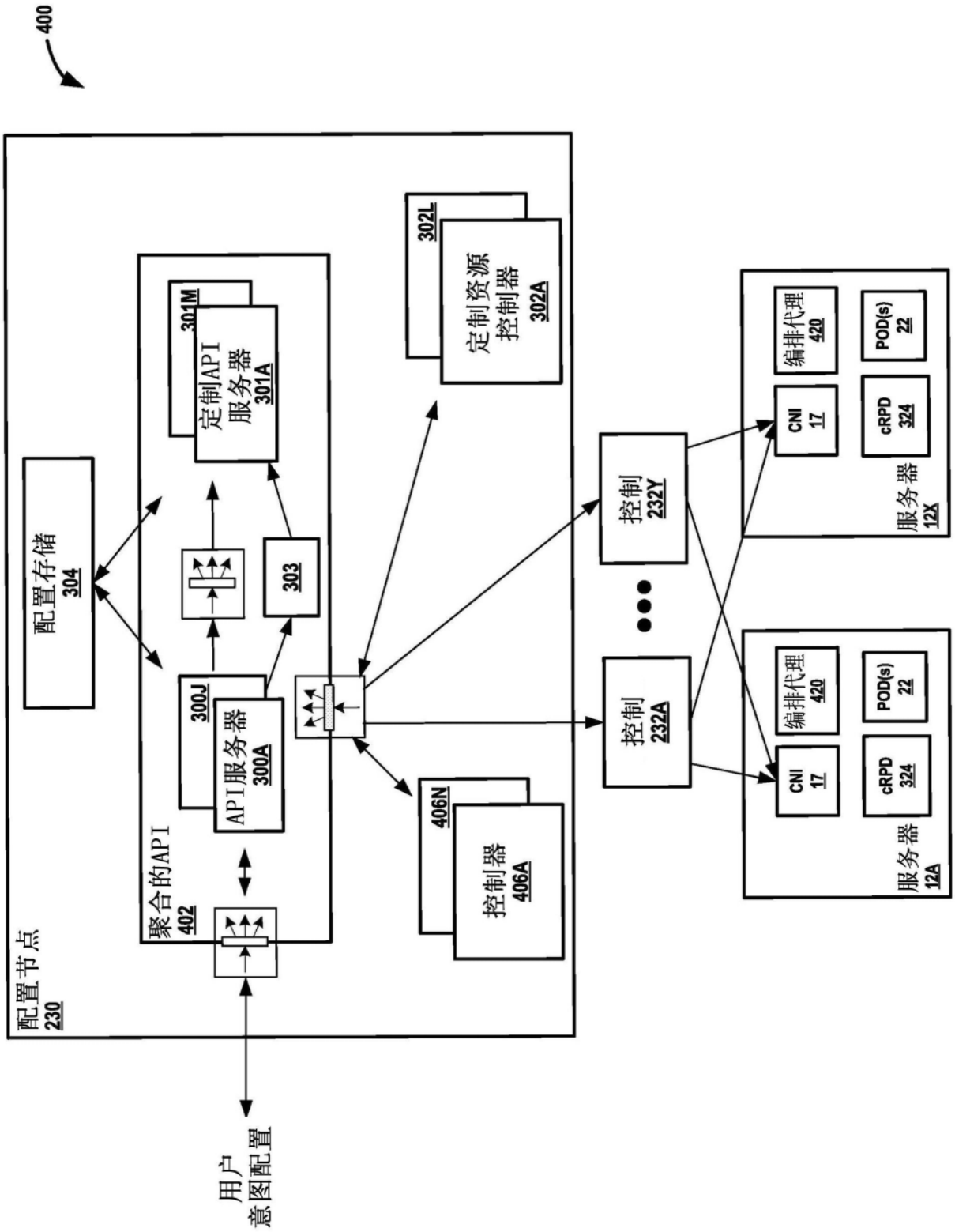

400
配置节点
230
聚合的API
402
配置存储
304
API服务器
300A
300J
定制API
服务器
301A
301M
303
定制资源
控制器
302A
302L
控制器
406A
406N
控制
232A
控制
232Y
服务器
12A
服务器
12X
编排代理
420
CNI
17
cRPD
324
POD(s)
22
用户
意图配置

图4

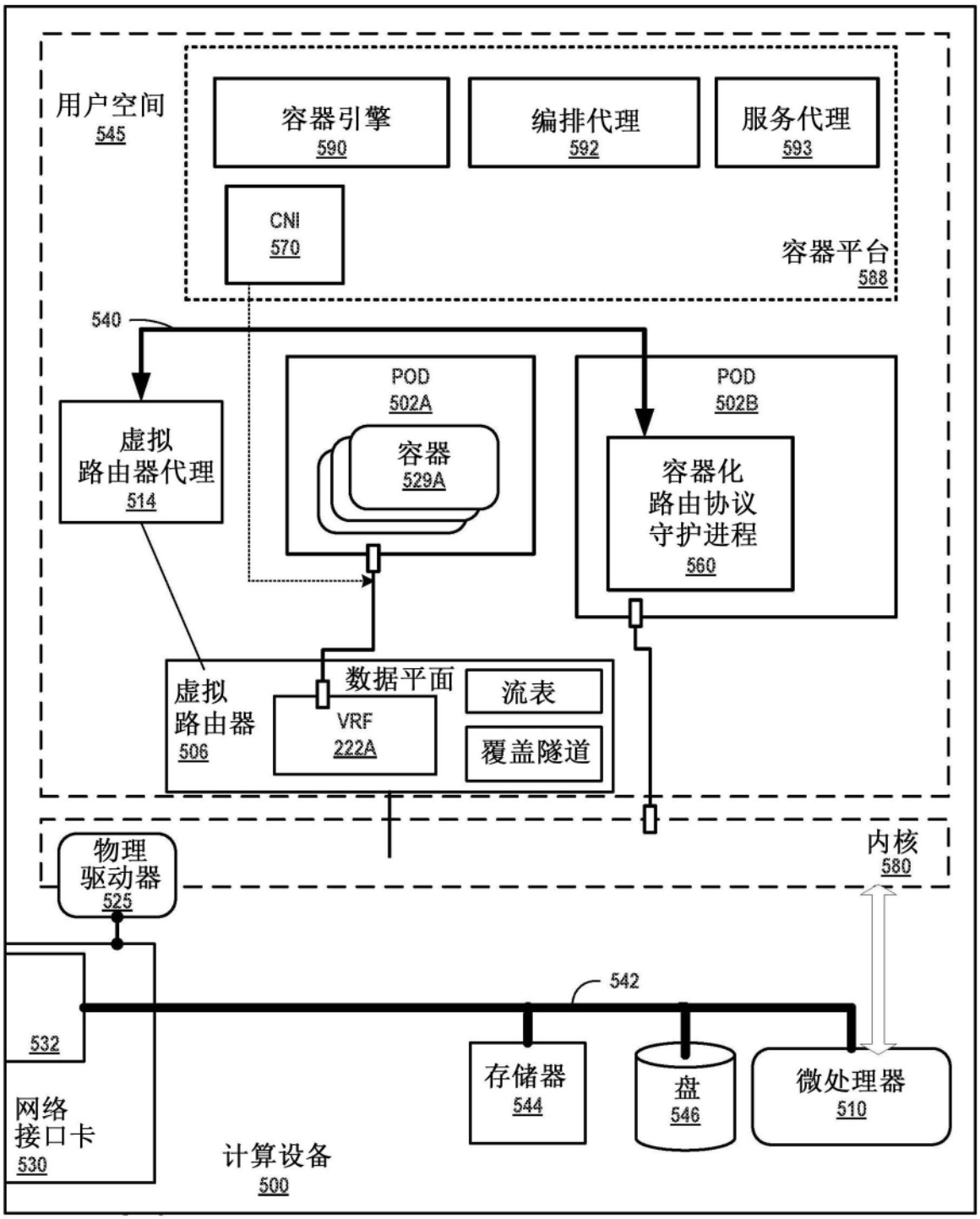

用户空间
545
容器引擎
590
编排代理
592
服务代理
593
CNI
570
容器平台
588
540
虚拟
路由器代理
514
POD
502A
容器
529A
POD
502B
容器化
路由协议
守护进程
560
虚拟
路由器
506
数据平面
VRF
222A
流表
覆盖隧道
物理
驱动器
525
内核
580
542
532
网络
接口卡
530
计算设备
500
存储器
544
盘
546
微处理器
510

图5

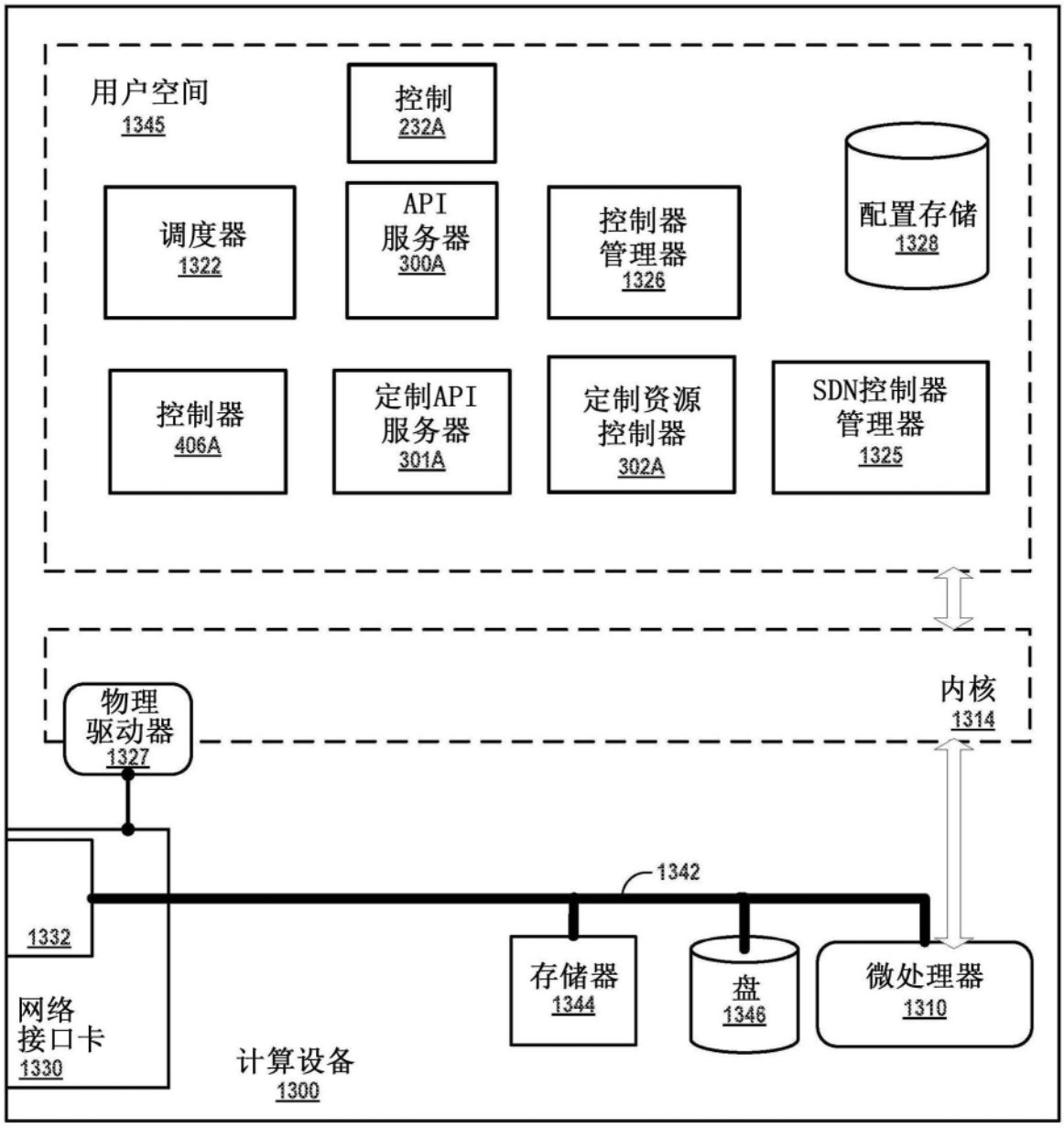

用户空间
1345
控制
232A
调度器
1322
API
服务器
300A
控制器
管理器
1326
配置存储
1328
控制器
406A
定制API
服务器
301A
定制资源
控制器
302A
SDN控制器
管理器
1325
内核
1314
物理
驱动器
1327
1342
1332
网络
接口卡
1330
存储器
1344
盘
1346
微处理器
1310
计算设备
1300

图6

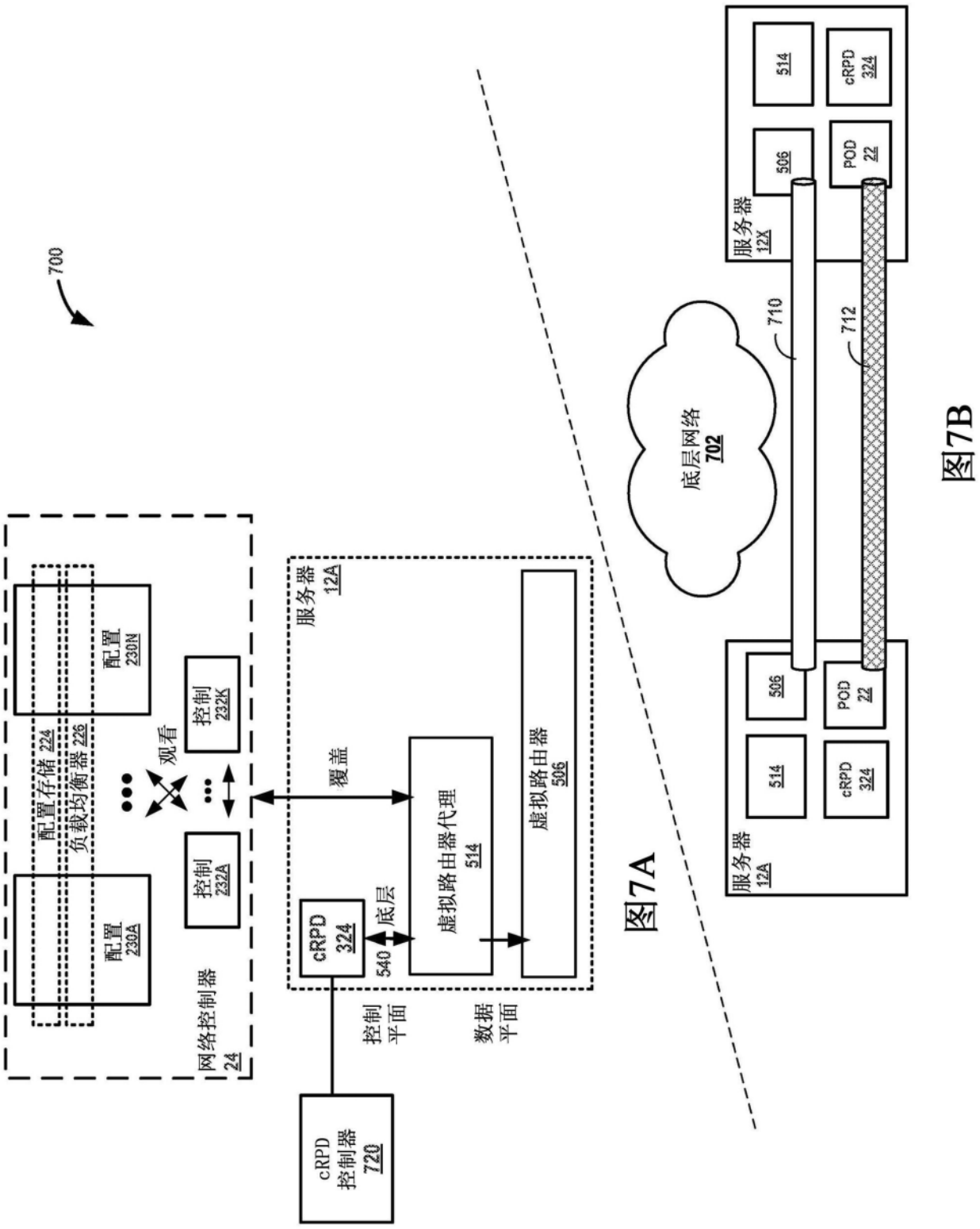

700
网络控制器 24
配置 230A
负载均衡器 226
配置存储 224
配置 230N
控制 232A
观看
控制 232K
服务器 12A
覆盖
cRPD 324
底层
540
虚拟路由器代理 514
虚拟路由器 506
控制平面
数据平面
cRPD 控制器 720
底层网络 702
710
712
服务器 12X
506
514
POD 22
cRPD 324
服务器 12A
506
514
POD 22
cRPD 324

图7A

图7B

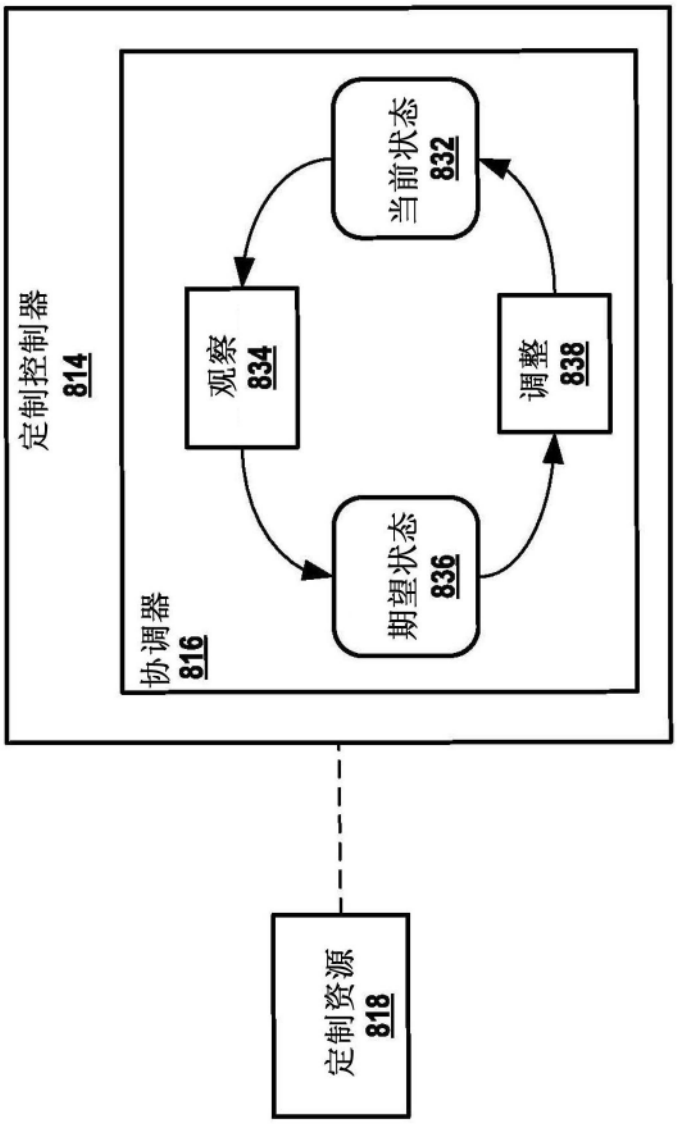


图8

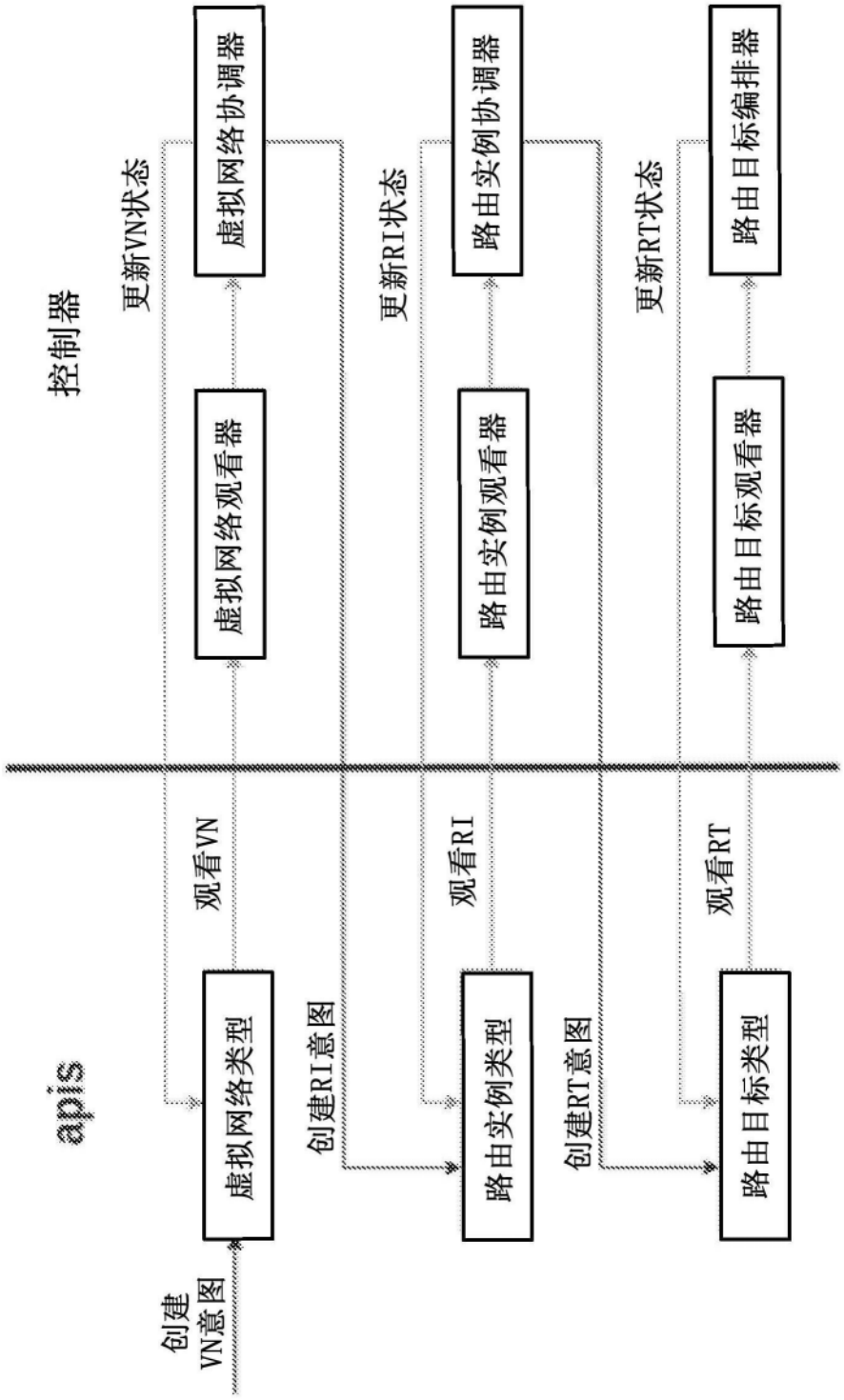

控制器
更新VN状态
虚拟网络协调器
虚拟网络观看器
更新RI状态
路由实例协调器
路由实例观看器
更新RT状态
路由目标编排器
路由目标观看器
APIs
观看VN
观看RI
观看RT
创建RI意图
创建RT意图
虚拟网络类型
路由实例类型
路由目标类型
创建
VN意图

图9

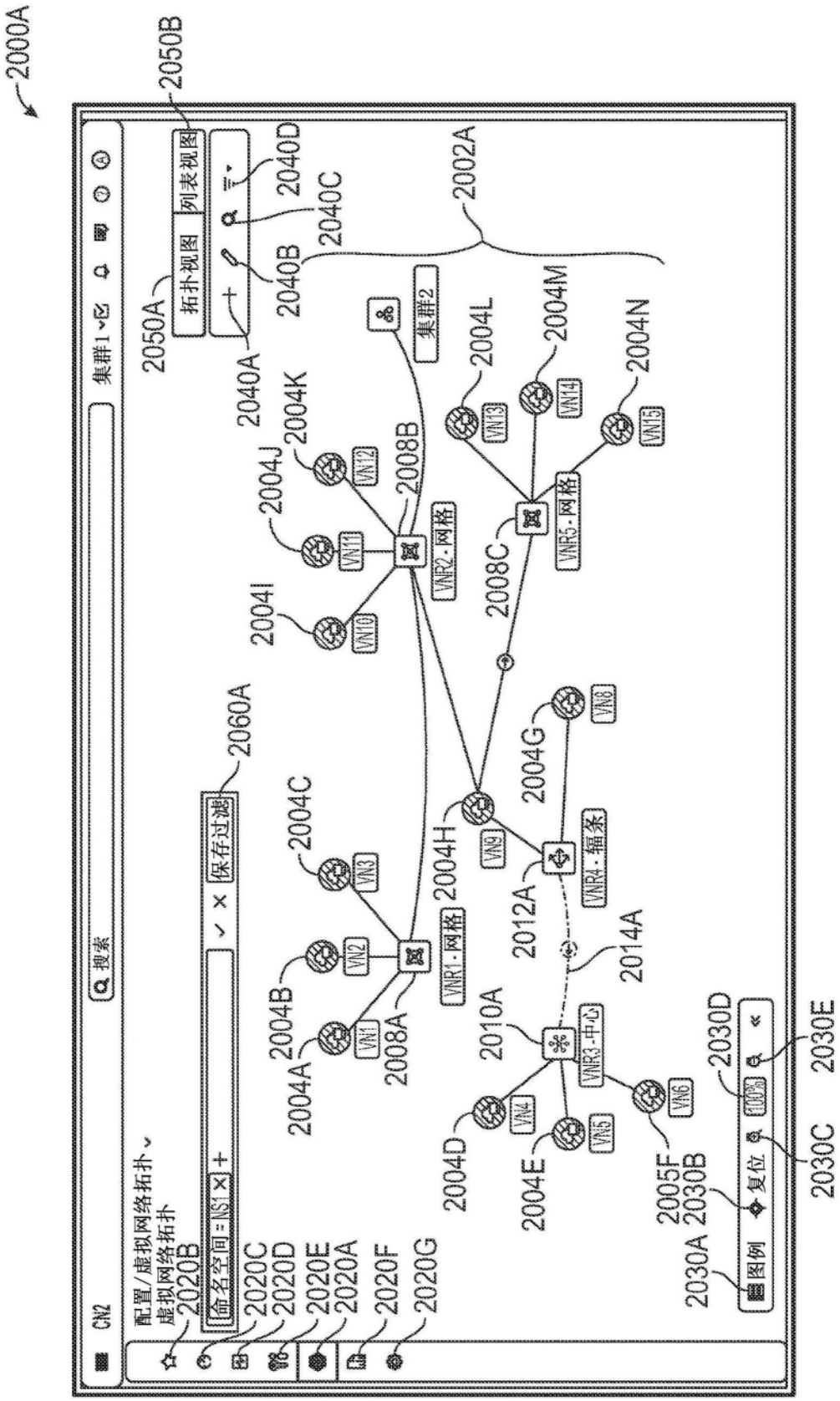

2000A
2050B
CN2
搜索
集群1
配置/虚拟网络拓扑
虚拟网络拓扑
2020B
命名空间 = NS1
保存过滤
2060A
拓扑视图
列表视图
2050A
2040A
2040B
2040C
2040D
2002A
集群2
2004K
2004J
2004I
2004L
2004M
2004N
2008B
2008C
VNR2 - 网格
VNR5 - 网格
VN10
VN11
VN12
VN13
VN14
VN15
2004C
2004H
2004G
VN3
VN9
VN8
2004B
2004A
VN2
VN1
2008A
VNR1 - 网格
2012A
VNR4 - 福条
2014A
2010A
VNR3 - 中心
2004D
2004E
VN4
VN5
VN6
2005F
2030B
2030A
2030D
2030E
2030C
图例
复位
100%
2020C
2020D
2020E
2020A
2020F
2020G

图10A

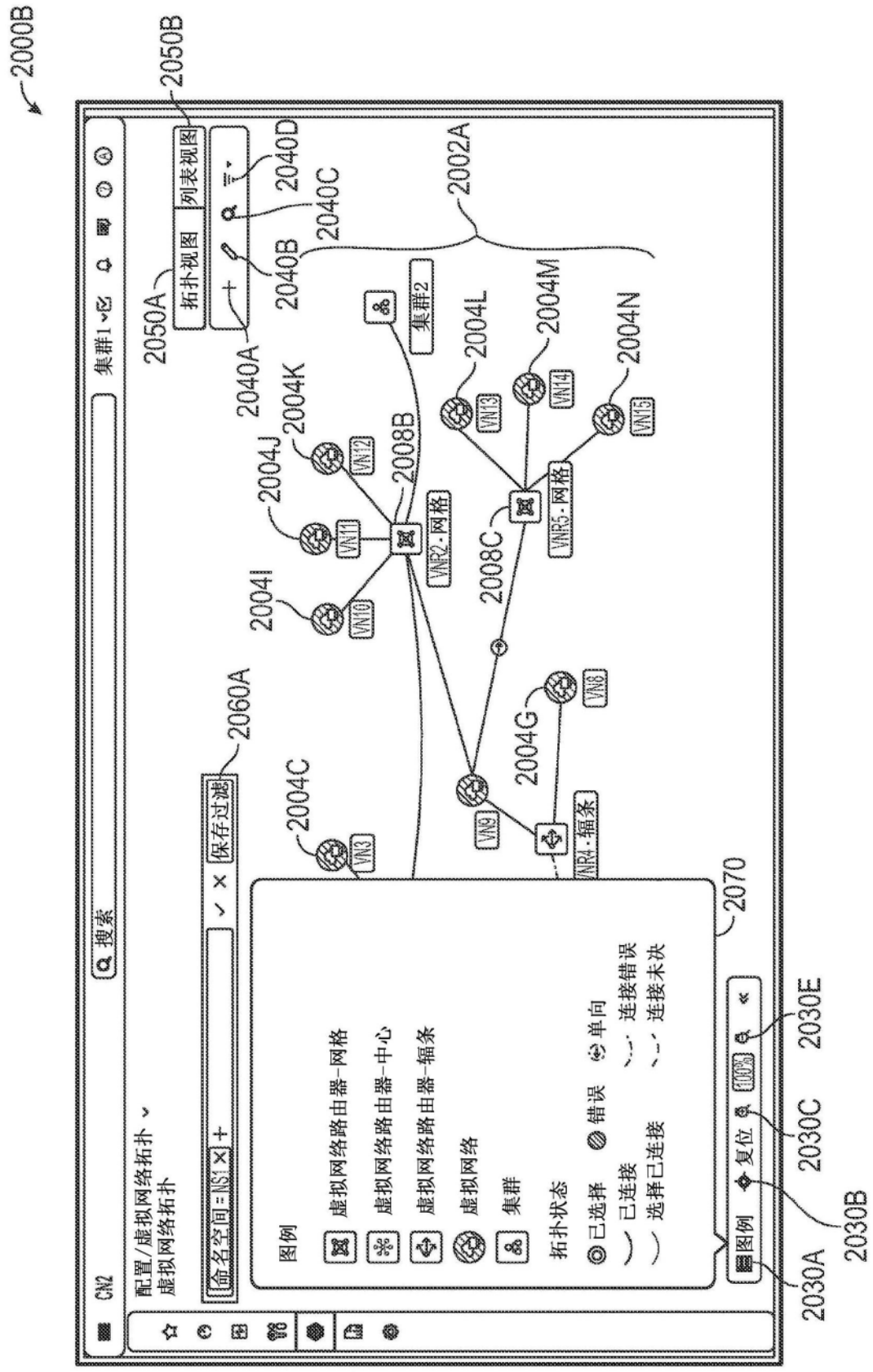

2000B
2050B
CN2
搜索
集群1
配置/虚拟网络拓扑
虚拟网络拓扑
拓扑视图
列表视图
2050A
2040A
2040B
2040C
2040D
2002A
命名空间 = NS1
保存过滤
2060A
集群2
2004K
2004J
2004I
2004L
2004M
2004N
2008B
2008C
VN12
VN11
VN10
VN13
VN14
VN15
VNR2 - 网格
VNR5 - 网格
2004C
VN3
2004G
VN8
VN9
VNR4 - 辐条
2070
图例
虚拟网络路由器 - 网格
虚拟网络路由器 - 中心
虚拟网络路由器 - 辐条
虚拟网络
集群
拓扑状态
已选择
错误
单向
已连接
连接错误
选择已连接
连接未决
图例
复位
100%
2030A
2030B
2030C
2030E

图10B

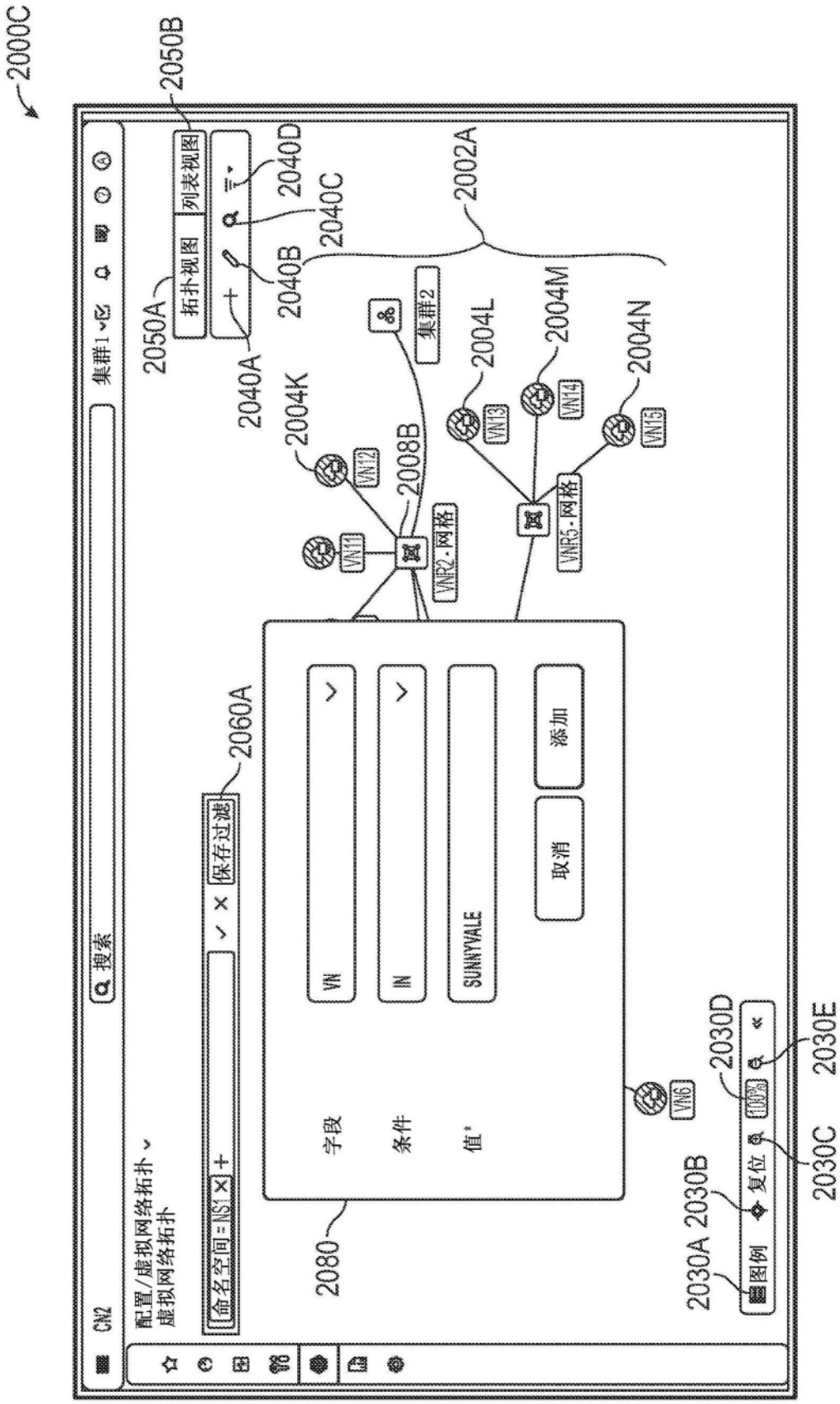

2000C
2050B
列表视图
拓扑视图
2050A
2040A
2040B
2040C
2040D
2002A
集群1
搜索
CN2
配置/虚拟网络拓扑
虚拟网络拓扑
命名空间=NS1
保存过滤
2060A
2004K
2004L
2004M
2004N
VN11
VN12
VN13
VN14
VN15
VN6
2008B
VNR2-网格
VNR5-网格
集群2
字段
条件
值'
VN
IN
SUNNYVALE
取消
添加
2080
2030A
2030B
2030C
2030D
2030E
图例
复位
100%

图10C

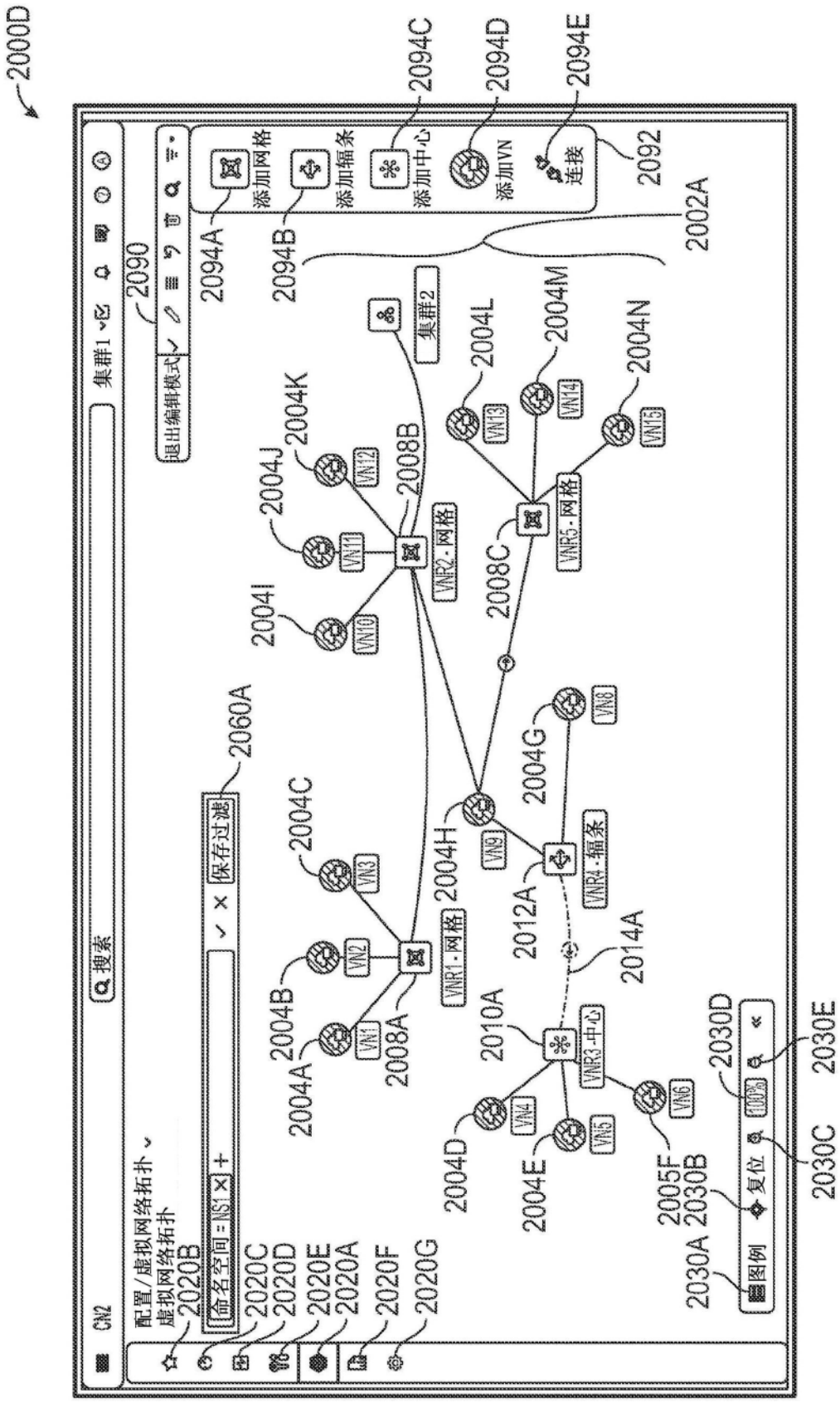

2000D
CN2
搜索
集群1
配置/虚拟网络拓扑
虚拟网络拓扑
2020B
命名空间 = NS1
保存过滤
2060A
2020C
2020D
2020E
2020A
2020F
2020G
2090
退出编辑模式
2094A
添加网格
2094B
添加辐条
2094C
添加中心
2094D
添加VN
2094E
连接
2092
2002A
集群2
2004A
2004B
2004C
2004D
2004E
2004F
2004G
2004H
2004I
2004J
2004K
2004L
2004M
2004N
VN1
VN2
VN3
VN4
VN5
VN6
VN8
VN9
VN10
VN11
VN12
VN13
VN14
VN15
2008A
2008B
2008C
VNR1 - 网格
VNR2 - 网格
VNR5 - 网格
2010A
2012A
2014A
VNR3 - 中心
VNR4 - 辐条
2030A
图例
2030B
复位
2030C
2030D
2030E
2005F

图10D

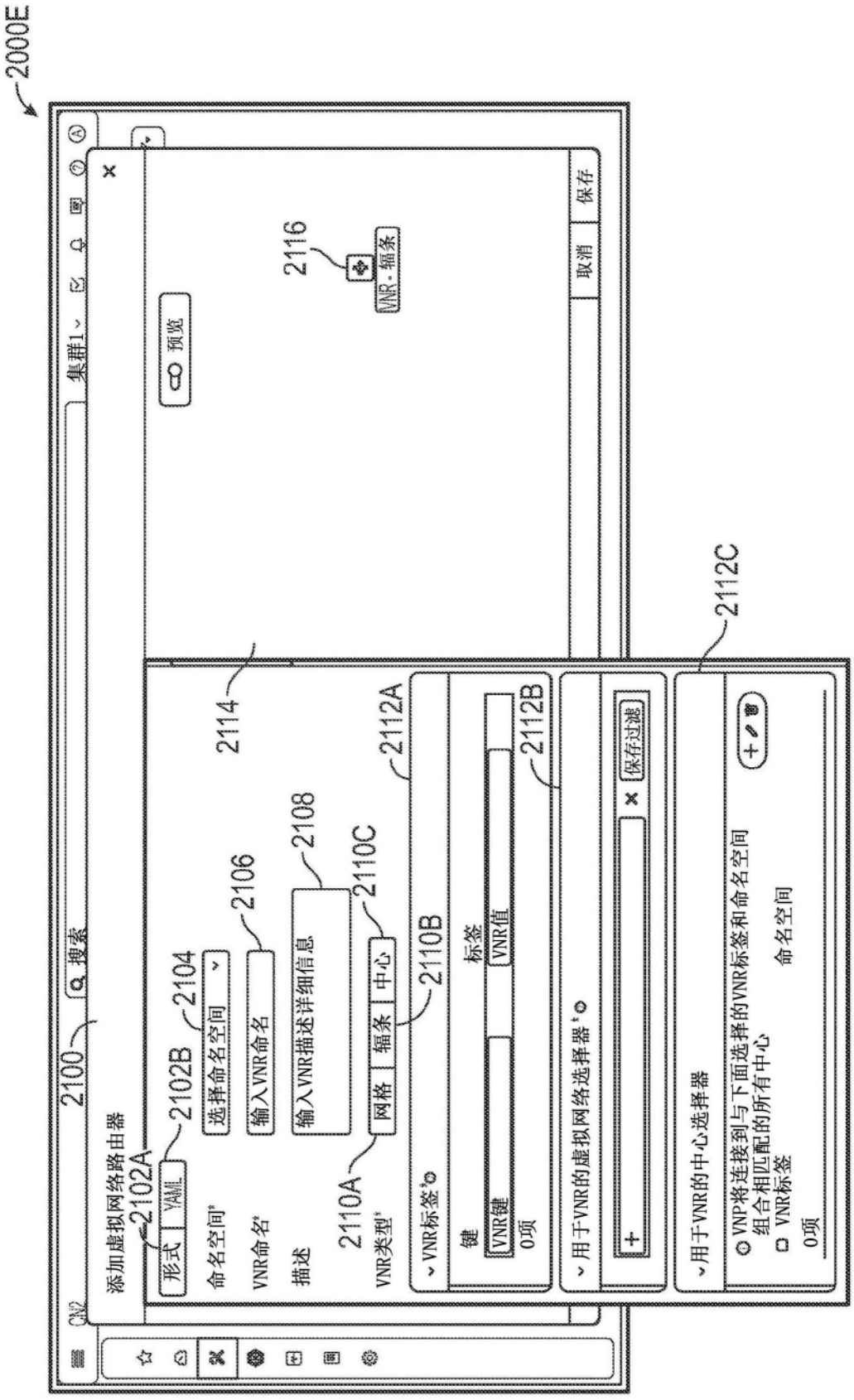

2000E
2100
搜索
集群1
添加虚拟网络路由器
2102A
形式
YAML
2102B
命名空间'
选择命名空间
2104
VNR命名'
输入VNR命名
2106
描述
输入VNR描述详细信息
2108
VNR类型'
网格
辐条
中心
2110A
2110B
2110C
2112A
VNR标签'
键
标签
VNR键
VNR值
0项
2112B
用于VNR的虚拟网络选择器'
保存过滤
2112C
用于VNR的中心选择器
VNP将连接到与下面选择的VNR标签和命名空间组合相匹配的所有中心
VNR标签
命名空间
0项
2114
预览
2116
VNR-辐条
取消
保存

图10E

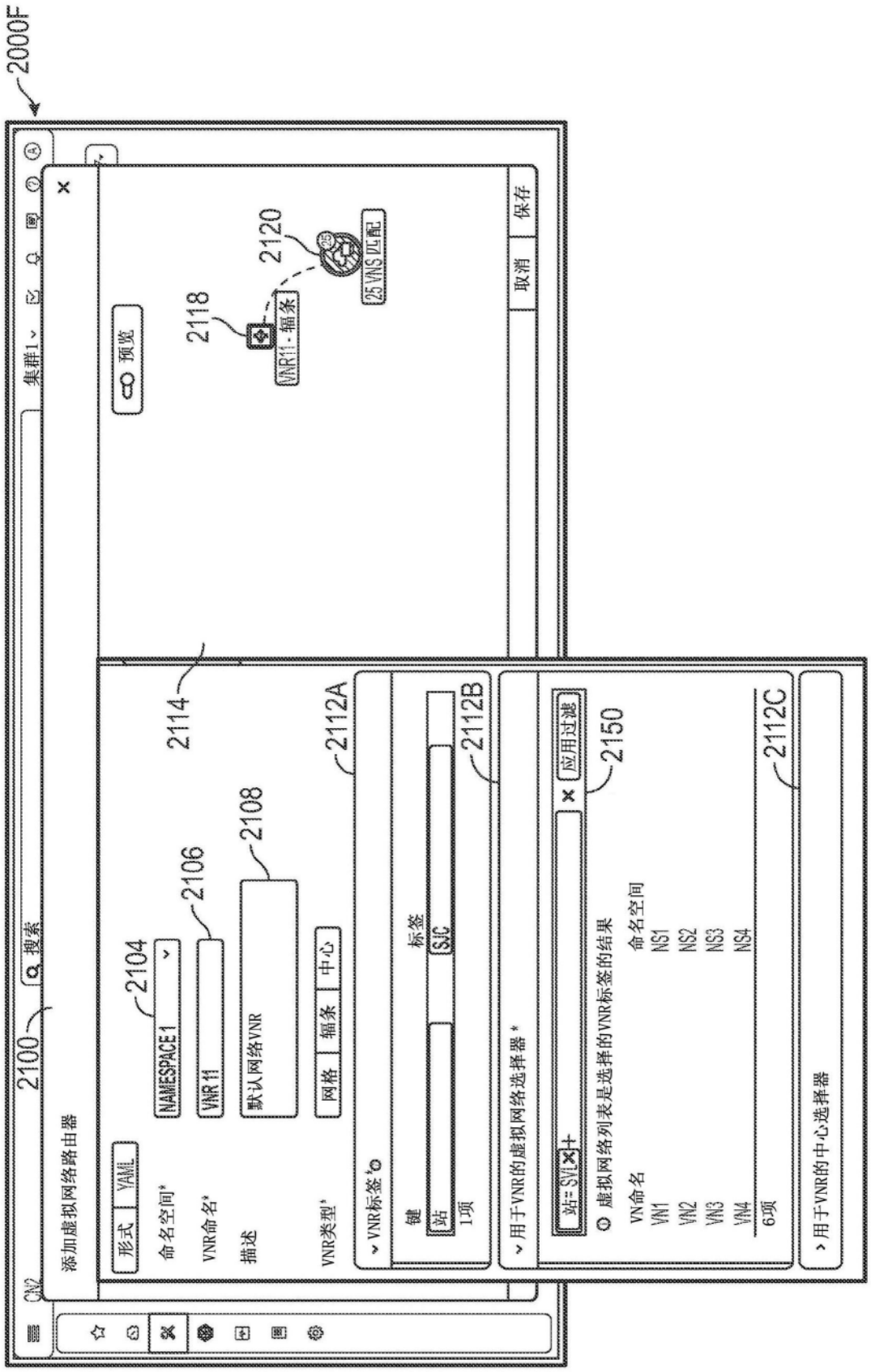

2000F
2100
添加虚拟网络路由器
形式
YAML
命名空间'
VNR命名'
描述
VNR类型'
NAMESPACE1
VNR 11
默认网络VNR
网格
辐条
中心
2104
2106
2108
2114
2112A
VNR标签'
键
标签
站
SJC
1项
2112B
用于VNR的虚拟网络选择器'
站= SVL
应用过滤
2150
虚拟网络列表是选择的VNR标签的结果
VN命名
命名空间
VN1
NS1
VN2
NS2
VN3
NS3
VN4
NS4
6项
2112C
用于VNR的中心选择器
预览
2118
VNR11 - 辐条
2120
25 VNS 匹配
取消
保存

图10F

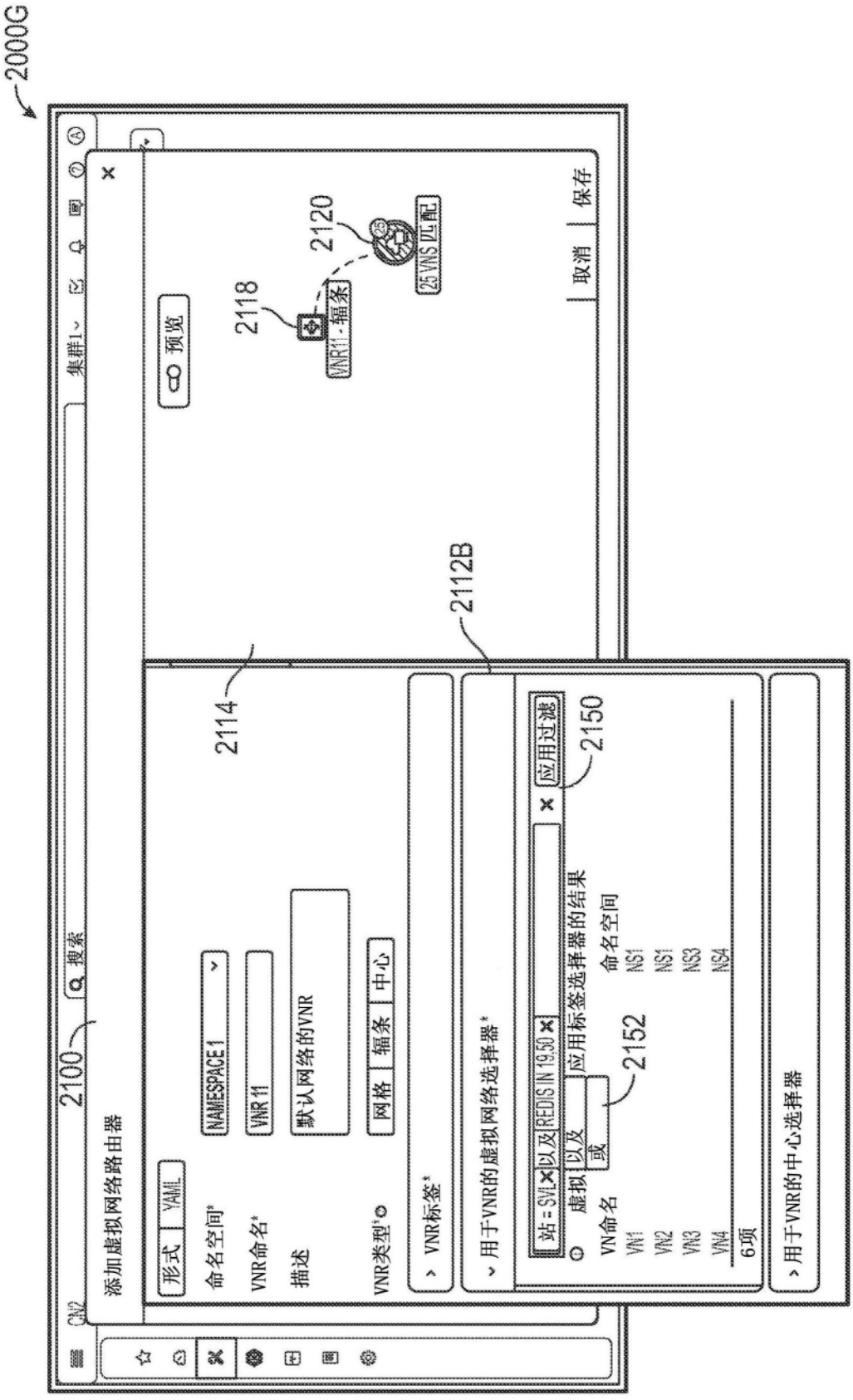

2000G
2100
添加虚拟网络路由器
搜索
集群1
形式
YAML
命名空间*
NAMESPACE1
VNR命名*
VNR 11
描述
默认网络的VNR
VNR类型*
网格
辐条
中心
VNR标签*
用于VNR的虚拟网络选择器*
站 = SVL 以及 REDIS IN 19.50
虚拟
以及
或
应用过滤
应用标签选择器的结果
VN命名
命名空间
VN1 NS1
VN2 NS1
VN3 NS3
VN4 NS4
6项
用于VNR的中心选择器
预览
VNR11 - 辐条
25 VNS 匹配
取消
保存
2114
2112B
2118
2120
2150
2152

图10G

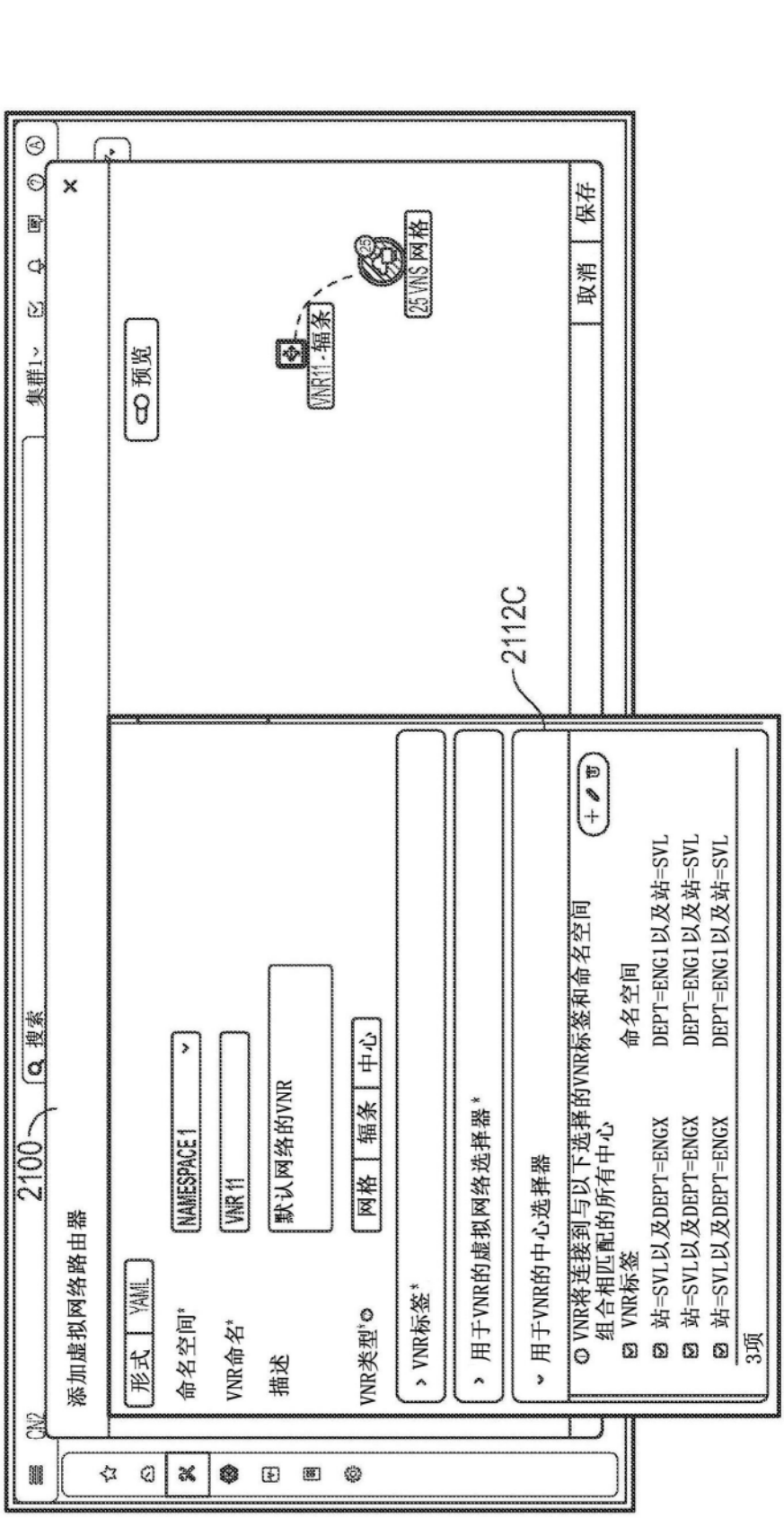

2000H
2100
搜索
集群1
添加虚拟网络路由器
形式
YAML
命名空间*
NAMESPACE1
VNR命名*
VNR 11
描述
默认网络的VNR
VNR类型*
网格
辐条
中心
VNR标签*
用于VNR的虚拟网络选择器*
用于VNR的中心选择器
VNR将连接到与以下选择的VNR标签和命名空间组合相匹配的所有中心
VNR标签
命名空间
站=SVL以及DEPT=ENGX
DEPT=ENG1以及站=SVL
站=SVL以及DEPT=ENGX
DEPT=ENG1以及站=SVL
站=SVL以及DEPT=ENGX
DEPT=ENG1以及站=SVL
3项
2112C
预览
VNR11 - 辐条
25 VNS 网格
取消
保存

图10H

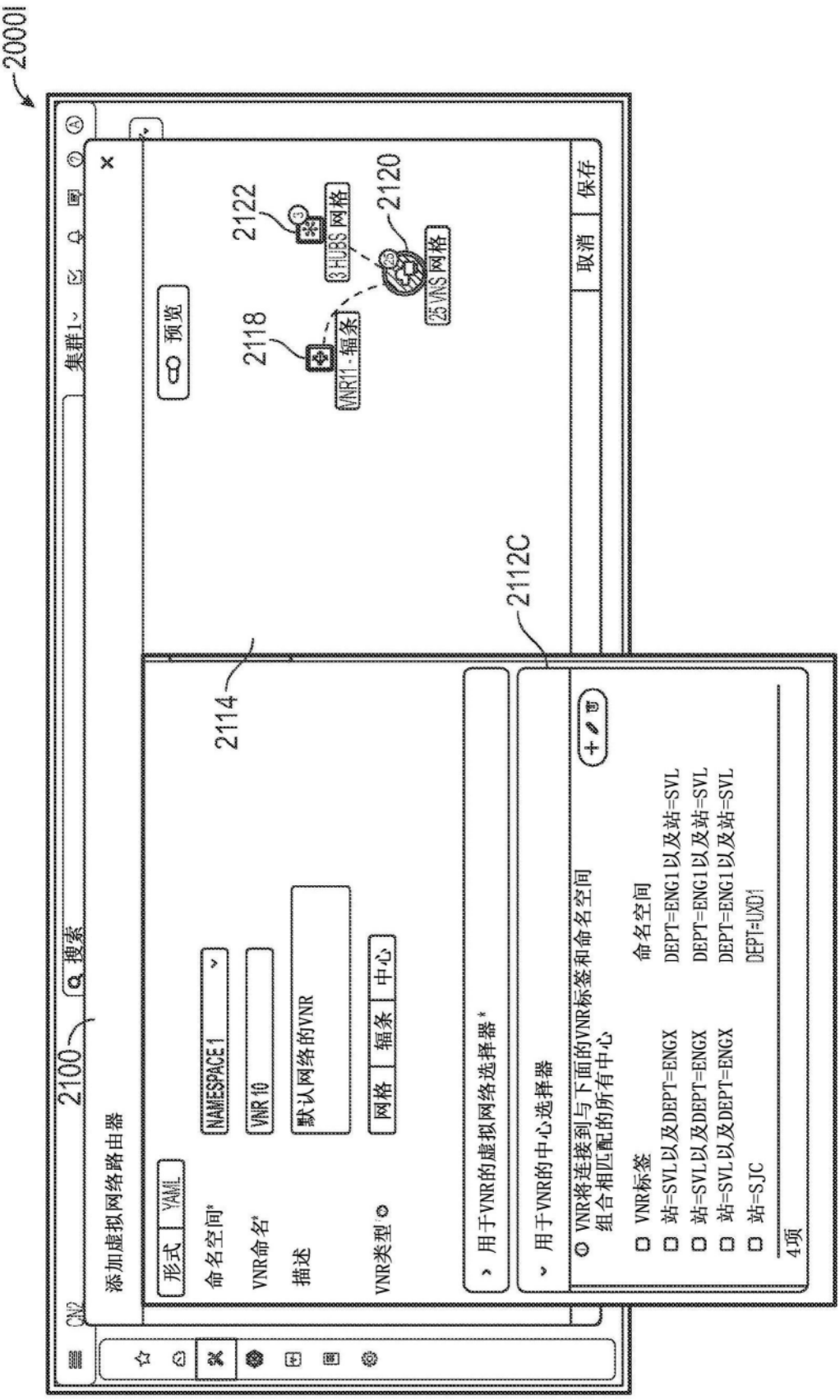

20001
2100
CN2
搜索
集群1
添加虚拟网络路由器
形式
YAML
命名空间*
NAMESPACE1
VNR命名*
VNR 1@
描述
默认网络的VNR
VNR类型 ⓘ
网格
辐条
中心
用于VNR的虚拟网络选择器*
用于VNR的中心选择器
ⓘ VNR将连接到与下面的VNR标签和命名空间组合相匹配的所有中心
VNR标签
命名空间
站=SVL以及DEPT=ENGX
DEPT=ENG1以及站=SVL
站=SVL以及DEPT=ENGX
DEPT=ENG1以及站=SVL
站=SVL以及DEPT=ENGX
DEPT=ENG1以及站=SVL
站=SJC
DEPT=LXD1
4项
预览
2114
2112C
2118
VNR11 - 辐条
2122
3 HUBS 网格
2120
25 VNS 网格
取消
保存

图10I

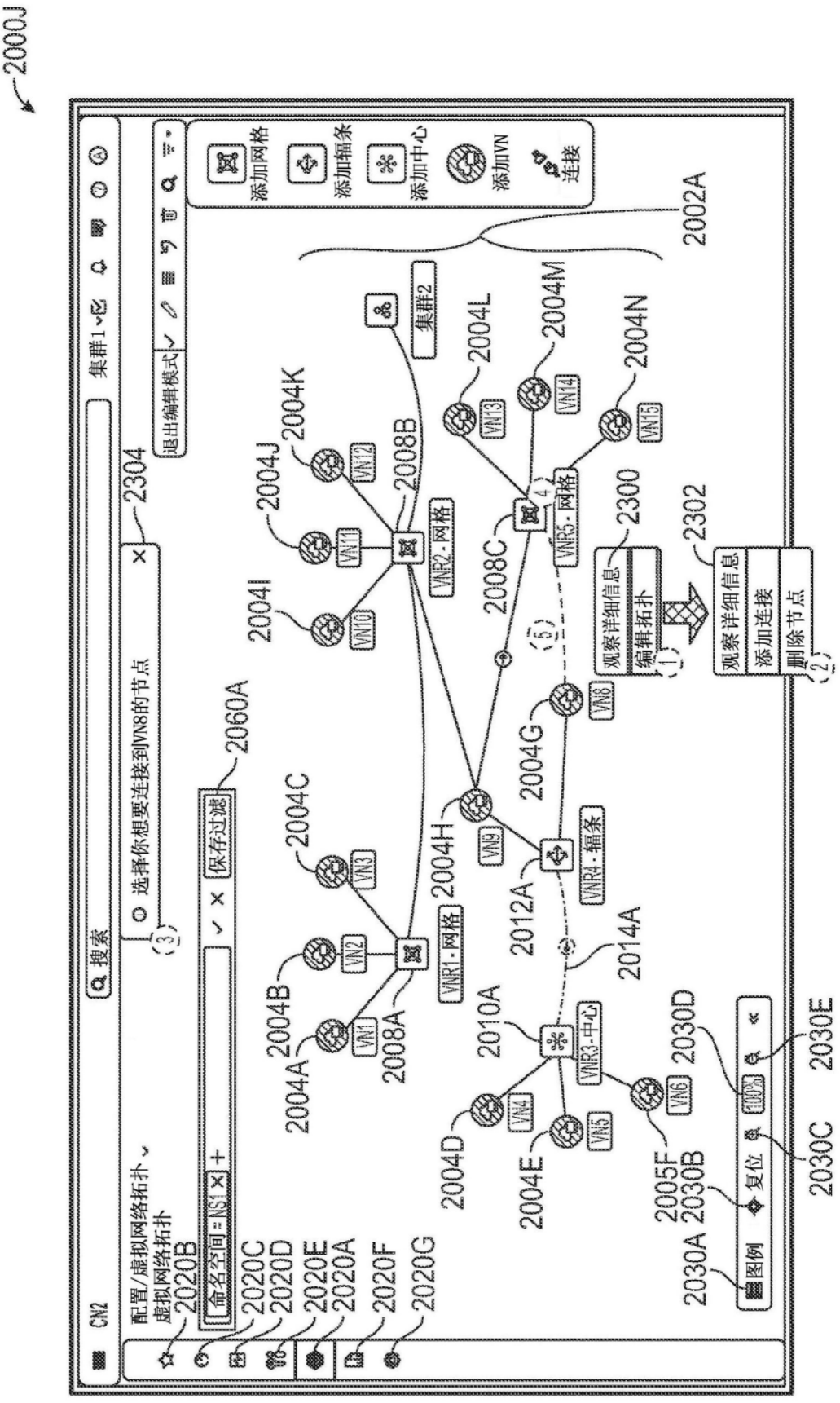

2000J
CN2
搜索
集群1
配置/虚拟网络拓扑
虚拟网络拓扑
2020B
选择你想要连接到VN8的节点
2304
退出编辑模式
添加网格
添加辐条
添加中心
添加VN
连接
2002A
命名空间 = NS1
保存过滤
2060A
2020C
2020D
2020E
2020A
2020F
2020G
2004A
2004B
2004C
2004D
2004E
2004F
2004G
2004H
2004I
2004J
2004K
2004L
2004M
2004N
VN1
VN2
VN3
VN4
VN5
VN6
VN7
VN8
VN9
VN10
VN11
VN12
VN13
VN14
VN15
VNR1 - 网格
VNR2 - 网格
VNR3 - 中心
VNR4 - 辐条
VNR5 - 网格
2008A
2008B
2008C
2010A
2012A
2014A
集群2
2300
2302
观察详细信息
编辑拓扑
添加连接
删除节点
图例
复位
2030A
2030B
2030C
2030D
2030E
2005F

图10J

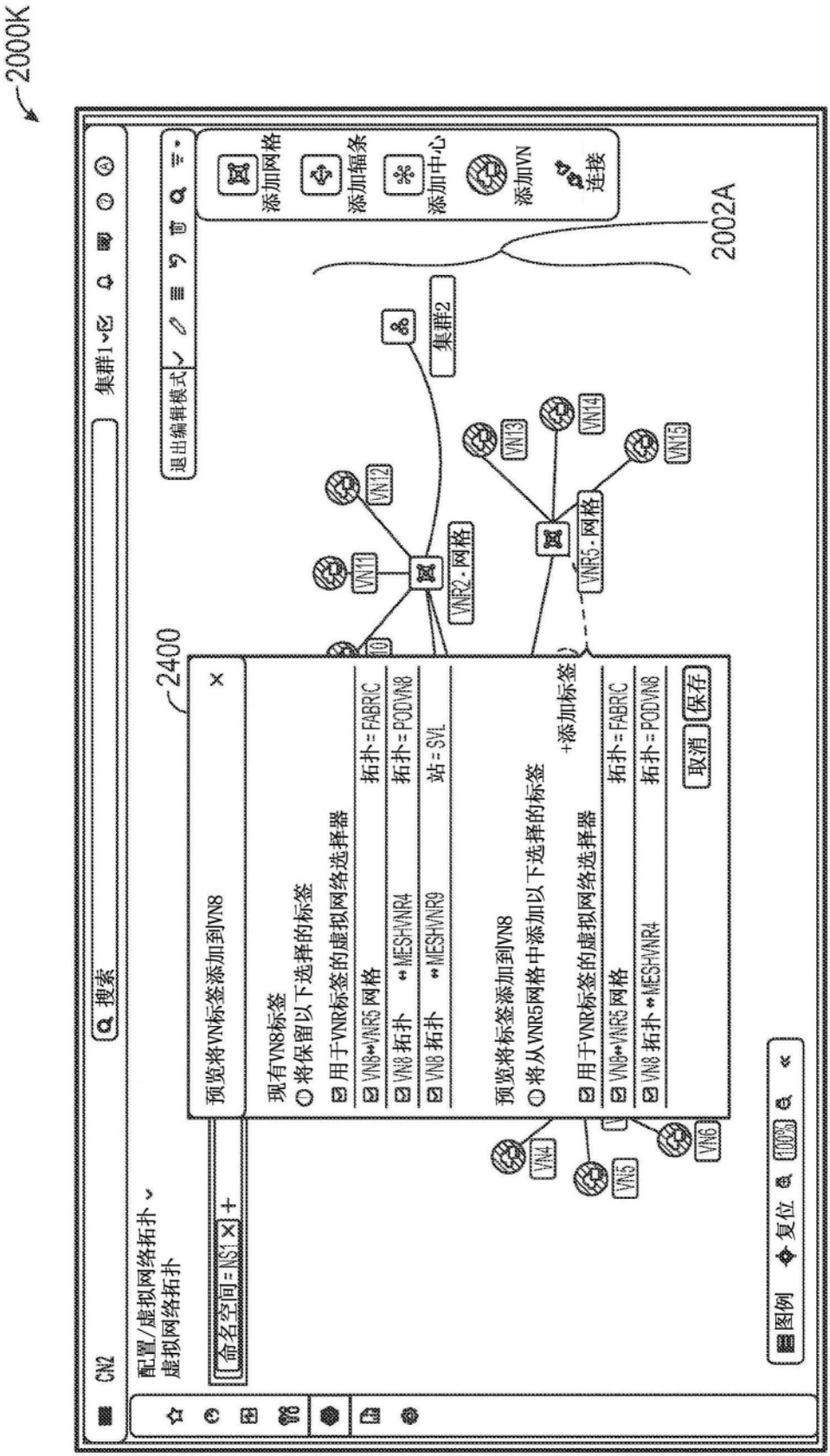

2000K
CN2
搜索
集群1
配置/虚拟网络拓扑
虚拟网络拓扑
命名空间 = NS1
退出编辑模式
添加网格
添加辐条
添加中心
添加VN
连接
集群2
2002A
2400
预览将VN标签添加到VN8
现有VN8标签
将保留以下选择的标签
用于VNR标签的虚拟网络选择器
VN8↔VNR5 网格
VN8 拓扑 ↔ MESHVNR4
VN8 拓扑 ↔ MESHVNR9
拓扑 = FABRIC
拓扑 = PODVN8
站 = SVL
预览将标签添加到VN8
将从VNR5网格中添加以下选择的标签
+添加标签
取消
保存
VNR2 - 网格
VNR5 - 网格
VN4
VN5
VN6
VN11
VN12
VN13
VN14
VN15
图例
复位
100%

图10K

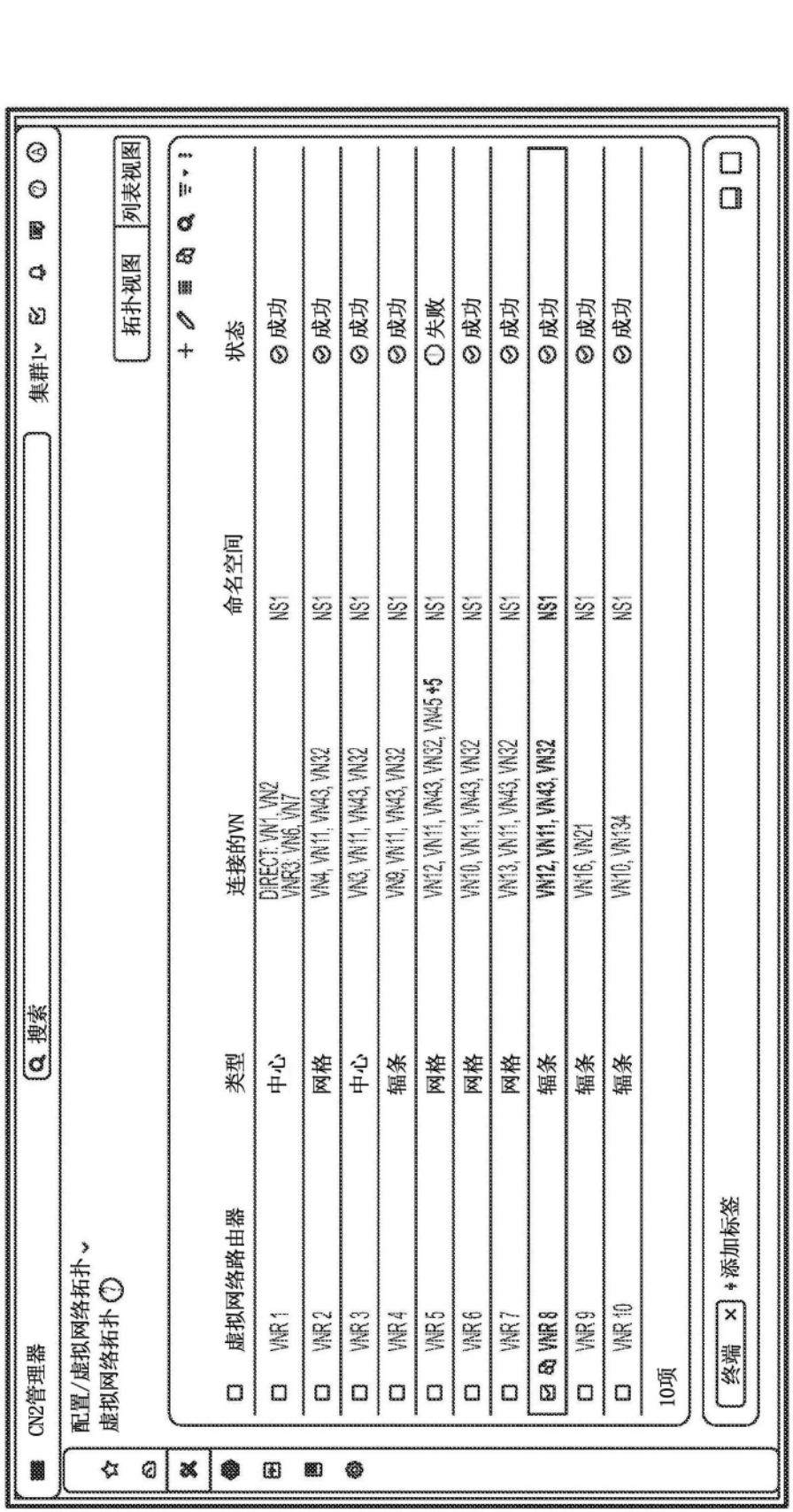

2000L
CN2管理器
搜索
集群1
配置/虚拟网络拓扑
虚拟网络拓扑
拓扑视图
列表视图
虚拟网络路由器
类型
连接的VN
命名空间
状态
VNR 1
中心
DIRECT: VN1, VN2
VNR3: VN6, VN7
NS1
成功
VNR 2
网格
VN4, VN11, VN43, VN32
NS1
成功
VNR 3
中心
VN3, VN11, VN43, VN32
NS1
成功
VNR 4
辐条
VN9, VN11, VN43, VN32
NS1
成功
VNR 5
网格
VN12, VN11, VN43, VN32, VN45 +5
NS1
失败
VNR 6
网格
VN10, VN11, VN43, VN32
NS1
成功
VNR 7
网格
VN13, VN11, VN43, VN32
NS1
成功
VNR 8
辐条
VN12, VN11, VN43, VN32
NS1
成功
VNR 9
辐条
VN16, VN21
NS1
成功
VNR 10
辐条
VN10, VN134
NS1
成功
10项
终端
添加标签

图10L

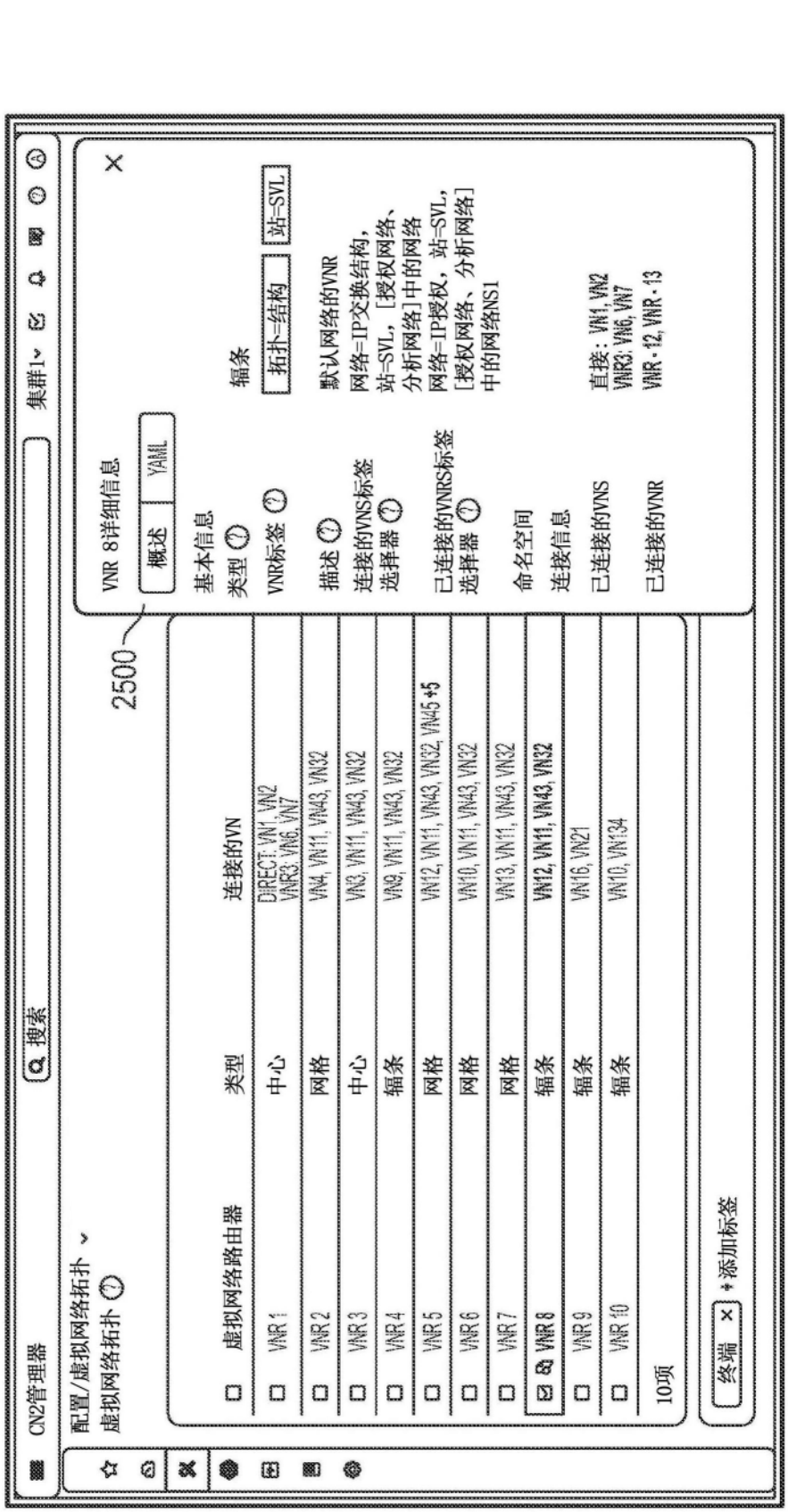

2000M
CN2管理器
搜索
集群1
配置/虚拟网络拓扑
虚拟网络拓扑
2500
虚拟网络路由器
类型
连接的VN
VNR 1
中心
DIRECT: VN1, VN2
VNR3: VN6, VN7
VNR 2
网格
VN4, VN11, VN43, VN32
VNR 3
中心
VN3, VN11, VN43, VN32
VNR 4
辐条
VN9, VN11, VN43, VN32
VNR 5
网格
VN12, VN11, VN43, VN32, VN45 +5
VNR 6
网格
VN10, VN11, VN43, VN32
VNR 7
网格
VN13, VN11, VN43, VN32
VNR 8
辐条
VN12, VN11, VN43, VN32
VNR 9
辐条
VN16, VN21
VNR 10
辐条
VN10, VN134
10项
终端
添加标签
VNR 8详细信息
概述
YAML
基本信息
类型
辐条
VNR标签
拓扑=结构
站=SVL
描述
默认网络的VNR
网络=IP交换结构，
站=SVL，[授权网络、
分析网络]中的网络
连接的VNS标签
选择器
网络=IP授权，站=SVL，
[授权网络、分析网络]
已连接的VNRS标签
选择器
中的网络NS1
命名空间
连接信息
已连接的VNS
直接：VN1, VN2
VNR3: VN6, VN7
已连接的VNR
VNR-12, VNR-13

图10M

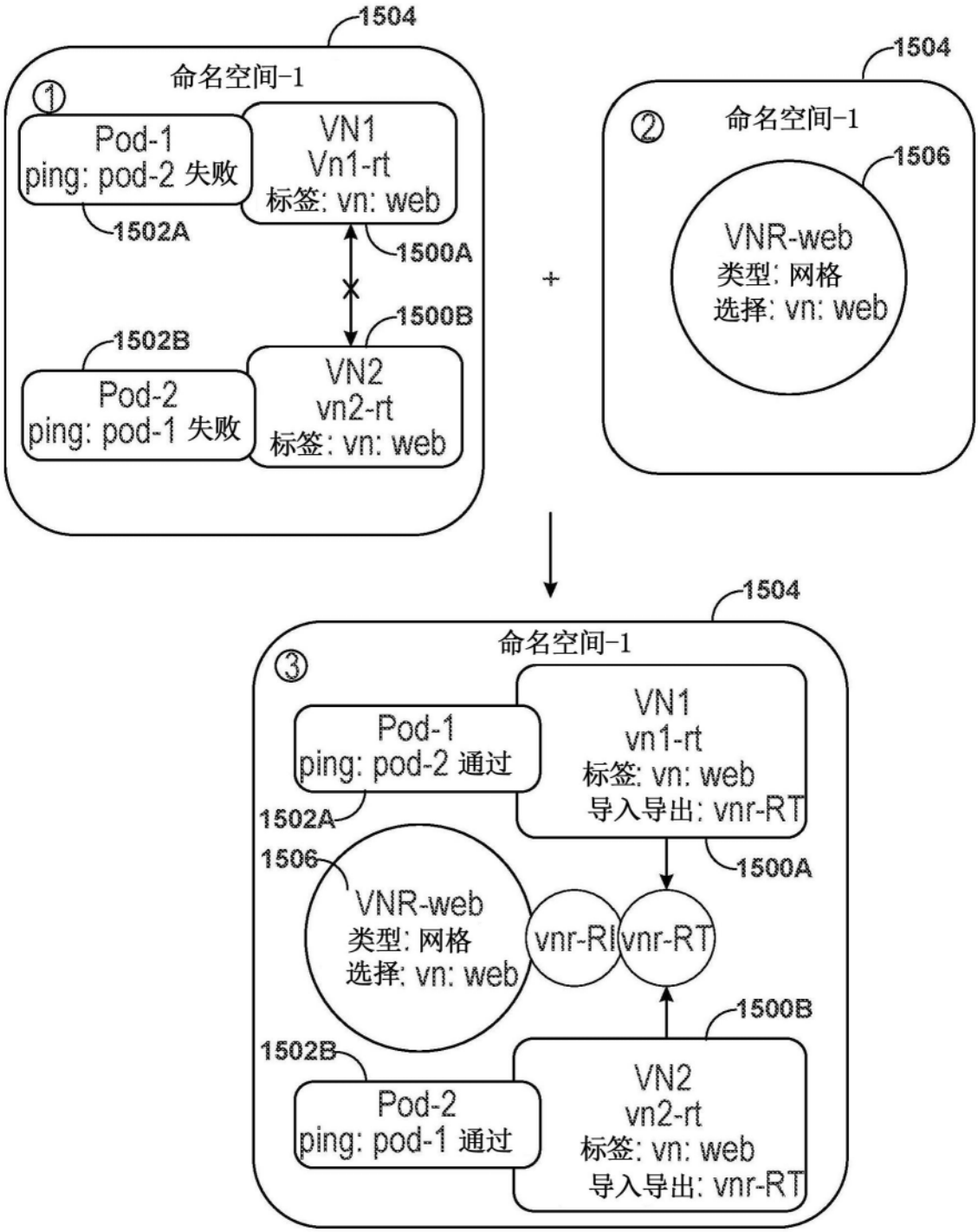

1504
命名空间-1
①
Pod-1
ping: pod-2 失败
VN1
Vn1-rt
标签: vn: web
1502A
1500A
1500B
1502B
Pod-2
ping: pod-1 失败
VN2
vn2-rt
标签: vn: web
+
1504
②
命名空间-1
1506
VNR-web
类型: 网格
选择: vn: web
1504
③
命名空间-1
Pod-1
ping: pod-2 通过
VN1
vn1-rt
标签: vn: web
导入导出: vnr-RT
1502A
1506
1500A
VNR-web
类型: 网格
选择: vn: web
vnr-RI
vnr-RT
1500B
1502B
Pod-2
ping: pod-1 通过
VN2
vn2-rt
标签: vn: web
导入导出: vnr-RT

图11

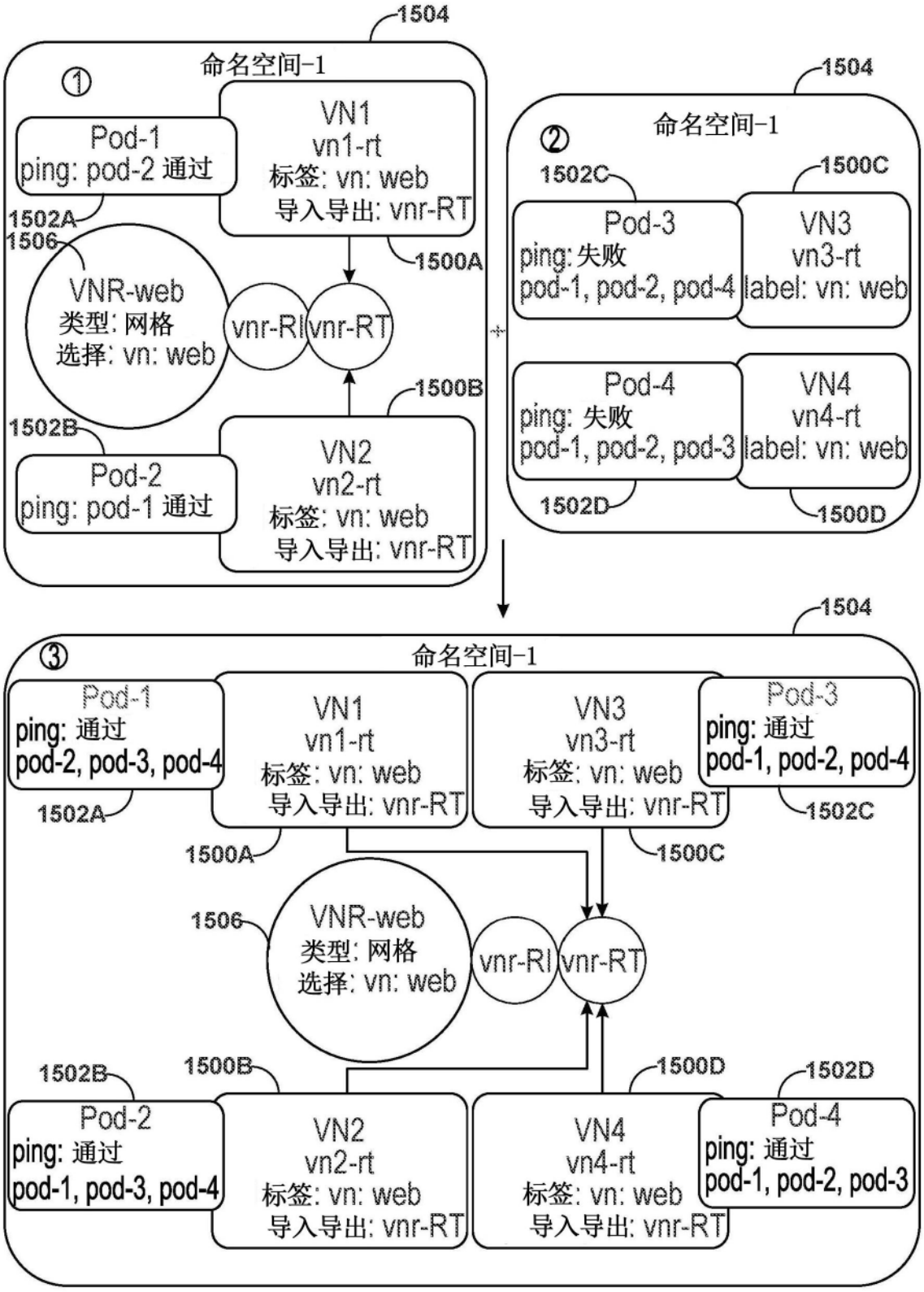

1504
命名空间-1
①
Pod-1
ping: pod-2 通过
VN1
vn1-rt
标签: vn: web
导入导出: vnr-RT
1502A
1506
1500A
VNR-web
类型: 网格
选择: vn: web
vnr-RI
vnr-RT
1500B
1502B
Pod-2
ping: pod-1 通过
VN2
vn2-rt
标签: vn: web
导入导出: vnr-RT
1504
命名空间-1
②
1502C
1500C
Pod-3
ping: 失败
pod-1, pod-2, pod-4
VN3
vn3-rt
label: vn: web
Pod-4
ping: 失败
pod-1, pod-2, pod-3
VN4
vn4-rt
label: vn: web
1502D
1500D
1504
命名空间-1
③
Pod-1
ping: 通过
pod-2, pod-3, pod-4
VN1
vn1-rt
标签: vn: web
导入导出: vnr-RT
VN3
vn3-rt
标签: vn: web
导入导出: vnr-RT
Pod-3
ping: 通过
pod-1, pod-2, pod-4
1502A
1500A
1500C
1502C
1506
VNR-web
类型: 网格
选择: vn: web
vnr-RI
vnr-RT
1502B
1500B
1500D
1502D
Pod-2
ping: 通过
pod-1, pod-3, pod-4
VN2
vn2-rt
标签: vn: web
导入导出: vnr-RT
VN4
vn4-rt
标签: vn: web
导入导出: vnr-RT
Pod-4
ping: 通过
pod-1, pod-2, pod-3

图12

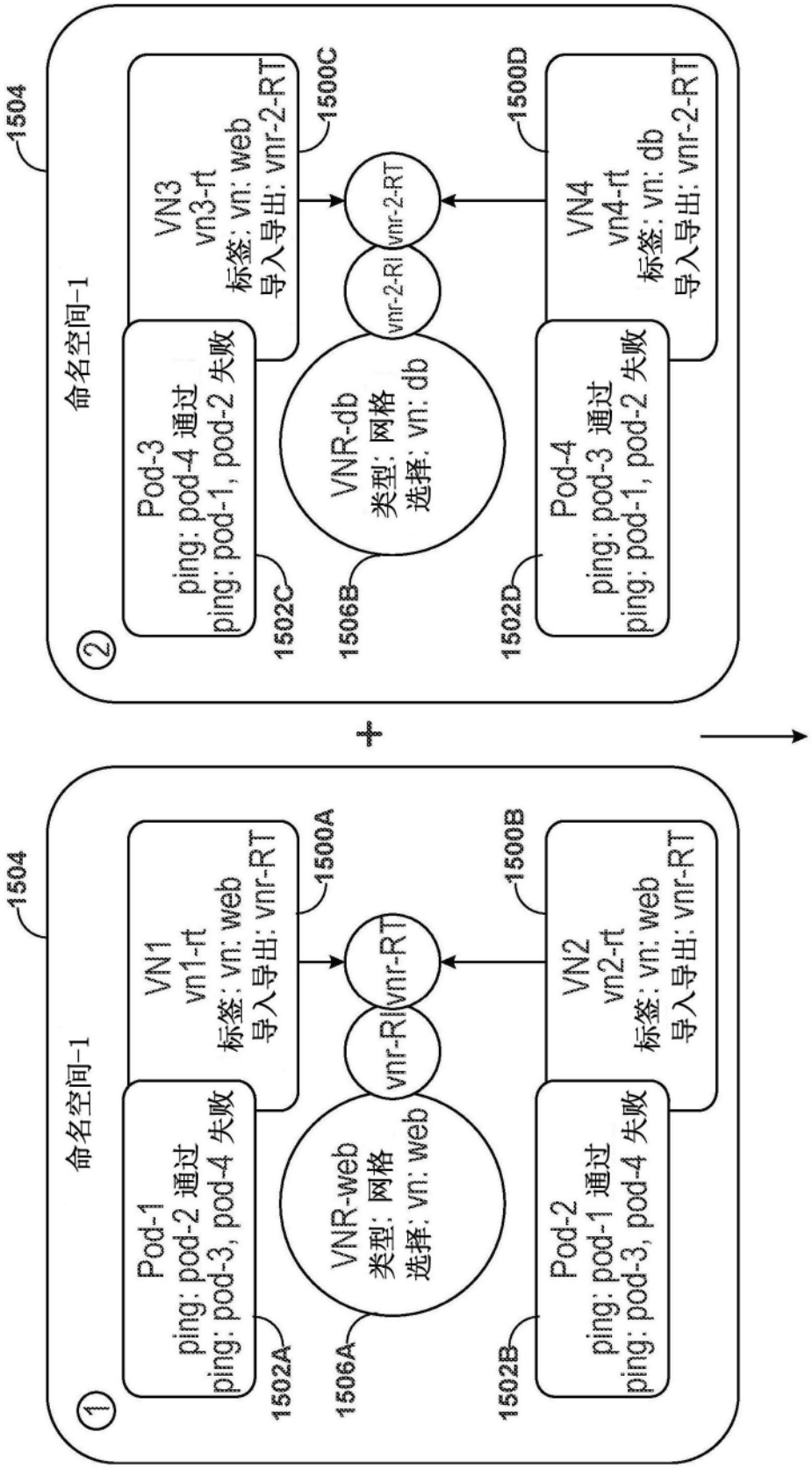

1504
命名空间-1
①
Pod-1
ping: pod-2 通过
ping: pod-3, pod-4 失败
1502A
VN1
vn1-rt
标签: vn: web
导入导出: vnr-RT
1500A
VNR-web
类型: 网格
选择: vn: web
1506A
vnr-RI
vnr-RT
Pod-2
ping: pod-1 通过
ping: pod-3, pod-4 失败
1502B
VN2
vn2-rt
标签: vn: web
导入导出: vnr-RT
1500B
+
1504
命名空间-1
②
Pod-3
ping: pod-4 通过
ping: pod-1, pod-2 失败
1502C
VN3
vn3-rt
标签: vn: web
导入导出: vnr-2-RT
1500C
VNR-db
类型: 网格
选择: vn: db
1506B
vnr-2-RI
vnr-2-RT
Pod-4
ping: pod-3 通过
ping: pod-1, pod-2 失败
1502D
VN4
vn4-rt
标签: vn: db
导入导出: vnr-2-RT
1500D

图13A

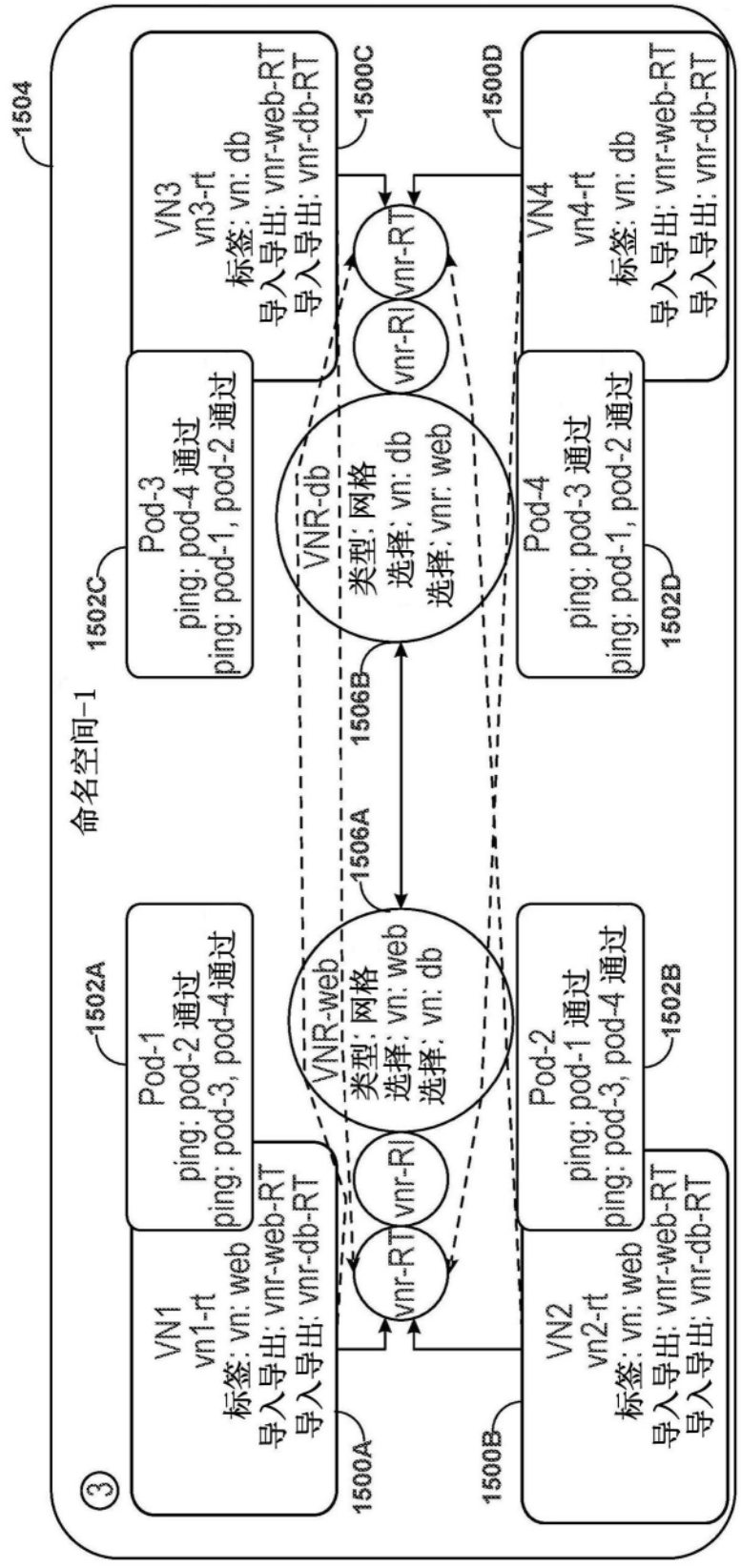

1504
命名空间-1
③
VN1
vn1-rt
标签: vn: web
导入导出: vnr-web-RT
导入导出: vnr-db-RT
1500A
Pod-1
ping: pod-2 通过
ping: pod-3, pod-4通过
1502A
VN2
vn2-rt
标签: vn: web
导入导出: vnr-web-RT
导入导出: vnr-db-RT
1500B
Pod-2
ping: pod-1 通过
ping: pod-3, pod-4 通过
1502B
VN3
vn3-rt
标签: vn: db
导入导出: vnr-web-RT
导入导出: vnr-db-RT
1500C
Pod-3
ping: pod-4 通过
ping: pod-1, pod-2 通过
1502C
VN4
vn4-rt
标签: vn: db
导入导出: vnr-web-RT
导入导出: vnr-db-RT
1500D
Pod-4
ping: pod-3 通过
ping: pod-1, pod-2 通过
1502D
VNR-web
类型: 网格
选择: vn: web
选择: vn: db
1506A
VNR-db
类型: 网格
选择: vn: db
选择: vnr: web
1506B
vnr-RT
vnr-RI
vnr-RT
vnr-RI

图13B

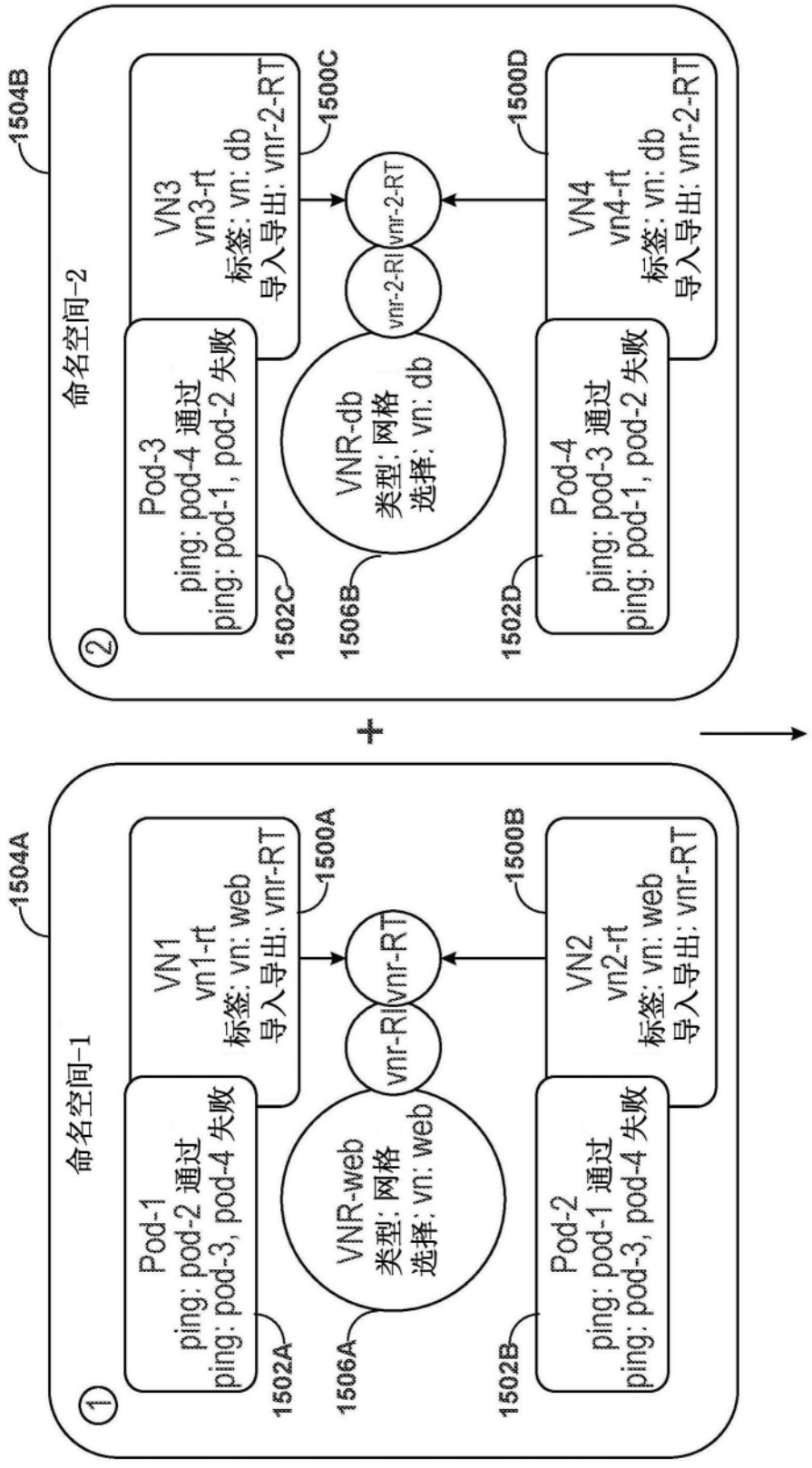

1504A
命名空间-1
①
Pod-1
ping: pod-2 通过
ping: pod-3, pod-4 失败
1502A
VN1
vn1-rt
标签: vn: web
导入导出: vnr-RT
1500A
VNR-web
类型: 网格
选择: vn: web
1506A
vnr-RI
vnr-RT
1500B
VN2
vn2-rt
标签: vn: web
导入导出: vnr-RT
Pod-2
ping: pod-1 通过
ping: pod-3, pod-4 失败
1502B
1504B
命名空间-2
②
Pod-3
ping: pod-4 通过
ping: pod-1, pod-2 失败
1502C
VN3
vn3-rt
标签: vn: db
导入导出: vnr-2-RT
1500C
VNR-db
类型: 网格
选择: vn: db
1506B
vnr-2-RI
vnr-2-RT
1500D
VN4
vn4-rt
标签: vn: db
导入导出: vnr-2-RT
Pod-4
ping: pod-3 通过
ping: pod-1, pod-2 失败
1502D

图14A

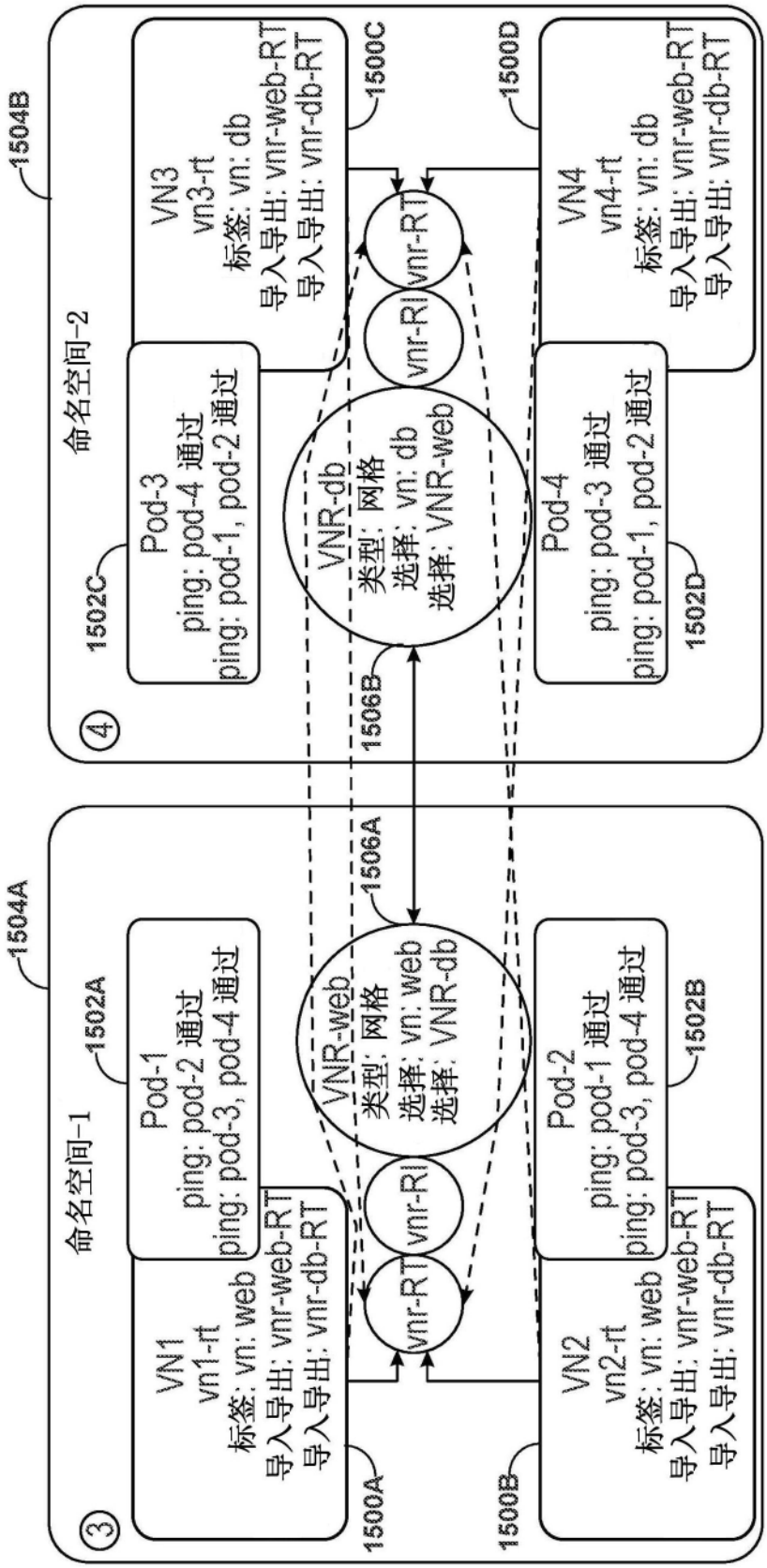

1504B
命名空间-2
④
1502C
Pod-3
ping: pod-4 通过
ping: pod-1, pod-2 通过
1500C
VN3
vn3-rt
标签: vn: db
导入导出: vnr-web-RT
导入导出: vnr-db-RT
1500D
VN4
vn4-rt
标签: vn: db
导入导出: vnr-web-RT
导入导出: vnr-db-RT
1506B
VNR-db
类型: 网格
选择: vn: db
选择: VNR-web
vnr-RI
vnr-RT
Pod-4
ping: pod-3 通过
ping: pod-1, pod-2 通过
1502D
1504A
命名空间-1
③
1502A
Pod-1
ping: pod-2 通过
ping: pod-3, pod-4 通过
1506A
VNR-web
类型: 网格
选择: vn: web
选择: VNR-db
vnr-RI
vnr-RT
Pod-2
ping: pod-1 通过
ping: pod-3, pod-4 通过
1502B
1500A
VN1
vn1-rt
标签: vn: web
导入导出: vnr-web-RT
导入导出: vnr-db-RT
1500B
VN2
vn2-rt
标签: vn: web
导入导出: vnr-web-RT
导入导出: vnr-db-RT

图14B

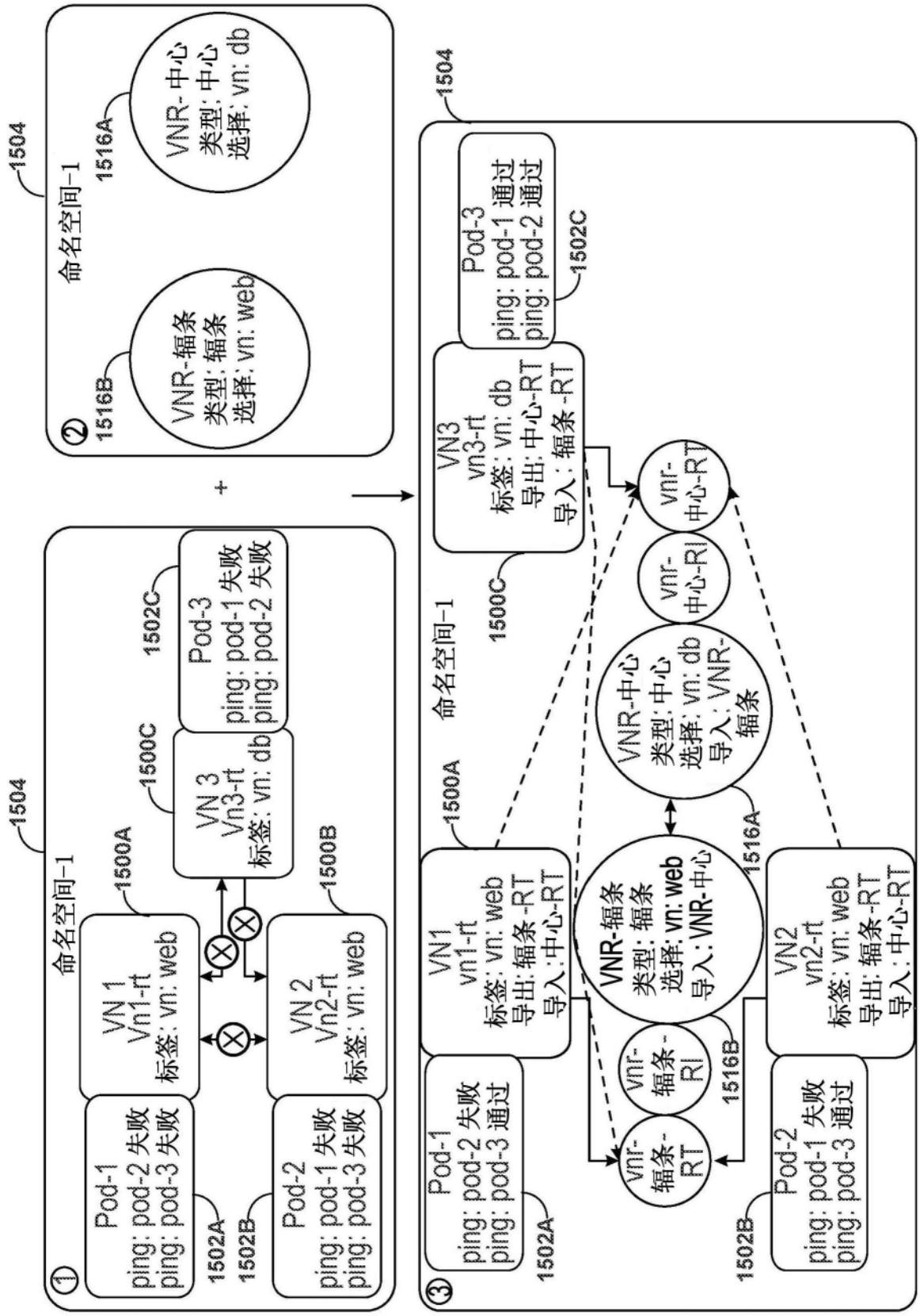

1504
命名空间-1
①
Pod-1
ping: pod-2 失败
ping: pod-3 失败
1502A
VN 1
Vn1-rt
标签: vn: web
1500A
Pod-2
ping: pod-1 失败
ping: pod-3 失败
1502B
VN 2
Vn2-rt
标签: vn: web
1500B
VN 3
Vn3-rt
标签: vn: db
1500C
Pod-3
ping: pod-1 失败
ping: pod-2 失败
1502C
+
1504
命名空间-1
②
1516B
VNR-辐条
类型: 辐条
选择: vn: web
1516A
VNR- 中心
类型: 中心
选择: vn: db
1504
命名空间-1
③
Pod-1
ping: pod-2 失败
ping: pod-3 通过
1502A
VN1
vn1-rt
标签: vn: web
导出: 辐条 -RT
导入 : 中心-RT
1500A
VNR-辐条
类型: 辐条
选择: vn: web
导入 : VNR-中心
1516A
VN3
vn3-rt
标签: vn: db
导出: 中心-RT
导入 : 辐条 -RT
1500C
Pod-3
ping: pod-1 通过
ping: pod-2 通过
1502C
VNR-中心
类型: 中心
选择: vn: db
导入 : VNR-辐条
vnr-辐条-RT
vnr-辐条-RI
1516B
vnr-中心-RI
vnr-中心-RT
Pod-2
ping: pod-1 失败
ping: pod-3 通过
1502B
VN2
vn2-rt
标签: vn: web
导出: 辐条-RT
导入 : 中心-RT

图15

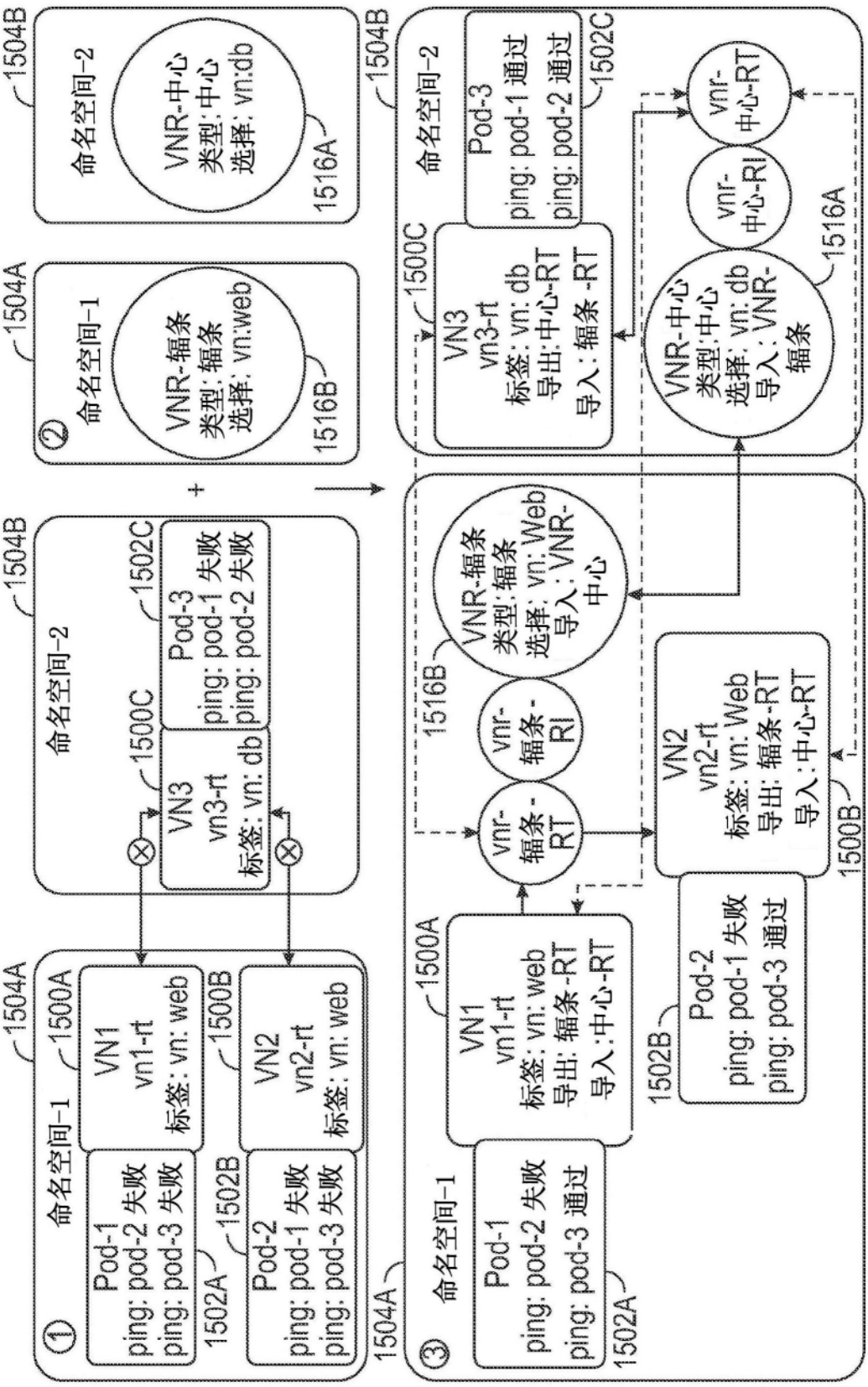

①
命名空间-1
Pod-1
ping: pod-2 失败
ping: pod-3 失败
Pod-2
ping: pod-1 失败
ping: pod-3 失败
VN1
vn1-rt
标签: vn: web
VN2
vn2-rt
标签: vn: web
1502A
1502B
1500A
1500B
1504A
命名空间-2
VN3
vn3-rt
标签: vn: db
Pod-3
ping: pod-1 失败
ping: pod-2 失败
1500C
1502C
1504B
②
命名空间-1
VNR-辐条
类型: 辐条
选择: vn:web
1516B
命名空间-2
VNR-中心
类型: 中心
选择: vn:db
1516A
③
命名空间-1
Pod-1
ping: pod-2 失败
ping: pod-3 通过
VN1
vn1-rt
标签: vn: web
导出: 辐条-RT
导入: 中心-RT
vnr-辐条-RT
vnr-辐条-RI
VNR-辐条
类型: 辐条
选择: vn: Web
导入: VNR-中心
Pod-2
ping: pod-1 失败
ping: pod-3 通过
VN2
vn2-rt
标签: vn: Web
导出: 辐条-RT
导入: 中心-RT
命名空间-2
Pod-3
ping: pod-1 通过
ping: pod-2 通过
VN3
vn3-rt
标签: vn: db
导出: 中心-RT
导入: 辐条-RT
VNR-中心
类型: 中心
选择: vn: db
导入: VNR-辐条
vnr-中心-RI
vnr-中心-RT

图16

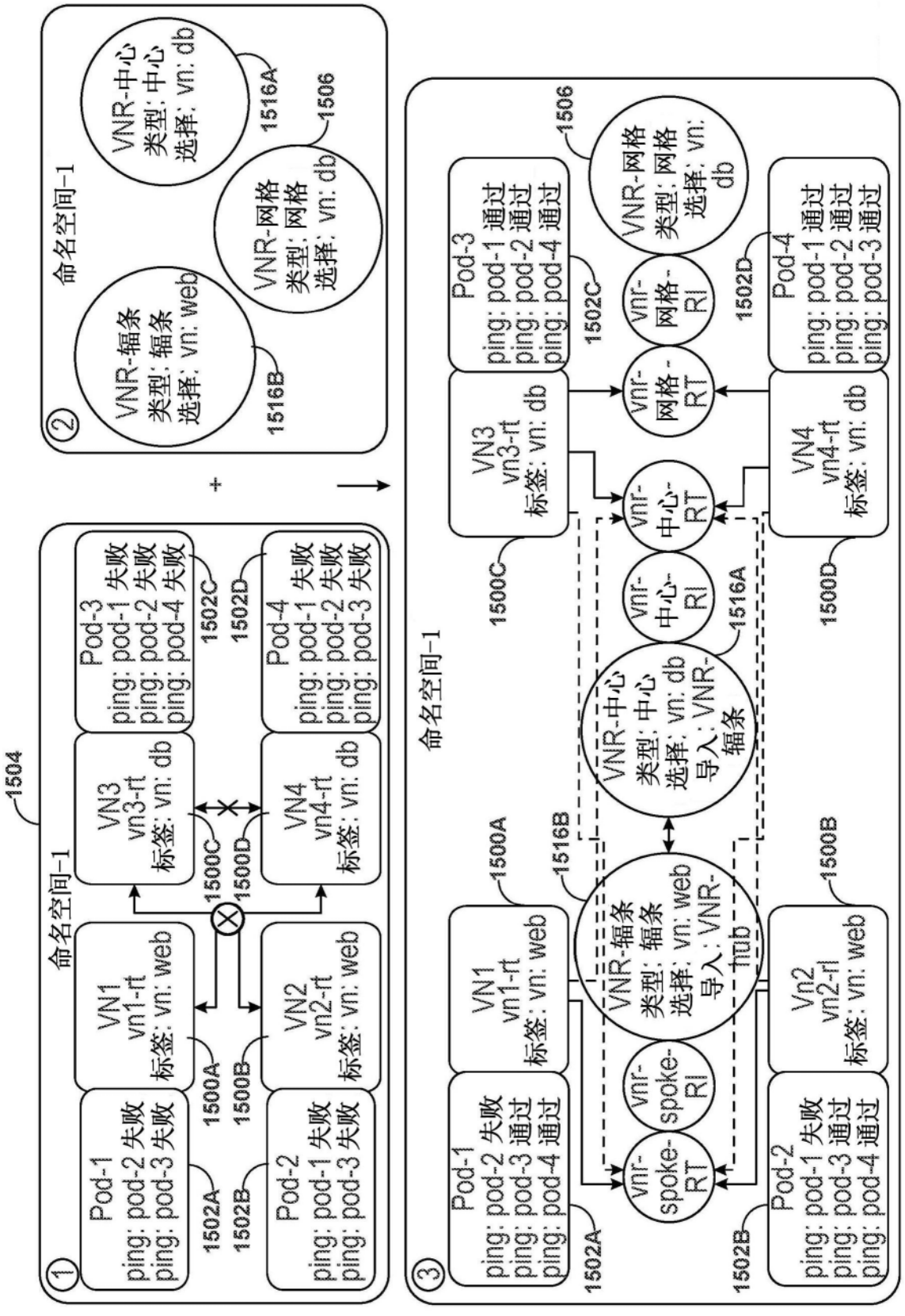

1504
命名空间-1
VN1
vn1-rt
标签: vn: web
VN2
vn2-rt
标签: vn: web
VN3
vn3-rt
标签: vn: db
VN4
vn4-rt
标签: vn: db
Pod-1
ping: pod-2 失败
ping: pod-3 失败
Pod-2
ping: pod-1 失败
ping: pod-3 失败
Pod-3
ping: pod-1 失败
ping: pod-2 失败
ping: pod-4 失败
Pod-4
ping: pod-1 失败
ping: pod-2 失败
ping: pod-3 失败
1500A
1500B
1500C
1500D
1502A
1502B
1502C
1502D
①
②
③
VNR-辐条
类型: 辐条
选择: vn: web
VNR-网格
类型: 网格
选择: vn: db
VNR-中心
类型: 中心
选择: vn: db
1516A
1516B
1506
命名空间-1
VNR-辐条
类型: 辐条
选择: vn: web
导入: VNR-hub
VNR-中心
类型: 中心
选择: vn: db
导入: VNR-辐条
vnr-spoke-RT
vnr-spoke-RI
vnr-中心-RT
vnr-中心-RI
vnr-网格-RT
vnr-网格-RI
Pod-1
ping: pod-2 失败
ping: pod-3 通过
ping: pod-4 通过
Pod-2
ping: pod-1 失败
ping: pod-3 通过
ping: pod-4 通过
Pod-3
ping: pod-1 通过
ping: pod-2 通过
ping: pod-4 通过
Pod-4
ping: pod-1 通过
ping: pod-2 通过
ping: pod-3 通过
VNR-网格
类型: 网格
选择: vn: db

图17

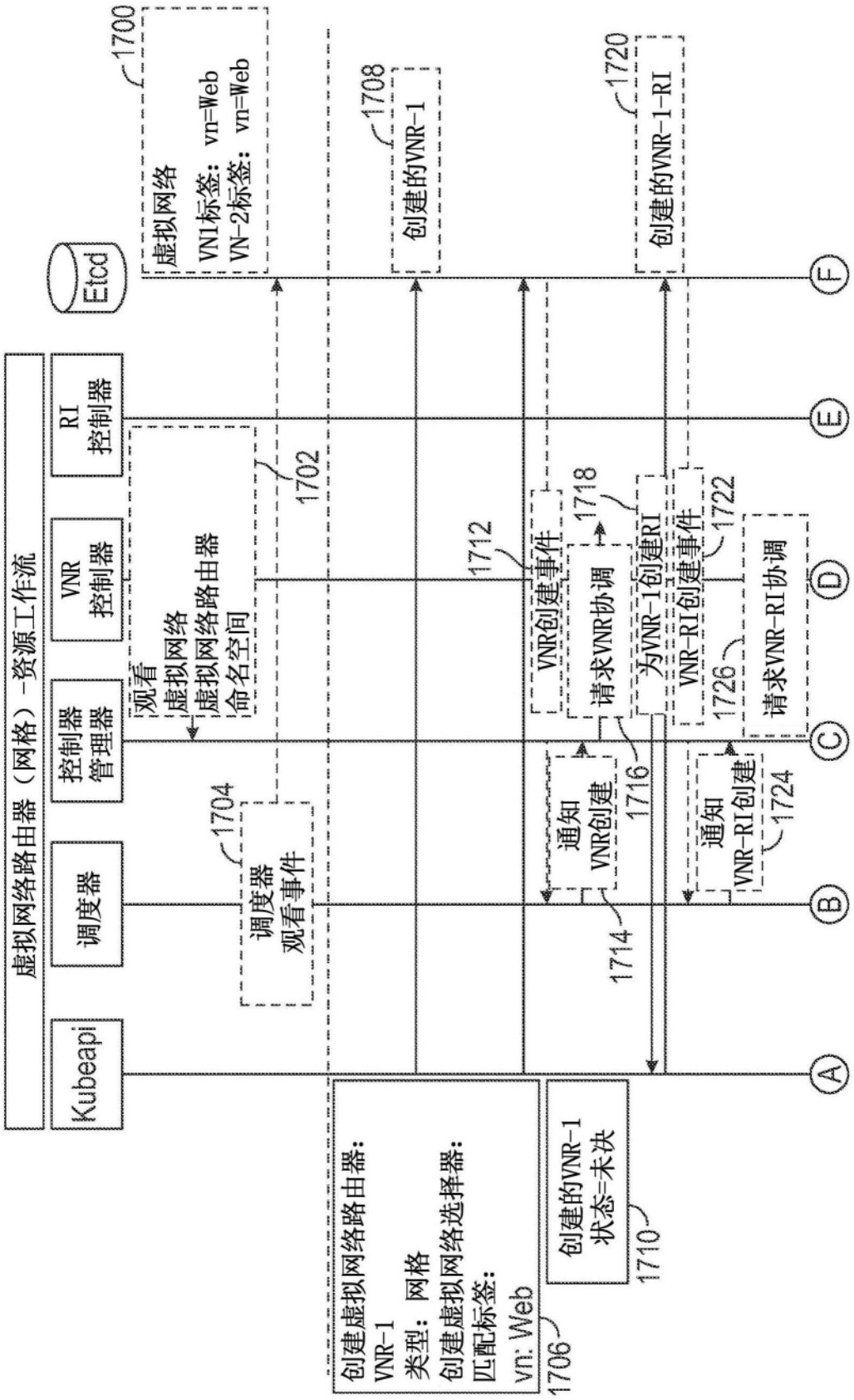

虚拟网络路由器（网格）-资源工作流
Kubeapi
调度器
控制器管理器
VNR控制器
RI控制器
Etcd
1700
虚拟网络
VN1标签：vn=Web
VN-2标签：vn=Web
1708
创建的VNR-1
1720
创建的VNR-1-RI
1702
观看
虚拟网络
虚拟网络路由器
命名空间
1704
调度器
观看事件
1712
VNR创建事件
1718
请求VNR协调
为VNR-1创建RI
1722
VNR-RI创建事件
1726
请求VNR-RI协调
1716
通知
VNR创建
1714
1724
通知
VNR-RI创建
A
B
C
D
E
F
创建虚拟网络路由器：
VNR-1
类型：网格
创建虚拟网络选择器：
匹配标签：
vn: Web
1706
创建的VNR-1
状态=未决
1710

图18A

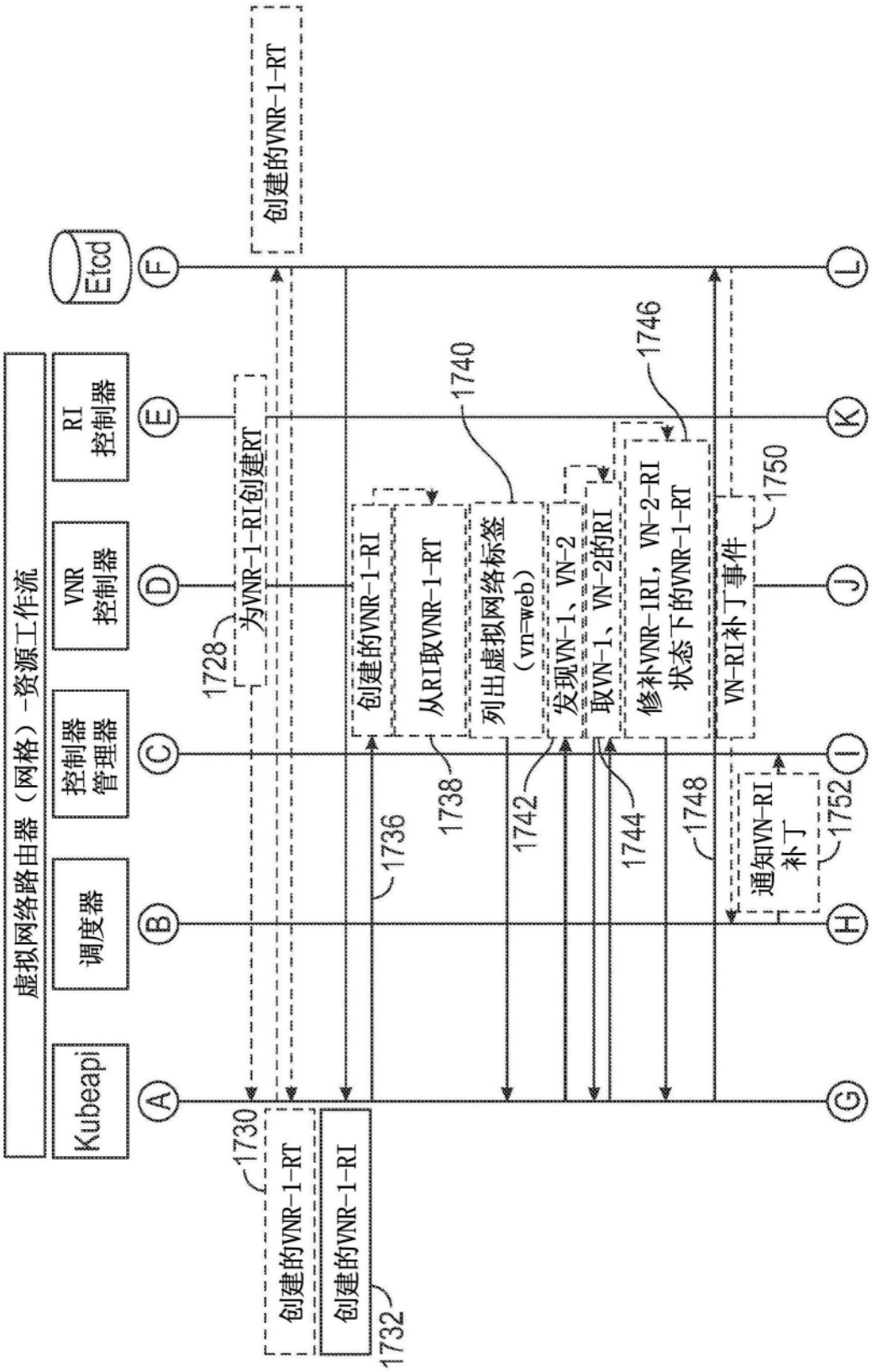

虚拟网络路由器（网格）-资源工作流
Kubeapi
调度器
控制器管理器
VNR控制器
RI控制器
Etcd
A
B
C
D
E
F
G
H
I
J
K
L
创建的VNR-1-RT
为VNR-1-RI创建RT
1728
1730
创建的VNR-1-RT
创建的VNR-1-RI
1732
1736
创建的VNR-1-RI
从RI取VNR-1-RT
1738
1740
列出虚拟网络标签
(vn=web)
1742
发现VN-1、VN-2
取VN-1、VN-2的RI
1744
修补VNR-1RI，VN-2-RI
状态下的VNR-1-RT
1746
1748
VN-RI补丁事件
1750
通知VN-RI
补丁
1752

图18B

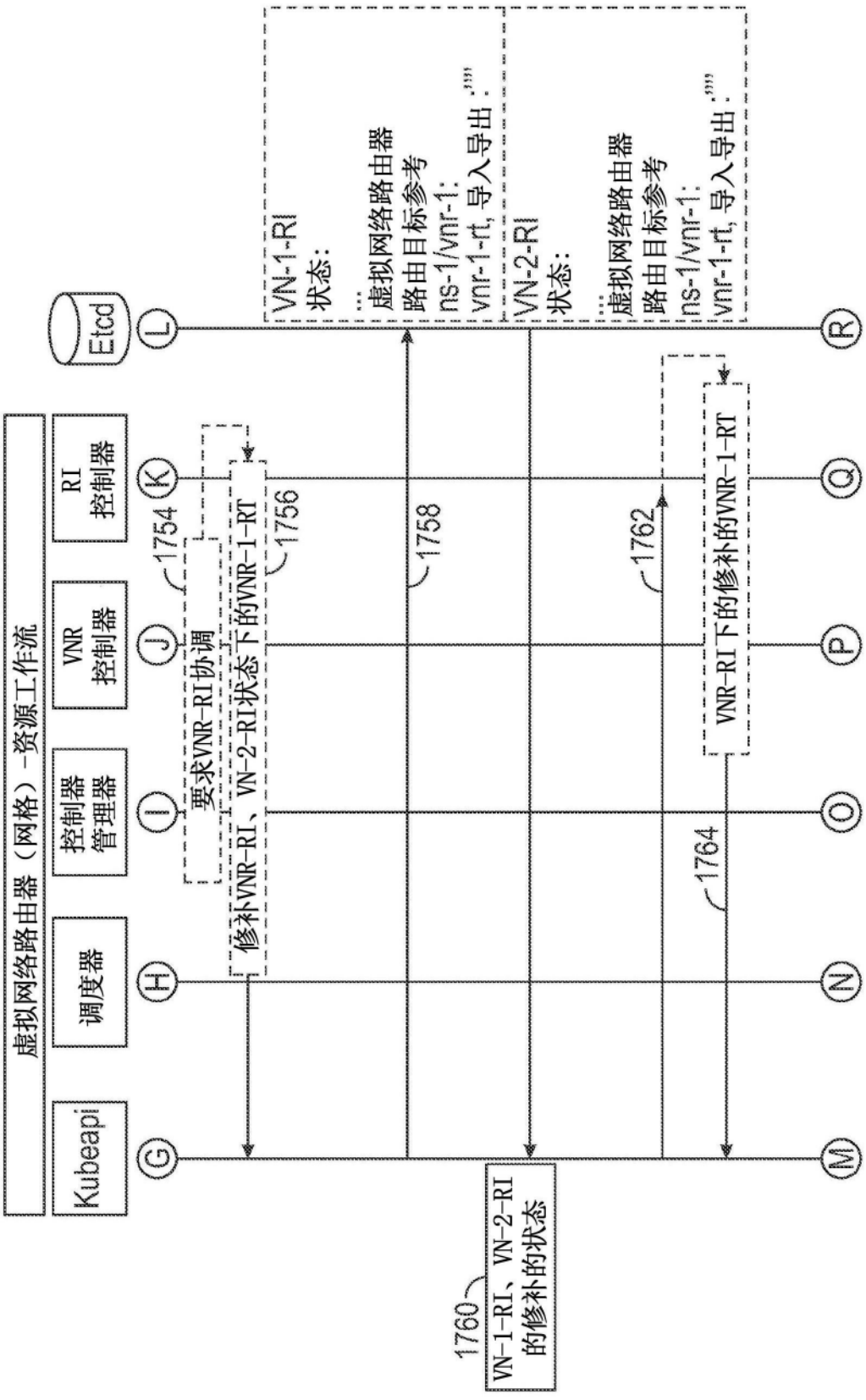

虚拟网络路由器（网格）-资源工作流
Kubeapi
调度器
控制器管理器
VNR控制器
RI控制器
Etcd
G
H
I
J
K
L
M
N
O
P
Q
R
VN-1-RI
状态：
...
虚拟网络路由器
路由目标参考
ns-1/vnr-1:
vnr-1-rt, 导入导出：""
VN-2-RI
状态：
...
虚拟网络路由器
路由目标参考
ns-1/vnr-1:
vnr-1-rt, 导入导出：""
1754
要求VNR-RI协调
1756
修补VNR-RI、VN-2-RI状态下的VNR-1-RT
1758
1762
VNR-RI下的修补的VNR-1-RT
1764
1760
VN-1-RI、VN-2-RI
的修补的状态

图18C

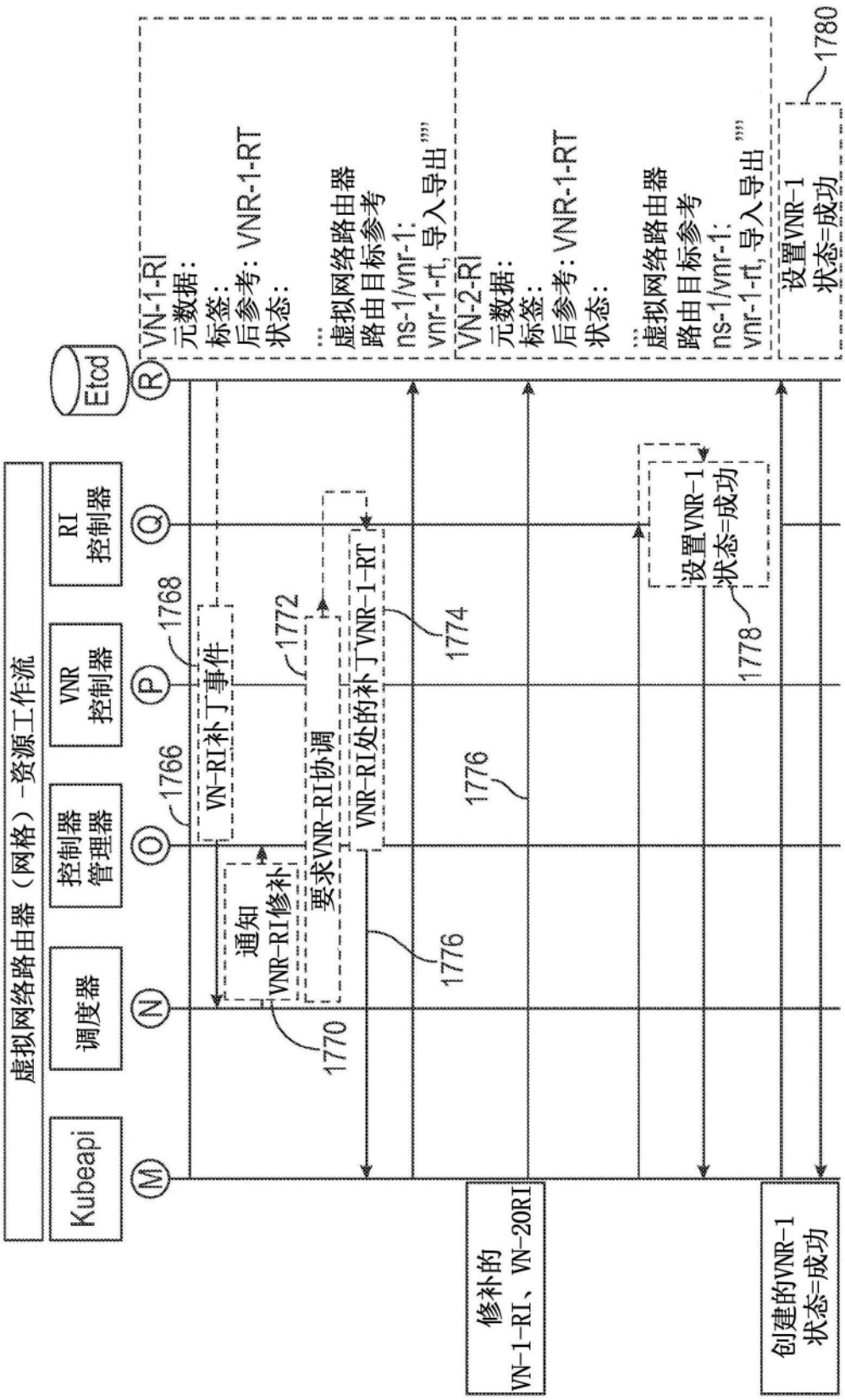

虚拟网络路由器（网格）-资源工作流
Kubeapi
调度器
控制器管理器
VNR控制器
RI控制器
Etcd
M
N
O
P
Q
R
VN-1-RI
元数据：
标签：
后参考：VNR-1-RT
状态：
...
虚拟网络路由器
路由目标参考
ns-1/vnr-1:
vnr-1-rt, 导入导出""
VN-2-RI
元数据：
标签：
后参考：VNR-1-RT
状态：
...
虚拟网络路由器
路由目标参考
ns-1/vnr-1:
vnr-1-rt, 导入导出""
设置VNR-1
状态=成功
1780
1766
VN-RI补丁事件
1768
通知
VNR-RI修补
1770
1772
要求VNR-RI协调
VNR-RI处的补丁VNR-1-RT
1774
1776
1776
设置VNR-1
状态=成功
1778
修补的
VN-1-RI、VN-2ORI
创建的VNR-1
状态=成功

图18D

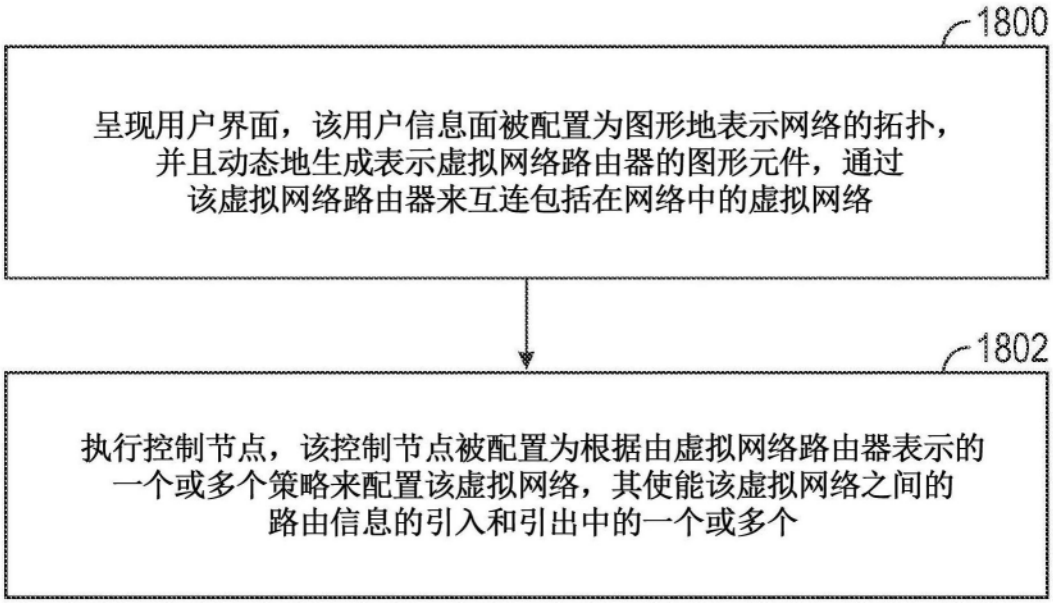

1800
呈现用户界面，该用户信息面被配置为图形地表示网络的拓扑，并且动态地生成表示虚拟网络路由器的图形元件，通过该虚拟网络路由器来互连包括在网络中的虚拟网络
1802
执行控制节点，该控制节点被配置为根据由虚拟网络路由器表示的一个或多个策略来配置该虚拟网络，其使能该虚拟网络之间的路由信息的引入和引出中的一个或多个

图19