



(10) 授权公告号 CN 111949367 B

(45) 授权公告日 2024. 10. 18

(21) 申请号 202010648622.0

(22) 申请日 2015.02.26

(65) 同一申请的已公布的文献号
申请公布号 CN 111949367 A

(43) 申请公布日 2020.11.17

(30) 优先权数据
14/193,034 2014.02.28 US(62) 分案原申请数据
201580011235.3 2015.02.26(73) 专利权人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼

(72) 发明人 张书锦

(74) 专利代理机构 广州三环专利商标代理有限公司 44202

专利代理师 李稷芳

(51) Int.Cl.
G06F 9/455 (2006.01)
G06F 8/30 (2018.01)(56) 对比文件
MOHAMMAD BNIKAZEMI ET AL.Meridian: An SDN Platform for Cloud Network Services.IEEE COMMUNICATIONS MAGAZINE, IEEE SERVICE CENTER.2013,第120-127页.

审查员 吴瑶

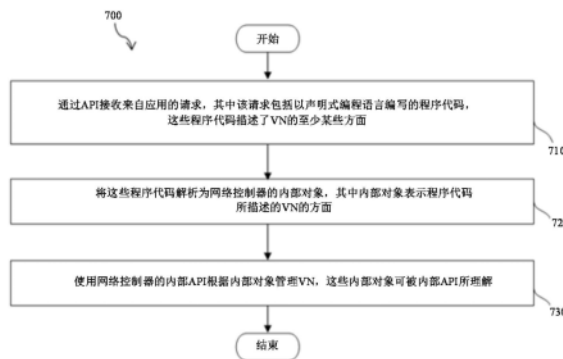
权利要求书2页 说明书9页 附图8页

(54) 发明名称

虚拟网络创建和操作的声明式方法

(57) 摘要

一种网络控制器可以通过应用编程接口(application programming interface,API)接收来自应用的请求,其中所述请求包括以声明式编程语言编写的程序代码,所述程序代码描述了虚拟网络(virtual network,VN)的至少某些方面。所述网络控制器还可以将所述程序代码解析为所述网络控制器的内部对象,其中所述内部对象表示所述程序代码所描述的所述VN的所述方面。所述网络控制器随后可以根据从所述程序代码转换而来的所述内部对象管理所述VN。



1. 一种由网络控制器实施的用于虚拟网络VN管理的方法,其特征在于,所述方法包括:
通过应用编程接口API接收请求,所述请求包括以声明式编程语言编写的程序代码,所述声明式编程语言编写的程序代码用于执行所述VN的相关任务,所述相关任务包括设置所述VN,所述设置所述VN包括查询所述VN;
所述VN包括网络节点、所述网络节点上的端口、以及所述端口之间的链路;
将所述VN的相关任务解析为所述网络控制器的内部对象;
根据所述内部对象执行所述VN的相关任务。
2. 根据权利要求1所述的方法,其特征在于,所述相关任务还包括执行所述VN的网络策略或处理所述VN的事件中的一种或多种任务。
3. 根据权利要求1所述的方法,其特征在于,所述设置所述VN还包括创建所述VN、删除所述VN、更新所述VN中的一种或多种方式。
4. 根据权利要求2或3所述的方法,其特征在于,所述设置所述VN包括设置所述VN的所述网络节点、所述网络节点上的端口、以及所述端口之间的链路中的一个或多个。
5. 根据权利要求1-3任一所述的方法,其特征在于,所述请求包括元素,所述元素用于标识所述元素为所述VN的合法名称;
在将所述VN的相关任务解析为所述网络控制器的内部对象之前,所述方法还包括:
基于所述元素为合法名称,确定能够将所述相关任务解析为所述网络控制器的内部对象。
6. 根据权利要求1-3任一所述的方法,其特征在于,所述请求包括子代元素,所述子代元素包括用于描述所述VN的节点、端口、链路、带宽、策略和事件中的一个或多个的合法名称。
7. 根据权利要求1所述的方法,其特征在于,所述声明式编程语言为可扩展标记语言XML、Alpha、Atom、应用类型系统ATS、Curl、分布式应用规范语言、以数据为中心的编程语言、Erlang、可扩展应用标记语言XAML、框架逻辑F-logic、FXML、通用声明式语言GenDL、Glowe、GOAL代理编程语言、Gofer语言、知识机器KM编程语言、Lithe、Lucid、Lustre、Miranda、Pan、Qi、XML、QUILL、SequenceL、结构化查询语言SQL、同步多媒体集成语言SMIL,以及网络本体语言OWL中的一种或多种组合。
8. 根据权利要求1-3、7中任一所述的方法,其特征在于,所述网络控制器包括物理控制器和VN控制器,所述VN控制器获得与所述物理控制器对应的网络拓扑,所述网络拓扑用于设置所述VN。
9. 根据权利要求1-3、7中任一所述的方法,其特征在于,所述将所述VN的相关任务解析为所述网络控制器的内部对象包括:
使用Python将所述VN的相关任务解析为所述网络控制器的内部对象。
10. 根据权利要求1-3、7中任一所述的方法,其特征在于,所述VN为虚拟专用网VPN、软件定义网络SDN、服务为中心的网络SCN、信息为中心的网络ICN、内容为中心的网络CCN、虚拟运营商网络、虚拟企业网或应用特定网络。
11. 一种通信装置,其特征在于,包括:
接收器,通过应用编程接口API接收请求,所述请求包括以声明式编程语言编写的程序代码,所述声明式编程语言编写的程序代码用于执行VN的相关任务,所述相关任务包括设

置所述VN,所述设置所述VN包括查询所述VN;

所述VN包括网络节点、所述网络节点上的端口、以及所述端口之间的链路;

耦合到所述接收器的处理器,用于将所述VN的相关任务解析为所述装置的内部对象,根据所述内部对象对所述网络节点、所述网络节点上的端口和所述端口之间的链路中的一个或多个执行所述相关任务。

12.根据权利要求11所述的装置,其特征在于,所述相关任务还包括执行所述VN的网络策略或处理所述VN的事件的一种或多种任务。

13.根据权利要求11所述的装置,其特征在于,所述设置所述VN还包括创建所述VN、删除所述VN、更新所述VN和查询所述VN中的一种或多种方式。

14.根据权利要求12或13所述的装置,其特征在于,所述设置所述VN包括设置所述VN的所述网络节点、所述网络节点上的端口、以及所述端口之间的链路中的一个或多个。

15.根据权利要求11-13任一所述的装置,其特征在于,所述请求包括元素,所述元素用于标识所述元素为所述VN的合法名称;所述处理器,用于基于所述元素为合法名称,确定能够将所述相关任务解析为所述装置的内部对象。

16.根据权利要求11-13任一所述的装置,其特征在于,所述请求包括子代元素,所述子代元素包括用于描述所述VN的网络节点、端口、链路、带宽、策略和事件中的一个或多个的合法名称。

17.根据权利要求11所述的装置,其特征在于,所述声明式编程语言为可扩展标记语言XML、Alpha、Atom、应用类型系统ATS、Curl、分布式应用规范语言、以数据为中心的编程语言、Erlang、可扩展应用标记语言XAML、框架逻辑F-logic、FXML、通用声明式语言GenDL、Glowe、GOAL代理编程语言、Gofer语言、知识机器KM编程语言、Lithe、Lucid、Lustre、Miranda、Pan、Qi、XML、QUILL、SequenceL、结构化查询语言SQL、同步多媒体集成语言SMIL,以及网络本体语言OWL中的一种或多种组合。

18.根据权利要求11-13、17中任一所述的装置,其特征在于,所述处理器包括VN控制器VNC表示模块,所述VNC表示模块使用Python将所述VN的相关任务解析为所述装置的内部对象。

19.根据权利要求11-13、17中任一所述的装置,其特征在于,所述VN为虚拟专用网VPN、软件定义网络SDN、服务为中心的网络SCN、信息为中心的网络ICN、内容为中心的网络CCN、虚拟运营商网络、虚拟企业网或应用特定网络。

20.一种计算机可读存储介质,其特征在于,所述计算机可读介质中存储有指令或程序代码,当所述指令或所述程序代码在计算机上运行时,使得所述计算机执行权利要求1-10任一所述的方法。

虚拟网络创建和操作的声明式方法

[0001] 本申请是名称为“虚拟网络创建和操作的声明式方法”，专利申请号为201580011235.3,申请日为2015年2月26日的分案申请。

技术领域

[0002] 本发明大体涉及虚拟网络(virtual network, VN), 且在具体实施例中, 涉及虚拟网络创建和操作的技术。

背景技术

[0003] 虚拟网络(virtual network, VN)是一种包含虚拟网络节点和/或链路的计算机网络。由于云计算中对快速引入服务和节省运营成本的需求等各种原因, 虚拟网络(virtual network, VN)的概念和采用已变得十分普遍。VN可以利用或不利用对应的物理网络拓扑形成。为了创建和操作VN, 网络应用可以调用控制VN的网络控制器所提供的编程接口(application programming interface, API)。例如, 可以每次通过API将一个来自应用的请求传输到控制器, 并且VN可以根据该请求构建或更新以适应应用的需求。

[0004] 如今, 应用开发人员可以使用过程化方法来操作VN。在该方法中, 应用可以使用控制器API来构建一个表示操作VN的应用逻辑的逐步过程。构建过程化步骤可能要求应用开发人员非常熟悉VN(例如, 如何构建VN)。随着VN的规模扩大且复杂性提高, 对于应用开发人员而言, VN可能变得越来越难管理。

发明内容

[0005] 所公开的示例实施例可以有助于通过使用以声明式编程语言而非过程化编程语言编写的程序代码提升虚拟网络(virtual network, VN)管理以执行各种VN相关任务。在一示例实施例中, 网络控制器可以通过编程接口(application programming interface, API)接收请求开始管理VN, 其中所述请求包括以声明式编程语言编写的程序代码, 所述程序代码描述了VN的至少某些方面。所述网络控制器还可以将所述程序代码解析为所述网络控制器的内部对象, 其中所述内部对象表示所述程序代码描述的所述VN的所述方面。

[0006] 结合附图和权利要求书, 可从以下的详细描述中更清楚地理解这些和其它特征。

附图说明

[0007] 为了更透彻地理解本发明, 现参阅结合附图和具体实施方式而描述的以下简要说明, 其中的相同参考标号表示相同部分。

[0008] 图1是图示网络模型的示例实施例的示意图。

[0009] 图2是图示虚拟网络(virtual network, VN)方案架构的示例实施例的示意图。

[0010] 图3A是图示VN通信架构的示例实施例的示意图。

[0011] 图3B示出了以过程化方法编写的一条示例性伪代码。

- [0012] 图4是图示应用和网络控制器之间的通信架构的示例实施例的示意图。
- [0013] 图5A和5B示出了描述VN的至少某些方面的示例性可扩展标记语言(extensible markup language, XML)纲要。
- [0014] 图6A示出了描述示例性VN的一条示例性程序代码。
- [0015] 图6B示出了使用图6A所示的程序代码创建的VN。
- [0016] 图7是VN管理方法的示例实施例的流程图。
- [0017] 图8是网络设备的示例实施例的示意图。

具体实施方式

[0018] 首先应理解, 尽管下文提供一项或多项示例实施例的说明性实施方案, 但所公开的系统 and/或方法可使用任何数目的技术来实施, 无论该技术是当前已知还是现有的。本发明决不应限于下文所说明的说明性实施方案、附图和技术, 包括本文所说明并描述的示例性设计和实施方案, 而是可在所附权利要求书的范围以及其等效物的完整范围内修改。

[0019] 为了管理虚拟网络(virtual network, VN), 网络控制器可以提供应用编程接口(application programming interface, API) 与应用开发人员操作的应用进行交互。将API格式进行标准化的过程中存在一些成就。例如, 亚马逊提出的弹性计算云(Elastic Compute Cloud, EC2) 可被视为API的事实标准, OpenStack Quantum API可被视为EC2 API的变体。新提案也正在尝试扩展API以包含新特征或应用场景。

[0020] 一些现有的API方法拥有若干共同特征。首先, API可以是描述性的。一个API可以表示应用指示网络控制器执行的动作或任务, 应用开发人员可以提供任务的输入参数并在API调用完成之后获得某个结果。其次, API可以是过程化的。应用可以使用API来构建表示应用逻辑的过程, 因此应用开发人员可能需要非常熟悉虚拟网络如何通过过程化步骤构建。随着VN的规模扩大且复杂性提高, 应用开发人员可能需要编写冗长且复杂的程序代码来创建和操作VN。此外, 维护VN和避免潜在缺陷或错误的成本可能变得很高。

[0021] 本发明教示例实施例以通过在创建和操作VN时使用声明式方法而非过程化方法来解决当前API的缺点。具体而言, 包含程序代码的请求或命令可以通过声明式编程语言而非过程化编程语言生成。声明式语言, 本文中有时也称为虚拟网络定义语言(virtual network definition language, VNDL), 可以以各种格式或纲要, 例如可扩展标记语言(extensible markup language, XML) 格式编写。应用开发人员可以使用VNDL从诸如节点、端口、链路、带宽、策略和事件处理等已定义名称方面来描述VN。与过程化编程语言不同, VNDL可以在无需明确指定每个详细步骤来执行任务的情况下指定任务逻辑或目的。

[0022] 以VNDL编写的程序代码可以由应用生成, 随后可以通过网络控制器支持的API发送给网络控制器。网络控制器可以将VNDL代码解析为内部对象并使用控制器的内部API相应地执行内部对象。网络控制器可以执行各种任务来操作VN, 包括创建VN(例如, 定义节点、端口和链路)、更新VN、执行VN的网络策略, 并处理VN中发生的事件。

[0023] 图1是图示网络模型100的示例实施例的示意图, 网络模型100包括在物理网络层上配置的一个或多个VN。如图1所示, 物理网络层包括网络110和网络120, 这两个网络可被视为不同网络或相同网络的不同域。网络110和120包括物理网络节点, 其可通过物理链路相互互连。网络110和120中的物理网络节点可以与归属网中或外的其它网络节点互连。

[0024] 为了扩展网络功能,可以使用物理网络层中的某些网络节点形成一个或多个VN。例如,第一VN 130可以使用网络节点111、113、115,以及121至125进行配置;第二VN 140可以使用网络节点112、114、116、121至122,以及124至125进行配置。实际上,部分物理节点功能可以用于形成虚拟网络节点,这些节点反过来可以组成VN。因此,一个物理节点可以用于构建一个或多个虚拟节点。如图1所示,形成共享相同物理节点116的两个虚拟节点141和142,这两个虚拟节点均属于VN 140。虚拟链路可以使虚拟节点相互互连。然而,虚拟节点之间也可以没有任何连接。应理解,本文所论述的物理/虚拟网络节点和链路可以通过任意合适的技术实施。取决于应用,VN有时也可以以等效或类似的名称提及。例如,VN可以用作虚拟专用网(virtual private network,VPN)、软件定义网络(software-defined network,SDN)、服务为中心的网络(service-centric network,SCN)、信息为中心的网络(information-centric network,ICN)、内容为中心的网络(content-centric network,CCN)、虚拟运营商网络、虚拟企业网、应用特定网络等予以实现。因此,本文所公开的术语VN可以涵盖这些等效物和变体。

[0025] 为了管理包含多个网络节点的网络,网络控制器可以建立为监管网络的中央实体。该架构可以用于物理网络和虚拟网络中。图2是图示VN方案架构200的示例实施例的示意图,VN方案架构200涵盖物理网络层和虚拟网络层。在大型网络或网络域中,可以部署多个网络控制器,每个网络控制器监管整个网络的一部分。如图2所示,第一物理网络控制器(physical network controller,PNC)220可以用于管理第一多个物理网络节点211、212和213,而第二PNC 230可以用于管理第二多个物理网络节点214、215和216。网络节点211至216可以被实现具有任意功能,例如通用路由平台(versatile routing platform,VRP)。每个PNC包含各种模块或单元。例如,PNC 220可以包含网络服务器221和多个执行PNC服务和其它服务的服务模块。

[0026] 实际上,相对较大的网络可以划分为多个较小网络或域,每个网络或域可由PNC管理。多个PNC,例如PNC 220和230可进一步由上级网络控制器,例如图2所示的VN控制器(VN controller,VNC)240管理以创建两层控制器分层结构。注意,若需要,可以构建更多层或级。应用(application,app)250可以通过PNC 220、PNC 230,和/或VNC 240访问和控制网络资源,它们可以通过开放平台服务器(open platform server,OPS)实施。VNC 240可以例如通过向PNC 230传输请求或命令247经由API246与诸如PNC 230之类的下级控制器通信。注意,控制器220、230和240可以位于相同物理位置(集中式设置)或不同位置(分布式设置)。应用250可以作为在由应用开发人员操作的网络设备或节点上运行的任意合适形式的软件程序予以实现。

[0027] VNC 240可以是与多个PNC协调的应用。VNC 240可以提供使应用开发人员受益的许多不同类型的服务,例如提供上级接口供应用创建、配置、更新和操作VN。在一示例实施例中,VNC 240包括执行各种任务的各种模块,这些模块可以包括网络服务器241、VNC服务模块242、拓扑抽象层(topology abstraction layer,TAL)服务模块243、一个或多个路由计算(route calculation,RC)服务模块244、VNC表示模块245,以及其它服务模块。网络服务器241可以通过API 252与应用250交互。具体而言,网络服务器241可以嵌入在VNC 240中以接收请求254(例如,RESTful API请求),随后将请求254调度给VNC 240中的其它模块。请求254可以包含以VNDL编写的程序代码,用于操作网络。VNC服务模块242可以促进PNC 220

和230的管理。此外,TAL服务模块243可以收集来自每个PNC的拓扑,随后将多个拓扑合并为整个网络的单个拓扑。应用开发人员可以使用拓扑作为协助工具,以构建较大尺寸的VN,例如那些横跨多个物理网络的VN。

[0028] RC服务模块244可以为API 252提供实现选择,其可具有实现选择的参数以便API处理使用。VNC表示模块245可以允许应用开发人员使用声明式编程语言编写程序代码,并且使用程序代码来促进VN创建和操作。从实际角度看,应用开发人员可以将VNC或PNC视为SDN控制器。例如,VNC 240可被视为访问物理网络资源的SDN控制器。VNC 240可以提供任意合适类型的API以与应用开发人员和其它网络设备通信。例如,API 252和/或API 246可以为REST或RESTful API,例如亚马逊和OpenStack所提出的EC2,这是一种无状态类型的API。注意,API 246和252的类型可能相同或不同。此外,即使API类型相同,但其格式可能不同,因为每个厂商可能选择添加新功能。实际上,不同的控制器可能具有不同的API。尽管本发明主要在针对诸如VNC 240以及PNC 220和230之类的控制器的RESTful API场景下陈述,但是只要应用250可以向诸如VNC 240之类的控制器发送以VNDL编写的程序代码,则可以使用任意其它合适类型的API。

[0029] 图3A是图示VN通信架构300的示例实施例的示意图。所属领域普通技术人员将理解架构200和300之间的相似性,因此,出于简洁性目的,后续描述将着重于不同或未覆盖的方面(相同原则适用于其它附图)。一个或多个应用310可以通过API 322与控制器320通信。控制器320可以是VNC(例如,VNC 240),其管理覆盖在物理网络332之上的VN 330。实际上,应用310可以通过任意合适类型的软件来实施并且可以在物理和/或虚拟节点上运行。

[0030] 在一示例实施例中,使用应用310的开发人员可以以声明式编程语言编写程序代码312。程序代码312可以通过API 322发送给控制器320,控制器320可包括VNC表示模块或单元324。在接收以声明式编程语言编写的程序代码之后,VNC表示模块324可以在程序代码上执行各种任务。声明式编程语言,本文中有时称为VNDL,可以采用任意合适的格式,例如XML格式,这样术语或合法的名称可以通过XML纲要定义。VNDL允许应用开发人员描述VN。

[0031] 本文所公开的用于操作VN的声明式方法可以优于传统过程化方法。图3B示出了一条以过程化方法编写的示例性伪代码350。以过程化编程语言编写的程序代码可能需要明确指定每个详细步骤来执行任务。如图3B所示,为了创建名为“red”的网络,伪代码350指定添加节点、节点上的端口,以及端口之间的链路的每个步骤。此外,为了处理网络“red”中发生的事件(例如,如果链路异常),伪代码350需要指定如何处理该事件的准确步骤。因此,传统过程化方法可能要求应用开发人员非常清楚VN的详情。相比之下,使用本文所教导的声明式编程语言操作VN可以通过不要求指定每个详细步骤来避免这些问题。

[0032] 在计算机科学中,声明式编程可被视为构建计算机程序的编程范式或样式,其中程序代码在无需描述详细控制流的情况下表示计算逻辑。这种声明式方法可以通过描述程序从逻辑和目标方面完成的内容,而非描述如何将其作为编程语言原语序列完成来最小化或消除副作用。如何完成程序的详情可以遗留给另一设备(例如,VNC控制器)来确定。

[0033] 应理解,存在大量编程语言,这些编程语言可以划分或分类为声明式编程语言。声明式编程语言的类别包括声明式标记语言、函数式语言、逻辑式编程语言,以及函数式逻辑编程语言等子类别。声明式编程语言的具体示例包括但不限于:Alpha、Atom、应用类型系统(Applied Type System,ATS)、Curl、分布式应用规范语言、被称为ECL的以数据为中心的编

程语言、Erlang、可扩展应用标记语言 (Extensible Application Markup Language, XAML)、框架逻辑 (frame logic, F-logic)、FXML、通用声明式语言 (General-purpose Declarative Language, GenDL)、Glowe、GOAL代理编程语言、有利于进行等式推论的 (Good For Equational Reasoning, Gofer) 语言、知识机器 (Knowledge Machine, KM) 编程语言、Lithe、Lucid、Lustre、Miranda、Pan、Qi、XML、QUILL、SequenceL、结构化查询语言 (Structured Query Language, SQL)、同步多媒体集成语言 (Synchronized Multimedia Integration Language, SMIL), 以及网络本体语言 (Web Ontology Language, OWL), 及其组合。另一方面, 过程化编程语言, 有时也称为命令式编程语言的示例可以包括 Visual Basic、C、C++、面向商业的通用语言 (Common Business-Oriented Language, COBOL)、Java、Perl、Python, 以及称为PHP的开源服务器端脚本语言。此外, 应理解, 本文所公开的声明式方法可以与现有过程化方法兼容并且在某些场景下可以用作互补技术。

[0034] 图4是图示应用410和网络控制器420之间的通信架构400的示例实施例的示意图。网络控制器420可以用作VNC 240或控制器320予以实现, 而图4中所示的其组成部分可以通过VNC表示模块和其它相关模块予以实现。在一示例实施例中, 应用410可以为各种VN任务生成以VNDL编写的命令或请求。例如, 应用开发人员可以使用应用410写入请求以创建和/或操作VN。应用410可以通过发送包含以VNDL编写的程序代码的命令或请求412调用控制器420。请求412可以以任意合适的形式打包或格式化, 其中的程序代码可能具有任意合适的格式或长度 (例如, 包含脚本、字符串, 或者代码)。应用410可以通过北向API 421传输请求412, 该API可以是无状态的API或RESTful API。

[0035] 嵌入在控制器420中的超文本传输协议 (Hypertext Transfer Protocol, HTTP) 服务器422可以接收来自应用410的请求412。包含VNDL代码的请求412随后被传递给主模块424, 其可开启和/或协调转换流程, 例如通过将VNDL代码传递给执行模块426。执行模块426随后可将VN代码传递给解析器428以解析VNDL代码。在解析期间, VNDL代码可以转换或变换为内部对象430, 其可被网络控制器420中的VNC内部API 432所理解。内部对象430可以表示以VNDL代码描述的VN和描述处理网络事件的活动。内部对象430可以保存在网络控制器420的对象目录中。只要网络控制器可以充分理解内部对象430以执行VN相关任务, 则内部对象430可以用作任意代码、数据结构、脚本、命令, 或其它信息予以实现。

[0036] 执行模块426可以了解如何处理VNDL代码定义的任务。具体而言, 执行模块426可以调用VNC内部API 432 (可被视为南向API) 以根据或基于从VNDL代码转换而来的内部对象430执行VN相关任务。VNC内部API 432可以执行各种任务, 例如创建、删除、更新、查询状态, 以及处理VN的各个方面, 包括节点、端口、链路、策略, 以及事件。

[0037] 如果某个事件发生在网络控制器420所控制的VN中, 事件处理器434可以相应地处理该事件。例如, 如果网络节点、端口, 或两个节点之间的链路异常, 则事件处理器434可以将事件信息传递给执行模块426, 执行模块426随后可以使用预定义的策略处理事件。网络控制器420中的平台可以通过任意合适的语言或环境予以实现。例如, Python运行模块436可以用于实施模块, 如图4所示。

[0038] 图5A和5B示出了描述VN的至少一部分或某些方面的示例性XML纲要500。所属领域普通技术任意将理解XML纲要500的语法, 因此后续描述着重于本文所公开的新颖特征。注意XML纲要500中所用的名称和设置是示例性的并且是非限制性的。XML纲要500使用元素来

定义用于描述VN的多个合法名称。例如,XML纲要500包括名为“network”的元素510(注意,元素510跨过多个代码行)。因此,当一条VNDL代码包含术语“network”时,其可被处理实体(例如,VNC)识别为VN的有效合法名称并且可进行相应地处理。未定义的术语,例如“networks”,可以标识为错误的。

[0039] 实际上,网络可由标识符(identification,id)进行标识并且可以包括其中的网络节点和链路。为了更为详细地描述VN,元素510可以包括定义其它合法名称的子元素或子代元素。例如,名为“id”的子代元素512可以指定VN的标识符,名为“node”的子代元素514可以指定VN内的节点,名为“link”的子代元素522可以指定VN内的网络链路或连接。

[0040] 由于节点可以由id、地址、节点名称,和/或端口进行标识,元素514还可包括描述该节点子代元素。具体而言,子代元素515、516、517和518分别描述该节点的id、地址、名称,和端口。此外,元素518包括描述端口名称的子代元素519。

[0041] 类似地,链路可能具有各种属性,例如源节点、目的节点,以及带宽。为了描述元素522所表示的链路,名为“from”的子代元素523指定该链路的源节点,名为“to”的子代元素524指定该链路的目的节点,名为“bandwidth”的子代元素525指定分配给该链路的网络带宽。

[0042] 图6A示出了描述示例性VN 650的一条示例性程序代码600,图6B示出了可以通过程序代码600创建的VN 650。应用开发人员使用本文所公开的声明式方法可以遵循XML纲要(例如,XML纲要500)来编写程序编码600。程序编码600可以由应用(例如,应用410)发送给网络控制器(例如,控制器420),网络控制器随后可以根据程序代码600创建VN 650。

[0043] XML纲要中定义的合法名称可以在程序代码中用大写显示。如图6A所示,节点、端口、链路和带宽(bandwidth,BW)等合法名称需要大写。基于程序编码600,待创建的网络具有示例性名称“Test”并且包括名为1至7的七个网络节点。例如,程序代码600的第610行指定节点1包括名为端口1和端口2的两个端口,其在图6B中示出,其中节点由用黑色填充的圆圈表示,端口由连接到圆圈的椭圆形方框表示。

[0044] 此外,链路可以在端口之间建立。例如,程序编码600的第620行包括“Link1.2=3.1”,这建立了一条从节点1:端口2到节点3:端口1的链路(图6B中示为链路652)。应注意,第620行还指定将1千兆字节(gigabyte,G)带宽分配给链路652。其它节点、端口和链路可以通过程序代码600以相同方式建立。

[0045] 所属领域普通技术人员将理解,以VNDL编写的程序代码可以完全可扩展以包含其它信息,例如处理事件和执行应用所设置的策略的描述。由于VN中可能发生各种事件,它们的处理策略可以由应用开发人员配置并在VNDL代码中指定。例如,当网络节点功能出现故障时,VNDL代码可以定义一种策略来恢复节点或找到替代节点。可以对端口或端口之间的链路进行类似处理。再例如,当物理链路利用率超过某个百分比时,策略可将相关VN链路从物理链路移至另一物理链路。又例如,当接口丢包计数达到某个阈值时,策略可选择使用另一接口。总体而言,本文所公开的原理可以应用于无数其它示例事件和策略。

[0046] 通过使用声明式编程语言,而非过程化编程语言,应用开发人员可以在无需知道或了解网络详情的情况下执行VN任务。转换VNDL代码的网络控制器可以确保所需网络和网络行为。应用开发人员可以使用RESTful API与VNC通信,并通过VNC操作VN。例如,VNDL代码可以用于查询节点、端口或链路的状态。VNDL代码还可以用于执行添加或移除特定节点、端

口,或链路等操作。

[0047] 声明式方法可以带来各种益处。程序员生产力可以提高(例如,实验中程序代码长度缩短了90%);由于程序员可以着重于声明式语句而非详细步骤,编程准确性可以得到提升;并且由于开发人员可以使用声明式语句来构建网络可视图,可视性可以随着网络描述和行为的简洁性得到提高。

[0048] 图7是VN管理方法700的示例实施例的流程图,VN管理方法700可以由网络控制器(例如,PNC 220、PNC 230、VNC 240、控制器320,或者控制器420)予以实现。方法700开始于步骤710,其中网络控制器可以通过AP接收来自应用的请求。该请求可以包括以声明式编程语言编写的程序代码,并且这些程序代码可以描述VN的至少某些方面。取决于应用的需要,程序代码可以包含不同内容。例如,程序代码可以包括创建VN的指令、操作VN的一个或多个策略,或者在VN中发生的事件的定义。在一示例实施例中,声明式编程语言可以使用含已定义的合法名称的XML纲要,并且在描述VN时使用的合法名称可以包括节点、端口、链路、带宽、策略和事件中的一个或多个。此外,应用可以远程耦合至网络控制器或者与网络控制器共置于同一位置,从而接收可以是远程的或是本地的。

[0049] 在步骤720中,网络控制器可以将程序代码解析为网络控制器的内部对象。内部对象表示程序代码所描述的VN的各个方面。解析可以以任意合适方式予以实现以将VNDL代码转换或变换为内部对象。

[0050] 在步骤730中,网络控制器可以使用网络控制器的内部API根据内部对象来管理VN,这些内部对象可被内部API所理解。应注意,管理VN可能涵盖创建VN、更新VN、管理VN的策略,以及处理VN中发生的事件。应理解,方法700用作示例实施例,因此替代方法可用于修改方法700,并且如有需要可并入额外步骤。

[0051] 图8是计算机系统或网络设备800的示例实施例的示意图。网络设备800可用作任意设备,例如本文所公开的运行应用的网络节点,或者网络控制器(例如,PNC 220、PNC 230、VNC 240、控制器320,或者控制器420),予以实现。网络设备800能够接收、处理和传输发往和来自网络的携带信息的信息,例如VN相关请求。网络设备800可以包括耦合到接收器(receiver,Rx)812的一个或多个入端口810,Rx 812可以用于从其它网络部件接收请求、响应、程序代码,以及其它信息。网络设备800还可以包括耦合到发射器(transmitter,Tx)832的一个或多个出端口830,Tx 832可以用于向其它网络部件传输请求、响应、程序代码,以及其它信息。

[0052] 网络设备800可以包括逻辑单元或处理器820,其与接收器812和发射器832通信。处理器820可以通过硬件或硬件和软件的组合予以实现。所述处理器820可用作一个或多个中央处理器(central processor unit,CPU)芯片、核芯(例如,多核处理器)、现场可编程门阵列(field-programmable gate array,FPGA)、专用集成电路(application specific integrated circuit,ASIC)和/或数字信号处理器(digital signal processor,DSP)予以实现。处理器820可以用于实施本文所述的任意功能模块或单元,例如网络服务器221、网络服务器241、VNC服务模块242、TAL服务模块243、RC服务模块244、VNC表示模块245、HTTP服务器422、主模块424、执行模块426、解析器428、VNC内部API 432、事件处理器434,以及Python运行模块436,或者所属领域普通技术人员已知的任何其它功能部件,或其组合。例如,VNC表示模块821可以并入处理器820中,并且可以用于处理进行VN任务的VNDL代码。

[0053] 网络设备800还可包括至少一个存储器822,其可用于存储诸如VNC的程序代码和内部对象之类的数据。请注意,实际上,可能存在网络设备800处理的双向业务,因此一些端口既可接收也可发送报文。在此意义上,入端口810和出端口830可位于同一处,或可视为耦合到收发器(transceiver,Rx/Tx)的同一端口的不同的功能。处理器820、存储器822、接收器812以及发射器832还可用于实施或支持本文描述的任意示例实施例,例如VN管理方法700。

[0054] 应理解的是,通过编程可执行指令和/或将可执行指令载入网络设备800,处理器820和存储器822中的至少一个会发生变化,从而将网络设备800部分转换成(例如具有本发明公开的功能的网络节点)的特定机器或装置。所述可执行指令可存储在所述存储器822中,并且加载到所述处理器820中执行。对于电力工程及软件工程技术来说基本的是,可通过将可执行软件加载到计算机中而实施的功能性可通过熟知设计规则而转换为硬件实施方案。在软件还是硬件中实施概念之间的决策通常取决于与设计的稳定性及待产生的单元的数目的考虑,而与从软件域转译到硬件域所涉及的任何问题无关。通常,仍在经受频繁改变的设计可以优选在软件中实施,因为重新设计硬件实施方案比重新设计软件设计更为昂贵。通常,稳定及大规模生产的设计更适于在如ASIC这样的硬件中实施,因为运行硬件实施的大规模生产比软件实施更为便宜。通常,一个设计可以软件形式开发及测试,且稍后通过熟知设计规则变换为对软件的指令进行硬连线的专用集成电路中的等效硬件实施方案。按照相同的方式,新型ASIC控制的机器是一种特定机器或装置,同样地,已编程和/或加载可执行指令的计算机也可视为一种特定的机器或装置。

[0055] 本发明的任意处理可通过使处理器(例如,通用CPU)执行计算机程序来实施。在此情况下,可以向计算机或使用任何类型的非暂时性计算机可读媒体的网络设备提供计算机程序产品。计算机程序产品可以存储于计算机或网络设备中的非暂时性计算机可读媒体中。非暂时性计算机可读媒体包含任何类型的有形存储媒体。非暂时性计算机可读介质的示例包括磁性存储介质(如软盘、磁带、硬盘驱动器等)、光磁性存储介质(如磁光盘)、只读光盘(compact disc read-only memory,CD-ROM)、可录光盘(compact disc recordable,CD-R)、可重写光盘(compact disc rewritable,CD-R/W)、数字多功能光盘(digital versatile disc,DVD)、蓝光(注册商标)光盘(Blu-ray disc,BD)和半导体存储器(如掩模ROM、可编程ROM(programmable ROM,PROM)、可擦PROM、闪存ROM和随机存取存储器(random access memory,RAM))。还可以向计算机或使用任何类型的暂时性计算机可读媒体的网络设备提供计算机程序产品。暂时性计算机可读媒体的实例包含电信号、光信号和电磁波。暂时性计算机可读媒体可以经由有线通信线(例如,电线和光纤)或无线通信线向计算机提供程序。

[0056] 本发明公开至少一项示例性实施例,且所属领域的普通技术人员对所述示例性实施例和/或所述示例性实施例的特征作出的变化、组合和/或修改均在本发明公开的范围内。因组合、合并和/或省略所述示例性实施例的特征而得到的替代性实施例也在本发明的范围内。在明确说明数字范围或限制的情况下,此类表达范围或限制可以被理解成包括在明确说明的范围或限制内具有相同大小的迭代范围或限制(例如,从约为1到约为10包括2、3、4等;大于0.10包括0.11、0.12、0.13等)。例如,只要公开具有下限 R_l 和上限 R_u 的数字范围,则明确公开了此范围内的任何数字。具体而言,在所述范围内的以下数字是明确公开的: $R =$

$R_1 + k * (R_u - R_1)$, 其中k为从1%到100%范围内以1%递增的变量, 即, k为1%、2%、3%、4%、5%……50%、51%、52%……95%、96%、97%、98%、99%或100%。此外, 由上文所定义的两个数字R定义的任何数字范围也是明确公开的。除非另有说明, 否则术语“约”是指随后数字的 $\pm 10\%$ 。相对于权利要求的任一元素使用术语“选择性地”意味着所述元素是需要的, 或者所述元素是不需要的, 两种替代方案均在所述权利要求的范围内。使用如“包括”、“包含”和“具有”等较广术语应被理解为提供对如“由……组成”、“基本上由……组成”以及“大体上由……组成”等较窄术语的支持。因此, 保护范围不受上文所陈述的说明限制, 而是由所附权利要求书界定, 所述范围包含所附权利要求书的标的物的所有等效物。每一和每条权利要求作为进一步揭示内容并入说明书中, 且所附权利要求书是本发明的示例实施例。对所述揭示内容中的参考进行的论述并非承认其为现有技术, 尤其是具有在本申请案的在先申请优先权日期之后的公开日期的任何参考。本发明中所引用的所有专利、专利申请案和公开案的揭示内容特此以引用的方式并入本文本中, 其提供补充本发明的示例性、程序性或其它细节。虽然本发明多个具体实施例, 但应当理解, 所公开的系统和方法也可通过其它多种具体形式体现, 而不会脱离本发明的精神或范围。本发明的实例应被视为说明性而非限制性的, 且本发明并不限于本文本所给出的细节。例如, 各种元件或部件可以在另一系统中组合或合并, 或者某些特征可以省略或不实施。

[0057] 此外, 在不脱离本发明的范围的情况下, 各种实施例中描述和说明为离散或单独的技术、系统、子系统和方法可以与其它系统、模块、技术或方法进行组合或合并。展示或论述为彼此耦合或直接耦合或通信的其它项也可以采用电方式、机械方式或其它方式通过某一接口、设备或中间部件间接地耦合或通信。其它变更、替换、更替示例对本领域技术人员而言是显而易见的, 均不脱离此处公开的精神和范围。

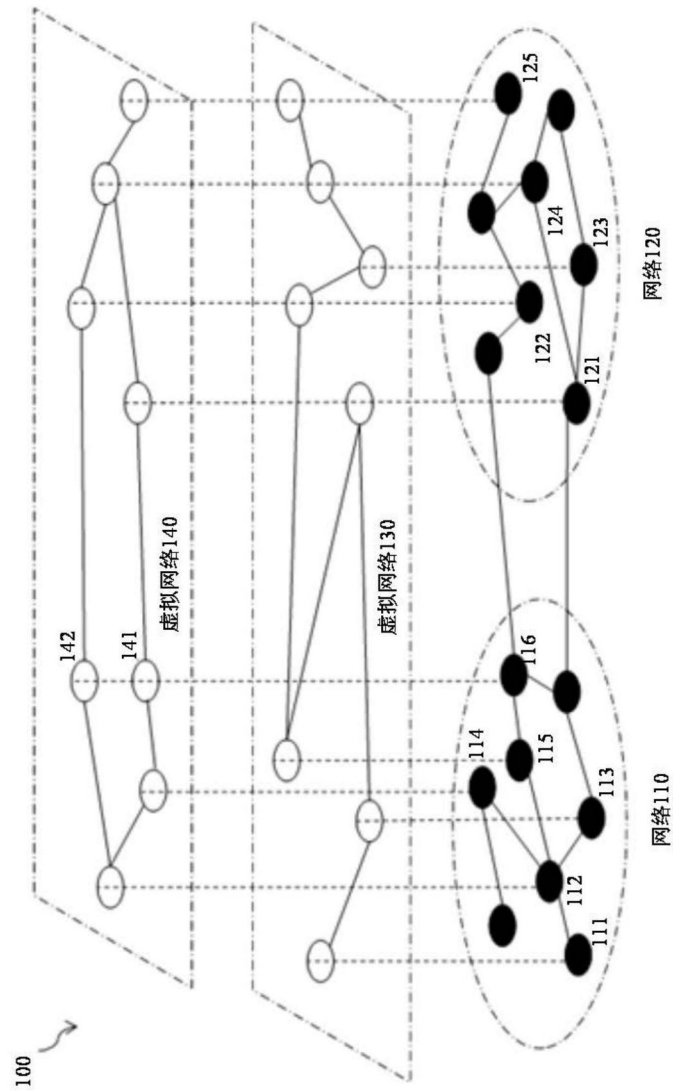


图1

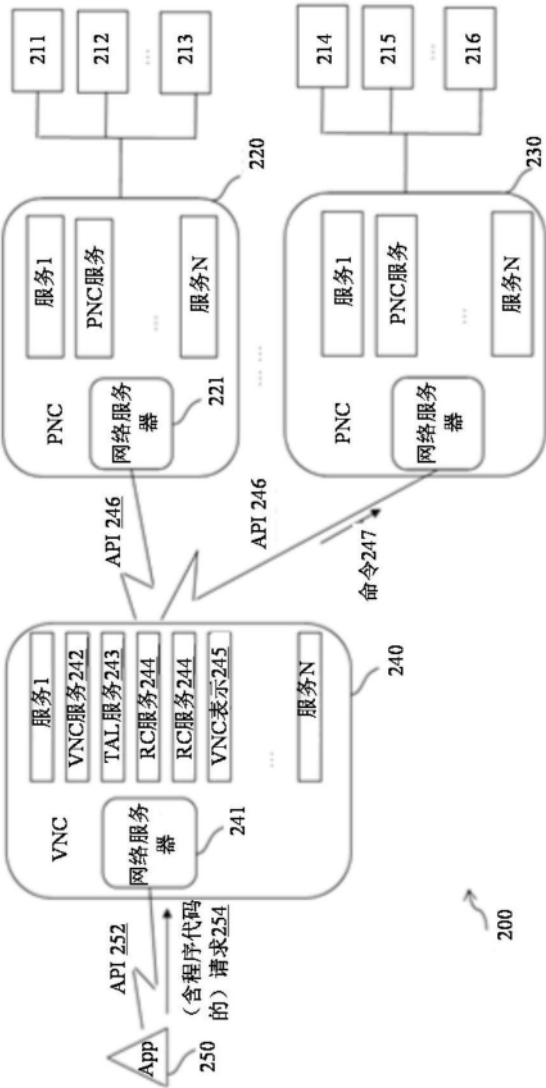


图2

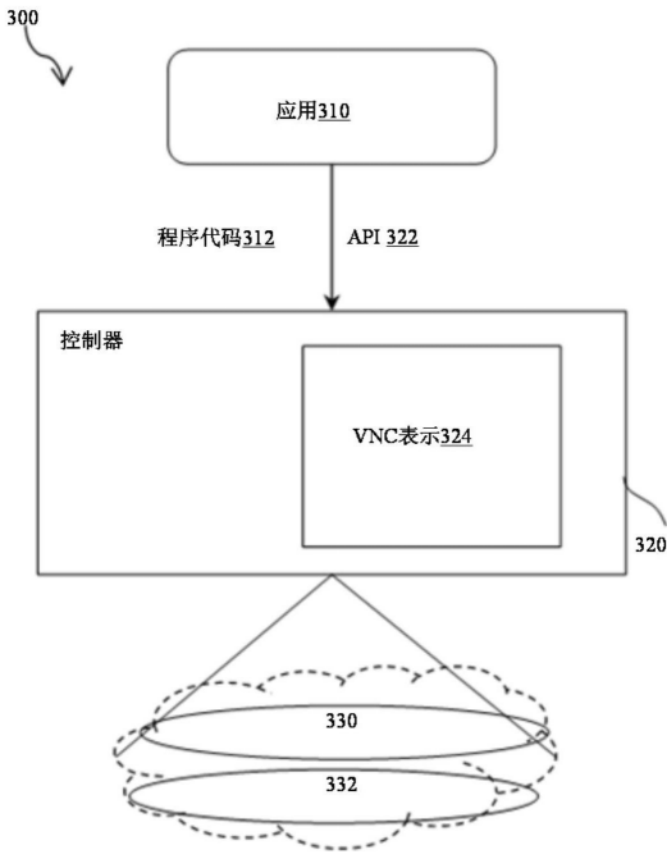


图3A

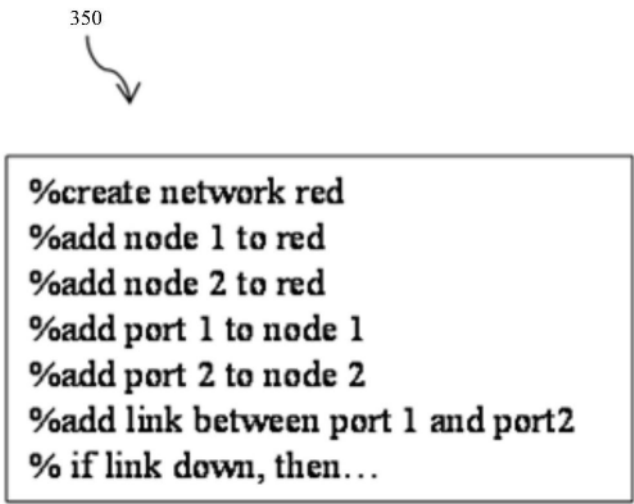


图3B

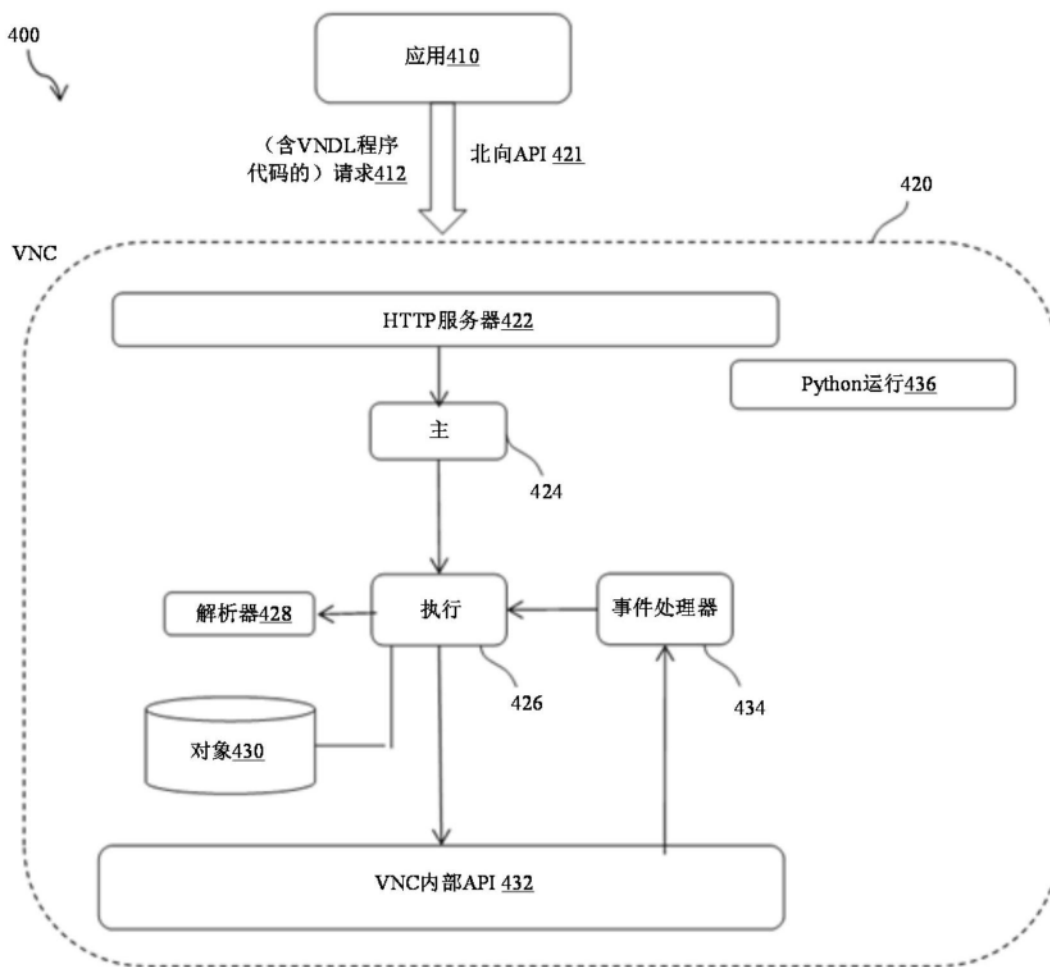


图4



图5A

500

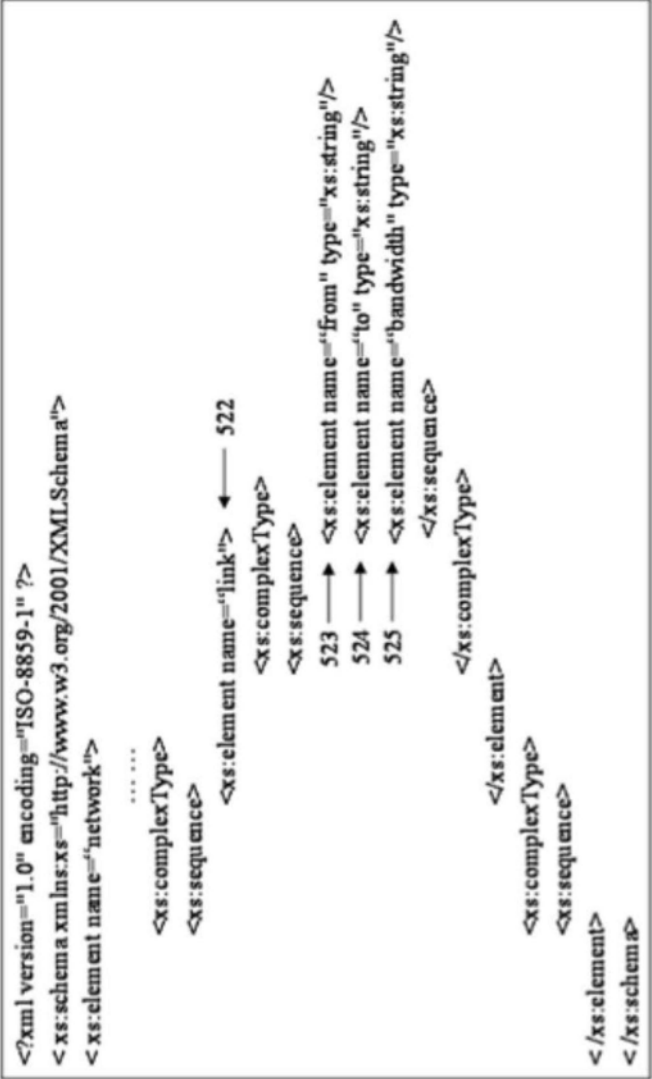


图5B

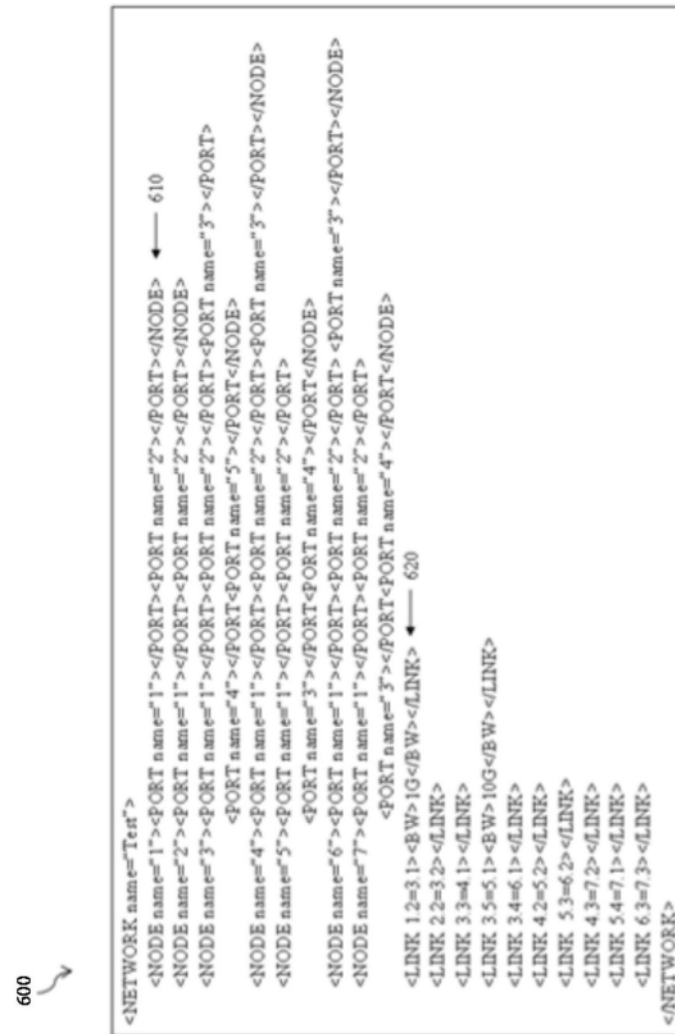


图6A

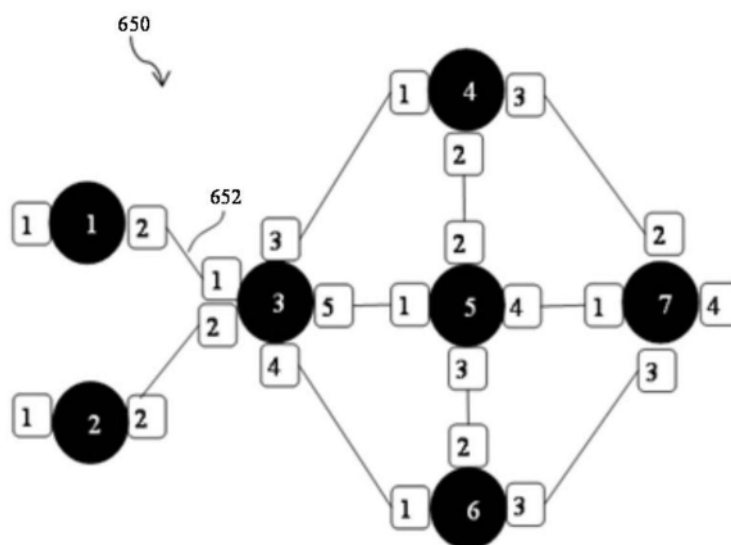


图6B

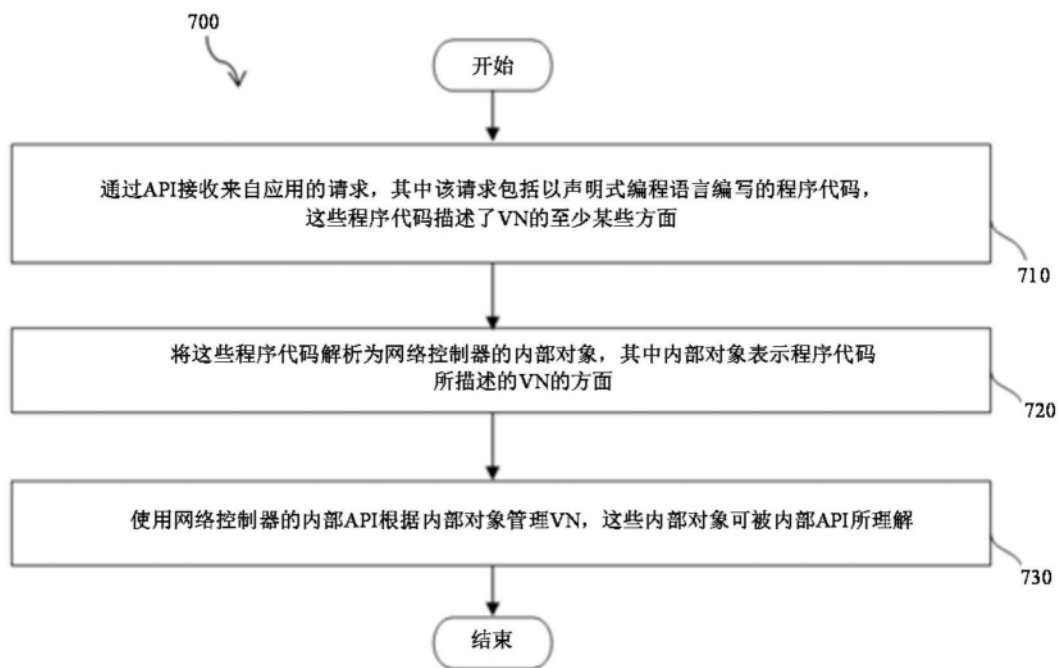


图7

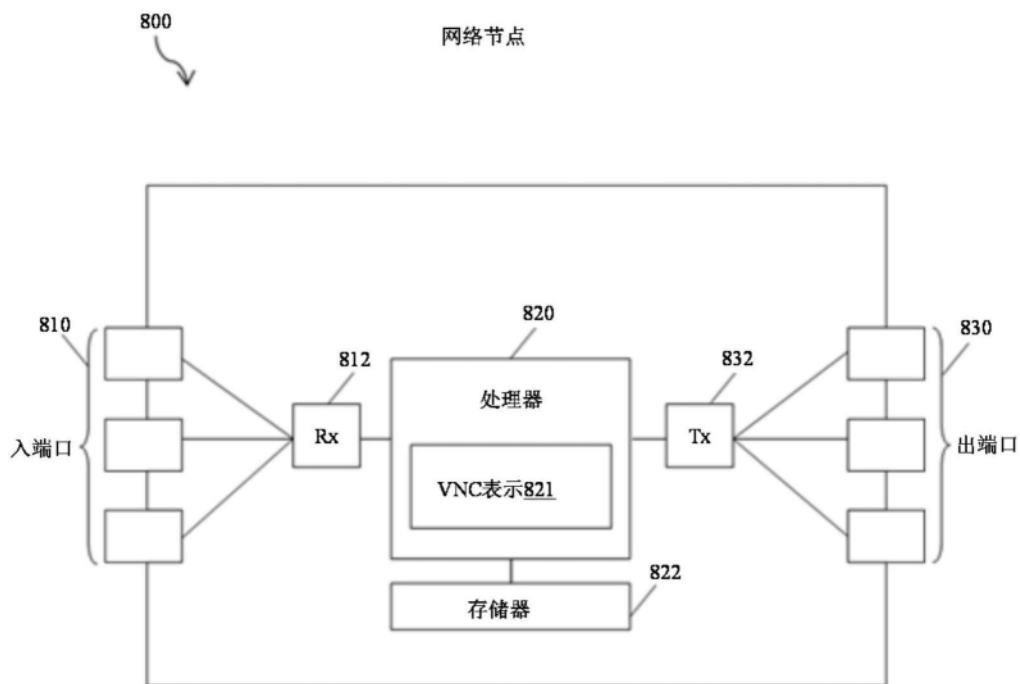


图8