虚拟二层网络交付方案 v1.0

14125224@bjtu.edu.cn

本文档简述虚拟网络综述以及应用于 NFV 场景偏二层的虚拟网络交付方案。

目录

1 引用

[1] NFV specification: Network Domain
[2] NFV specification: Infrastructure Overview
[3] RFC 6325 - Routing Bridges (RBridges) Base Protocol Specification
[4] RFC 5556 - Transparent Interconnection of Lots of Links (TRILL) Problem and Applicability Statement
[5] RFC 4762 - Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling
[6] 802.1ab specification ,Link Layer Discovery Protocol

2 缩写以及定义

NFV: Network Function Virtualization
VXLAN: virtual extensive LAN
IAAS: Infrastructure as a service
NFVI: Network Function Virtualization Infrastructure
VNF: virtualized Network Function
PW: Pseudo Wire
TRILL: Transparent Interconnection of Lots of Links
STP: Spanning Tree Protocol
ECMP: Equal Cost Multiple Path
SPT: Shortest Path Tree
ESADI：End Station Address Distribution Information
VNFC: VNF component
E-Line: Ethernet Line
E-Lan: Ethernet LAN
VPLS: Virtual Private LAN service
PBB: Provider Backbone Bridge
PE：Provider Edge
CE: Customer Edge
FIB：Forward Information Base
LDB：Label Distribution Protocol
BGP: Border Gateway Protocol
H-VPLS:　Hierarchical VPLS
VIM: Virtual Infrastructure Manager
LLDP: Link Layer Discovery Protocol
BEB: Backbone Edge Bridge
BCB: Backbone Core Bridge
OAM: Operation,Adminstration,Maintenance
BUM: Broadcast.Unknown,Multicast
FDB: forward Database

3 虚拟网络简述

对于虚拟网络[1]，通常是在原有的 ISO 七层网络模型的基础上，利用原有的各个层次功能并通过虚拟划分(virtual partitioning)、虚拟隔离(virtual isolation)、承载网络(virtual overlay)技术抽象和组建以满足多租户环境下的网络需求的一种网络模型，对于网络资源的复用和隔离是虚拟网络最基本的特征和需求。有两种类型：基于基础设施的虚拟网络(Infrastructure Based Virtual Networks)和分层虚拟网络(Layered Virtual Networks)。

在基于基础设施的虚拟网络中，通常用在 ISO 三层或者更高层的网络应用中，以**地址空**

[1] [1].6 Functional blocks within the domain

间划分(但不提供**流量隔离**)的手段来实施的虚拟网络。这种实现方法借助于访问控制列表(ACL)来明确限制需要放行或者阻断的流量，如 Linux iptables。

分层虚拟网络分为两类，即虚拟隔离和承载网络形式，下面逐一简述。

虚拟隔离是从网络基础设施的角度来实施的虚拟网络，即基础设施网络规定了流量的转发行为，典型的实现为 SDN 网络，以流表的形式对流进行匹配，并执行相应的策略，因为流表可配置，所以网络是可编程的。

虚拟承载网络是一种隧道技术，在 2/3 层网络中封装用户的流量使之能够适应不同的网络环境以及多租户的需求。封装技术比较常用的是 mac-in-mac 和 mac-in-ip，他们各自具有应用场景和实现开销。

虚拟网络用于云或者数据中心网络中[2]，也符合云尤其是基础设施云 iaas 中的虚拟资源具有特点。主要表现在：

1. 按需自提供(on-demand and self-service)
2. 资源池(resource pooling)方式提供
3. 快速弹性(rapid elasticity)提供
4. 计费(measured service)服务

其中，资源的按需并以资源池的方式提供主要使网络资源适应多租户的需求，快速弹性表现在资源的释放和开辟能够快速响应对变化的网络业务量。

3.1 NFV 简述

NFV 是 ETSI(European Telecommunications Standards Institute)2012 年起推出的一系列规范，NFV 旨在利用现有的标准 IT 虚拟化科技和标准的工业服务器来部署网络功能，从而形成新的电信行业的生态系统和加速电信行业革新。网络功能服务根据 IT 云的服务映射归纳起来有三种:

a) **NFVIaaS**(Network Function Virtualization Infrastructure as a Service): 与 IaaS 一样，该服务以基础设施的形式交付给用户，该服务并不提供虚拟网络功能(VNF)。
b) **VNPaaS**(Virtualized Network Function Platform as a Service):对应于 PaaS，以平台资源的方式交付给用户，用户可以根据需求定制 VNF。
c) **VNFaaS**(Virtualized Network Function as a Service):该服务直接给用户提供虚拟的网络功能，如带宽接入。

NFV 的框架结构如下图 1 所示:

[2] [2].6.2.1 NFVI and Cloud Computing

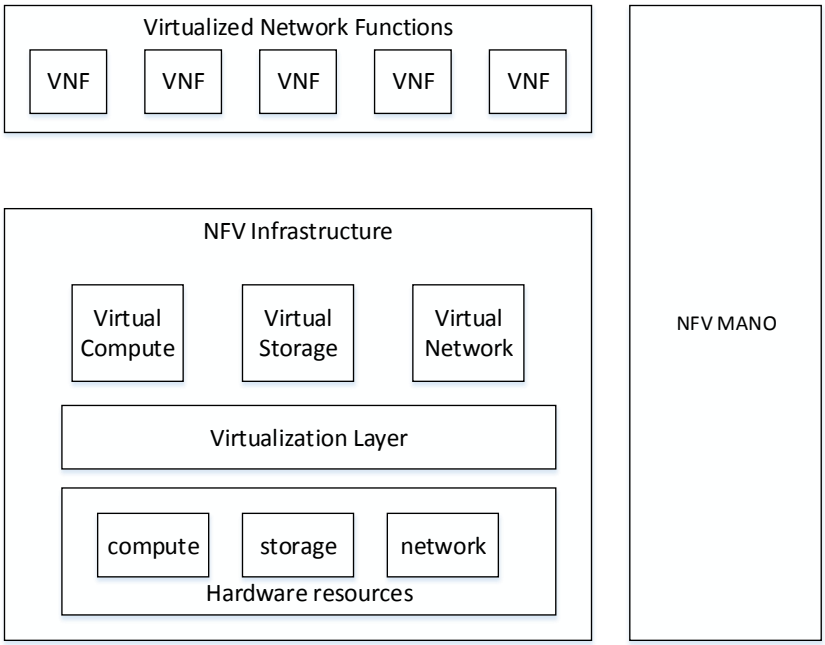

图 1 NFV 框架

NFV 规范从功能模块和引用点的角度出发说明 NFV 的框架细则并对基础设施域范围做出了划分[3]。主要有如下三个方面。

a) The compute domain.计算域，主要负责 host 部分，衔接 guest，并协调 hypervisor 域与基础设施网络域之间。
b) The hypervisor domain.虚拟机域，主要负责 VNF 的实现。
c) The network infrastructure domain.基础设施网络域，承载 VNF 的网络基础设施。

这三个域的详细介绍分别在 NFV 的三个域详细文档中。以及各个引用点的规范不再详述。NFV 的出现使得电信和云的关系交织得更为密切。

3.2 OpenStack Neutron 网络简述

在 OpenStack 的 Neutron 网络中，网络资源进行抽象和统一的管理,组成一个资源池提供用户。在 Neutron 中，网络资源以插件(plugin)的形式提供，插件分为两类:核心插件(core plugins)和服务插件(service plugins)。核心插件的主要功能是维护二层网络的状态，其主要维护的资源有如下：

网络(network):基础设施网的类型，如 flat,vlan 等，另外还有网络所属的租户信息。

子网(sub-network)：维护子网段,以及所依附的网络和所属的租户信息。

在 OpenStack Neutron Juno 版本后，为了整合不同的二层交换技术和虚拟交换厂家的特性，neutron 提出了 ML2 插件，该插件对核心网络做出抽象并规范响应的接口，极大的利于插件的定制开发。

服务插件依赖于网络核心插件，提供了路由(RaaS)，负载均衡(LBaaS)和 vpn(VPNaaS)等服务，下面以路由器为例说明 neutron 对 3 层网络服务的抽象。Neutron router 维护的资源有：

路由器(router),一个虚拟路由转发实体。

接口(interface)，路由器上的一个接口，通常映射为虚拟核心插件中**子网**。

网关(gateway),网关，严格地说也属于接口(子网)，其主要用于在 NAT 环境下的路由转

[3] [2].7 Domains of the NFV Infrastructure.

发。

端口(Port)，端口，带三层网络即 IP 地址的端点实体。

浮点 IP(floating-ip),本质属于端口(Port)，提供外部网络对 Neutron 网络的访问功能。

下面以 Open vSwitch 作为机制驱动(mechanism driver)来简述一个典型的 Neutron 网络的构成。

Neutron 使用了混合控制面的管理方法维护网络。需要三种类型的节点参与:

控制节点(Controller):主要负责网络资源的元数据维护操作。

网络节点(Network Node)：负责流量转发。

计算节点(Compute Node): 承担虚拟机流量的接入工作。

转发模型如图 2 所示：

首先，网络节点和计算节点通过管理网到控制节点维护相关的元数据，租户也通过控制节点提供的 API 对网络进行操作。

来自计算节点的数据流量通过网络节点集中转发，所以网络节点需要有 vSwitch 和 vRouter 的功能。

最后，通常来说，计算节点创建的接入网络节点的虚拟网络称为 Tenant Network,由管理员创建的网络称为 Provider Network。这两种网络分别由租户和管理员创建，有不同的操作权限，并且决定网络能否共享，能否作为 vRouter 的网关。在 OpenStack Juno 版本中，Tenant Network 类型有 vxlan、GRE、vlan、flat 和 local 类型，而 Provider Network 只支持 flat 和 local 类型。

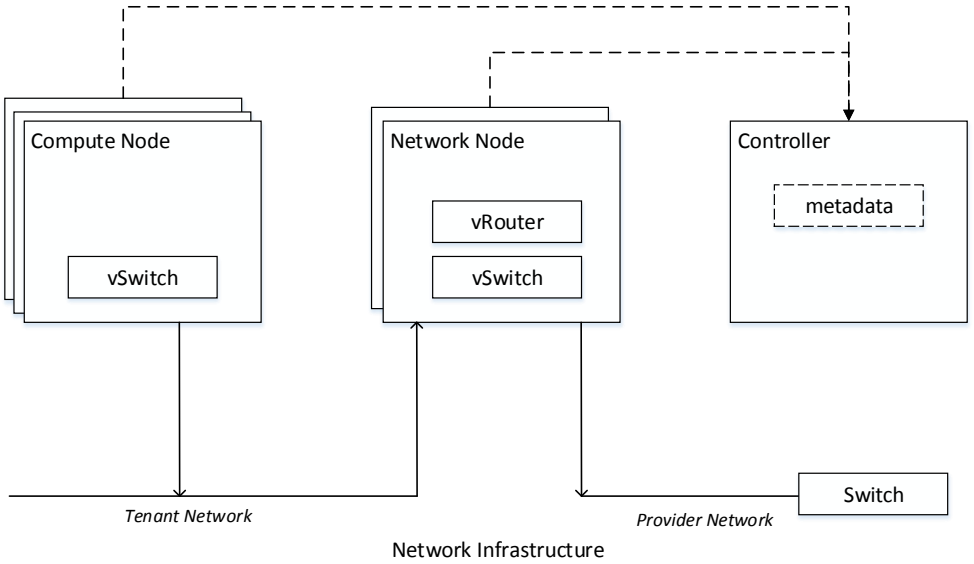

图 2 Neutron 转发模型

下面详述网络节点和计算节点内部的实现细则。

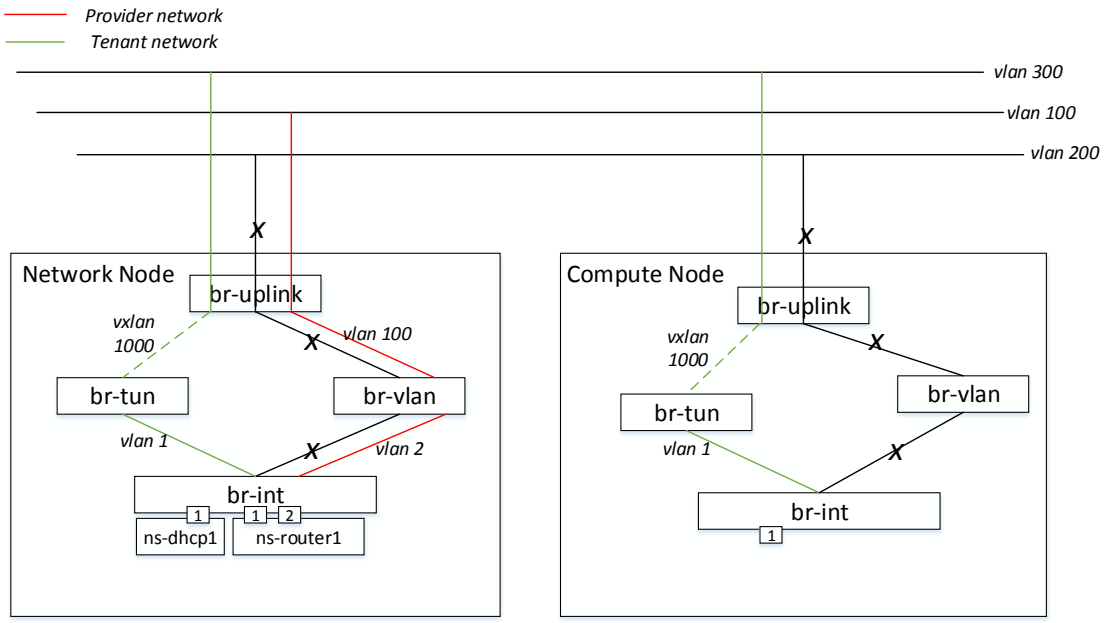

图 3 OpenStack 典型网络组成

如图 3 所示，在 Neutron 网络中创建了一个 vxlan id 为 1000 的 Tenant 网络，其链路层采用 vlan 300 来隔离；另外创建一个 vlan 100 来作为 provider 网络，该网络具有多个租户间共享的属性。

下面是几种类型的桥的介绍：

br-uplink:接入物理网络的桥，其中该桥包含物理网络设备接口。

br-vlan:vlan 转换桥，在这儿会对报文做全局与局部 vlan 的转换以便于后续处理。

br-int:集成桥，该桥负责虚拟机的接入和 DHCP 和转发等功能接入。

br-tun:隧道桥，该桥在 vxlan 或者 GRE 租户网络中转换隧道 ID 和局部 vlan ID。

OVS 实现的时候连接两个桥采用了 patch port，相当于传统的 Linux veth。对于每一个 3 层租户网络，会分配一个 DHCP 域，该 DHCP 程序 dnsmasq 运行在网络节点上，当有多个 dnsmasq 需要在同一个网络节点上运行时候，需要启用 Linux 命名空间来隔离。同样，多个路由功能需要隔离的时候也启用了不同的命名空间。

在网络节点和计算节点上，启动这不同的 L2/3 agent，这些代理通过控制节点的消息队列来传递消息并分配网络资源。

因此，可以看出 OpenStack Neutron 网络更像是一种面向租户的 3 层网络服务提供商。基于这一点，本文档提出了一种规则拓扑下的链路虚拟化解决方案，这是一种使用于 NFV 场景的二层虚拟网络。下面将逐一进行介绍。

4 总体模型概括

无论是运营商还是云和数据中心网络，根据网络业务量来适应网络功能的灵活性要求越来越来，这也是云的本质需求。另一方面，传统的交换技术也逐渐向智能化方向演进。TRILL 云交换正是在这种环境下出现。

在传统的二层网络中，保证网络冗余的措施通常有多链路运行生成树(STP)协议，在这个过程中，阻塞某些冗余的链路，当其他链路失败的时候，激活被阻塞的链路。从而保证网络有效的运行。

与此同时，由于链路拓扑变化会需要一定的时间才能收敛，并且最重要的一点是，被阻塞的链路不能转发流量，因此这部分带宽得不到有效的利用。尽管后续过程中出现了一系列的生成树协议变种，但是根本不能解决带宽的有效利用问题。

TRILL 协议中在协议感知链路拓扑后充分利用了所有的链路带宽，其在二层交换中引入了三层路由的概念来决定报文的转发行为。其抽象的模型如图 4 所示：

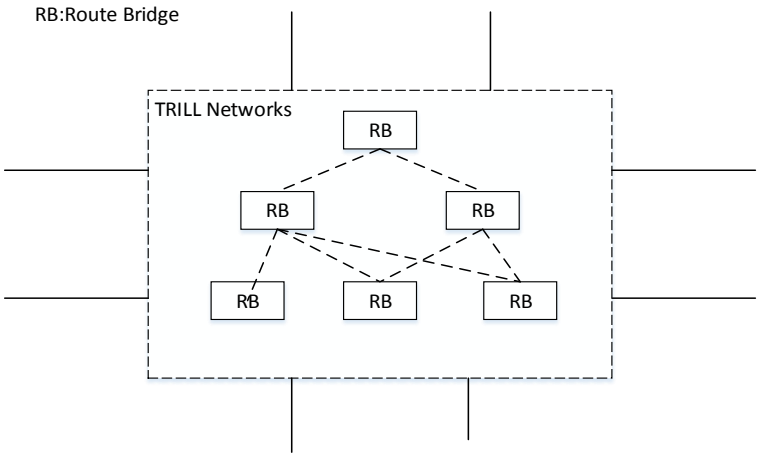

图 4 TRILL 抽象模型

在 TRILL 网络中引入了 RB(Route Bridge)的概念，RB 的功能为用户接入端点和报文的路由和转发。并且在 TRILL 网络中引入了 16-bit 的 Nickname，其功能类似于 L3 路由中的 IP 地址，但是 Nickname 根据二层协议协商得到。另外引入了 6-bit 的 hop count 字段，相当于 L3 路由中的 TTL，保证报文在网路中有限的生命周期，不至于网络环路的时候会无限的循环。

TRILL 中采用链路状态路由算法来路由报文，链路状态通过 L2 IS-IS 帧逐跳来传播，用户接入后通过 mac-in-mac 的封装形式，期间还有一层 TRILL 头部，其作用相当于 IP 头部，用来作为路由的根据。最后，TRILL 中端用户地址会根据 ESADI(通过 TRILL 封装)报文来广播。

TRILL 为了充分利用带宽，在路由单播报文的时候，对于来自不同用户的流量(通常采用 vlan 的形式区分，称为 c-vlan)，分别交给不同的计算出来的 ECMP 来做到有序的转发。在路由多播(广播和未知地址的单播)报文的时候，会计算一个最短路径树，并在上面转发报文。

因此，对于接入网络的用户而言，TRILL 看起来更像是一个智能的黑盒子，而 NFV 对此做了更高层次的抽象，其内部处理不局限在二层。如图 5 所示：

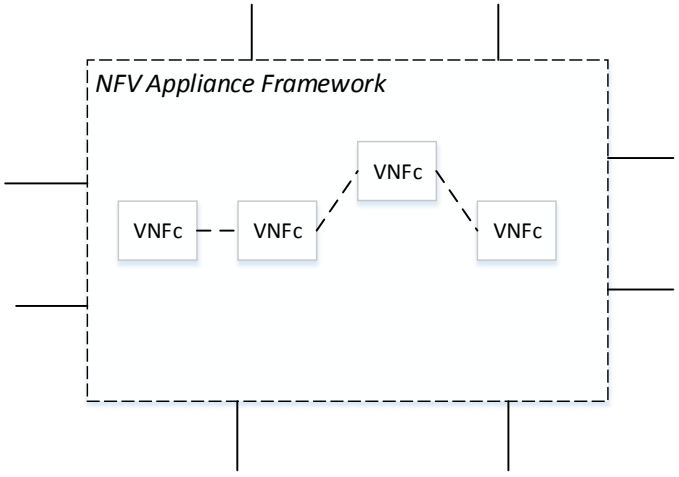

图 5 NFV 网络抽象模型

NFV 中虚拟的网络功能 VNF 通常含有多个组件，称为 VNFc，典型的 VNFc 有负载均衡，用户接入和网络功能(NF)。VNFc 的实现不再由基础设施网络负责，而是交由虚拟机来实现(很多情形下下也由物理机实现，这样的网络功能成为 PNF)。

之前说到，NFV 的实现不再局限于二层网络的处理，但依赖于二层网络作为最基础的载体，因此二层网络在 NFV 里面处于最核心的位置。如图 6 所示，NFV 的实现映射相应的服务模型。

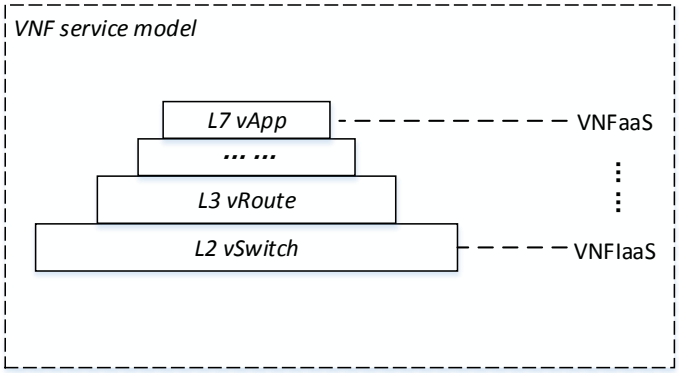

图 6 VNF 服务模型

其中最底层的 2 层虚拟交换服务作为最核心的网络功能，其负责网络拓扑结构的绘制，网络节点之间的流量传输和规划等工作，因此是一种 VNF 基础设施即服务模型。主要提供包括但不限于如下几种服务：

a) **E-Line**(Ethernet Line):提供点对点链路伪线(Pseudo Wire)服务。
b) **E-LAN**(Ethernet LAN):提供 LAN 交换服务。
c) **流量规划**:流量规划使得网络利用变得更为合理。
d) **流量工程**(可选):在二层实施 QoS，量化网络使之变得更精确。
e) **网络可视化**(可选)：网络可视化便于管理。

其基础设施节点模型如下图所示：

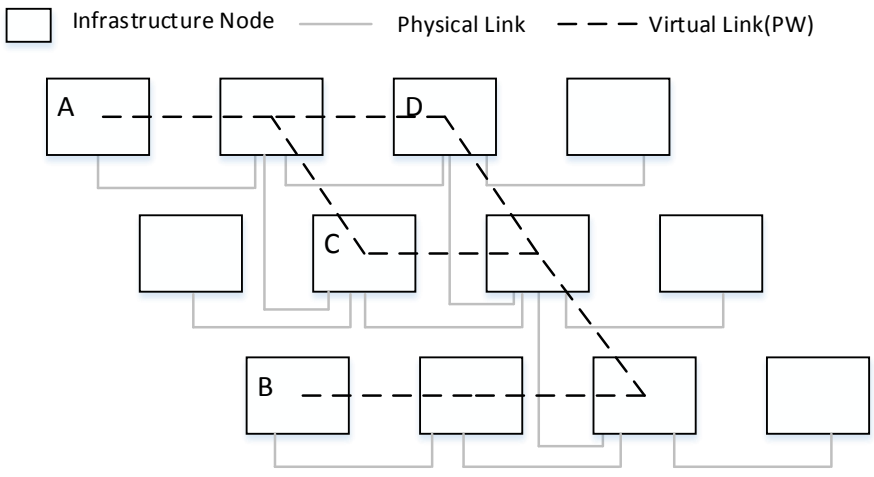

图 7 基础设施节点模型

其中基础设施节点构成了 NFV 中的计算域(Compute Domain),物理链路(包括基础设施物理设备)构成了基础设施网络域(Infrastructure Network Domain)的基础设施，基础设施网络域还包括虚拟链路(伪线,Pseudo Wire)。

图七所示中，基础设施网络负责建立一条由节点 A 到 B 的伪线，因此会根据拓扑计算一个最短的路径到节点 B,其中，由于经过 C 和 D 到节点 B 的代价是等同的，因此有两不同

的等价路径。

5 基础设施网络设计

在进行详细的模型介绍以前，下面先来简述 VPLS 模型和 PBB 封装。

5.1 VPLS 简介

VPLS 是一种在现有网络[4]的基础上模拟出多端点的虚拟私有的局域网服务的一种接口规范，其拓扑模型如图 8 所示:

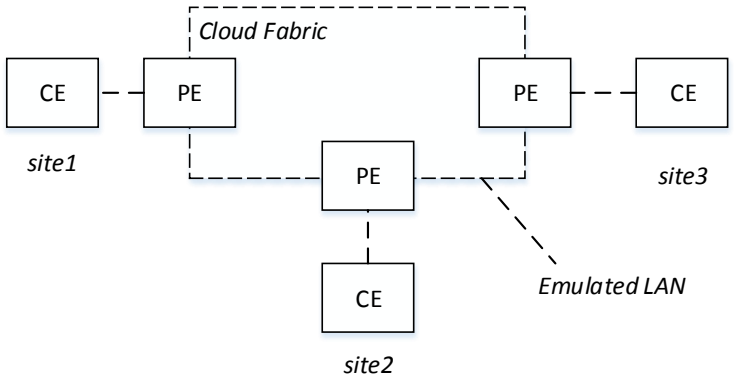

图 8 VPLS 拓扑模型

在 VPLS 网络中，要求任何两个 PE 之间均建立一条虚拟链路，即伪线。即 n 个 PE 之间存在 n*(n-1)/2 个虚拟连接。PE 负责存储客户端 mac 地址和对应的 PE 地址，称为转发信息基(FIB: Forward Information Base)。

在传统的方法中，通过分布式方法来建立伪线和传递转发信息[5]。常用的协议有信令分布协议 LDP 和边界网关协议 BGP。对于 VPLS 的封装，有两种形式:

a) 识别服务分隔符(Service Delimiter)的封装，比如 vlan 标签，此时 PE 负责删除这种该标签，并将该报文映射到对应的虚拟局域网上，TRILL 网络中的 vxlan-x 的指定转发 RB 正式采用这种方式来映射。

b) 不识别带服务标签，此时 PE 会根绝其他标识比如 access 端口来映射虚拟网络。

为了实现网络的课扩展性，出现了分层的 VPLS 网络 H-VPLS，在 H-VPLS 网络中，引入了面向网络的 PE(network facing PE,n-PE)和面向用户的 PE(user facing PE,u-PE)，一个 n-PE 会正常与其他 n-PE 建立全连接,但是一个 u-PE 会根据 VPLS 实例数与指定的 n-PE 创建伪线连接，从而减少了整个网络的信令开销。从而提高网络的可扩展性。

H-VPLS 拓扑模型如图 9 所示：

u-PE 与 n-PE 之间的伪线称为 spoke pw，因此 u-PE 不再和直属以外的 n-PE 建立伪线，极大的减少了伪线建立的开销。

值得一提的是，这种伪线的建立方法在传统网络中只能通过分布式的方法建立，但在云或者数据中心网络中，很可能采用混合或者集中式的控制面，因此伪线建立时候的开销将会大大的减少，因此伪线建立和转发开销将不再是网络可扩展性的主要因素。

---

[4] RFC 4762 规定，现有网络可能是 2/3 层网络，即 mac-in-mac 和 mac-in-ip 封装两种形式。

[5] 转发信息也可以通过自学习的方法来实现，详情参见 vxlan 的地址学习实现方式。

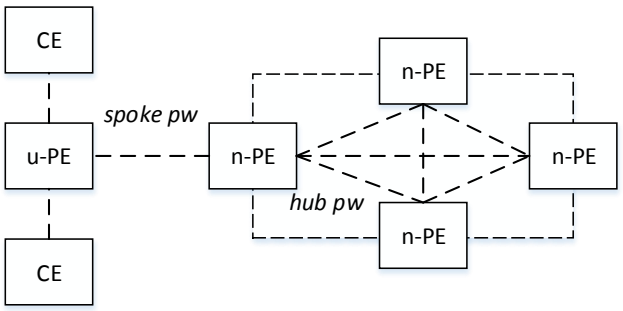

图 9 分层 H-VPLS 拓扑模型

5.2 PBB 简介

传统的网络中，通常会使用 802.1q VLAN 来隔离局域网流量，其主要做法是在链路层加入一个 tag 标签来实现。其帧格式如图 10 所示：

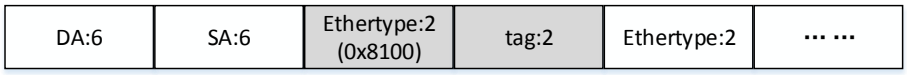

图 10 802.1Q 帧格式

通过 802.1q vlan 划分隔离或者 flat LAN 网络通常称为 Customer Bridge。服务提供商通常可能会扩展 Customer Bridge，因此出现了 QinQ 技术,QinQ 网络在服务提供商网络中透传用户流量，这种服务商网络被称为 Provider Bridge。进入 Provider Bridge 后，用户报文会被添加一层服务标签，该标签的以太网类型为 0x88a8，其帧格式如图 11 所示：

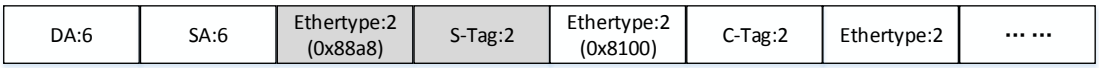

图 11 802.1ad 帧格式

QinQ 网络中采用了 2 层的 vlan 标签，因此也被称为栈式 VLAN(Stacked VLAN)，在实际实施的时候，会根据流量类型或者根据 c-vlan 类型来把用户流量映射到 Provider Network VLAN 上，进而根据外层 VLAN 应用 QoS 或者隔离流量。

一个显然问题是提供商网络因为采用 802.1Q 标签来，因此只能容纳 4095 个用户实例，因此 802.1ad 网络的实施一个可扩展性限制因素在于此。802.1ah 规范使得这个问题能够被解决，在 802.1ah 规范中，实施这种规范的网络称为(Provider Backbone Bridge,简称 PBB)，PBB 网路最核心的思想是解决服务提供商的客户实例可扩展性问题，因此不再使用 802.1q 标签来映射客户实例，而是采用一个新的标签，称为 I-TAG。802.1ah 帧格式如图 12 所示：

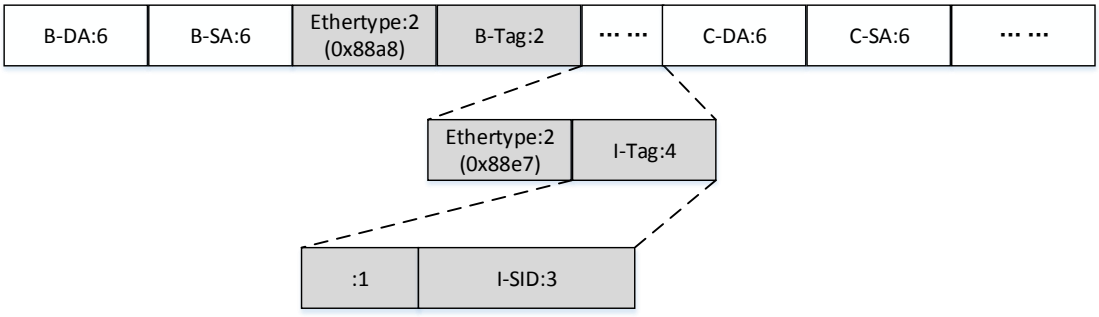

图 12 802.1ah 帧格式

在 PBB 网络中，B-Tag 可以看作是对用户实例的隔离，但不映射为任何客户实例，是

可选的。在 I-TAG 中，实例标识为 24 位，足够容纳很多情形下的用户。
关于 PBB 网络组建的细节相关部分不再述说，留待接下来的章节详述

5.3 网络功能模型

网络功能模型主要介绍本文提出的二层虚拟网络的功能模块和他们之间的引用点。根据 NFV 模型中的功能域的划分，其功能模块主要划分为 5 个部分:虚拟机(vm)、虚拟交换机(vSwitch)、物理链路(Physical Link)、现有基础设施网络(Existing Network)和虚拟资源管理者(Virtual Infrastructure Manager)。其模型如图所示：

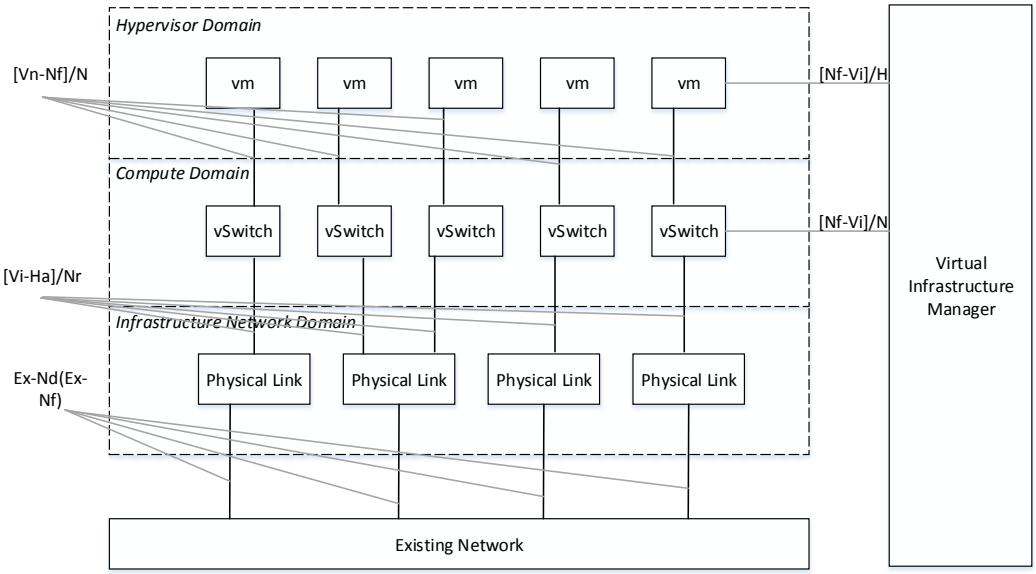

图 13 网络功能模型及其参考点

参考点规定了各个模块间的功能，其解决见表 1.

表格 1 参考点及其描述

参考点名称	功能描述
[Vn-Nf]/N	规定虚拟机接入虚拟网络的方式和虚拟机中 Guest/Host 的通信方式。
[Vi-Ha]/Nr	物理链路 Pseudo Wire 的实现方式，本文实现采用 802.1ah VPLS 实现。
Ex-Nd	负责建立基础设施网络拓扑等。
[Nf-Vi]/N	虚拟基础管理接口，协调虚拟网络的建立和维护。
[Nf-Vi]/H	虚拟网络元数据 metadata 查询。

接下来的章节会详述这几个参考点功能。

5.3.1 基础设施网络(Ex-Nd)

在 RFC 4762 中，VPLS 的实现中物理网络应该负责网络诸如 STP，SPB 或者 TRILL 来组建一个健壮的无环网络。因此本文中不再关注基础设施物理网络的组成，但会涉及到基础设施管理。

在本文中对基础设施的管理采用**混合控制面**的方法，其中分布式控制面用于建立和维护网络节点邻居关系，中央控制面用于拓扑建立和其他操作。

LLDP 是一种链路层协议，用户网络中两个节点间传输和协商节点信息，如节点标识，端口标识，端口描述，链路聚合，MTU 协商等，从而使网络设备能够建立起邻接拓扑。其报文

形式采用 TLV 的形式来进行封装，在一个 LLDP PDU 中，有系统标识(Chassis ID)，端口标识(Port ID)和 TTL。该报文不会被设备转发，但是设备会生成新的 MAC 地址作为标识。

其中，Chassis ID 设备会以中央控制面从中心控制节点分配一个全局的唯一标识(UUID)，端口 ID 通常为该端口的 MAC 地址，为了确保唯一性，设备在构成新的 LLDPDU 之前会向中心控制节点查询是否存在该节点，否则分配一个全局的唯一标识。

链路节点在接收到了 LLDPDU 后，解析协议数据单元携带的 Chassis ID 和 Port ID，并通过管理网络把邻接关系呈报给控制节点。典型的拓扑如图 14 所示：

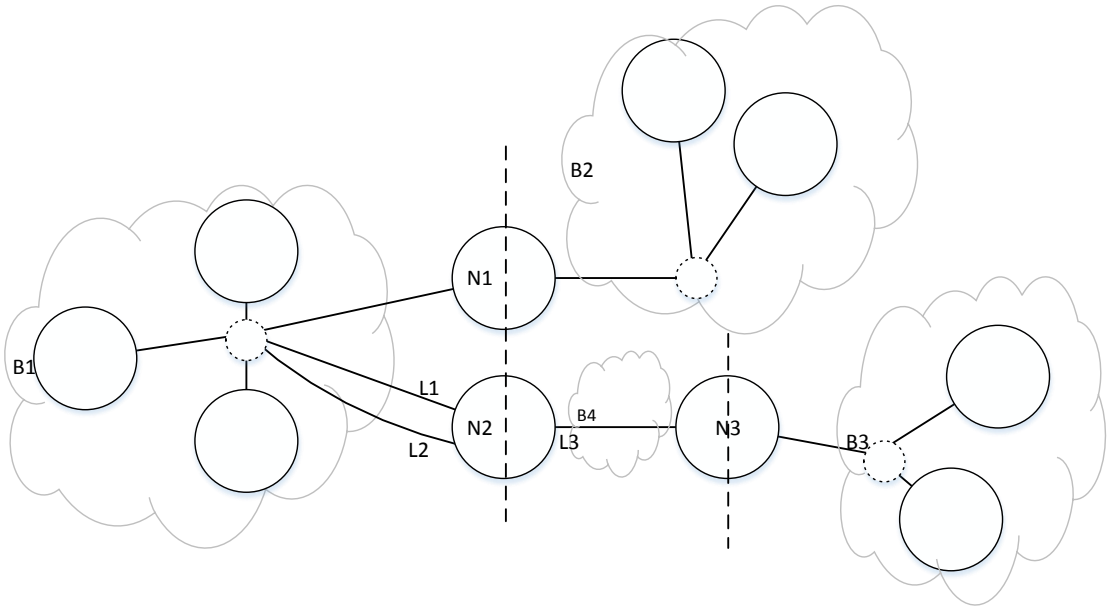

图 14 典型的基础设施网络拓扑

现做出如下几点说明：

a) 图中节点接入 LAN 中，因此 LAN 本身可以看成一个虚拟的节点，所以整个网络拓扑可以看成一个图。

b) 负责虚拟机接入的节点在本文中约定为接入节点(Access Point)，接入节点负责虚拟机接入网络。接入节点只能属于一个广播域内。图 14 中有 4 个广播域，分别标识为 B1,B2,B3,B4。

c) 接入多个广播域的节点称为连接节点(Adjacency Point)，负责多个连接多个不同的广播域的通道。在图中标识为 N1,N2,N3。

d) 无论是接入节点还是连接节点，均可以以多条链路接入同一个广播域，在该基础设施网络中， 多链路的角色一致，也是基础设施负载均衡以及等价多路径(ECMP)的重要支撑。在图示中标识为 L1,L2 和 L3，其中 L1 和 L2 都接入同一个广播域中。

e) 在多链路存在情况下，约定对于多链路接入同一个广播域中，在拓扑图中的度为 1。在拓扑中，任何度大于 1 的节点为连接节点。

拓扑维护工作包括不限于如下几方面的操作：

节点邻接信息和多路径检测:邻接信息和多路径用于确定一个广播域内的接入链路和连接链路。

节点广播域检测:其作为伪线等虚拟资源分配就近原则的依据。

等价多路径计算:以跳(hop)[6]为衡量单位，计算任意两个节点间(因为任意两个节点之间

[6] 考虑到云或者数据中心网络中网络部署的针对性和复杂性，接入链路和连接链路接入网络的速率不同，以跳为单位较为合理和简单，在实际部署中，也应该手动来部署和优化该网络，例如连接链路统一采用 10Gb 链路，接入网络采用 1Gb 链路。

很可能会建立一条伪线)的所有可能的最短路径(ECMP)，以及路径节点(链路)列表。

由上图还可以看出，使用连接节点的好处在于网络部署的过程中其扩展性制约因素，因为连接节点由于使不同的链路接入不同的广播域，因此可以使二层物理网络进行广播隔离。

关于拓扑实现的数据结构和算法留待设计文档中详述。

5.3.2 物理链路接入([Vi-Ha]/Nr)

物理链路接入的主要目的是使数据报文正确的在伪线段之间传输，因此规定了伪线的建立依据以及数据报文格式。

物理链路分为两种，能承载伪线的物理链路和承载原始报文的链路。其中承载原始报文的物理链路作为虚拟网络外部接入链路，是 Ex-Nf 的唯一实现，因此 vSwitch 必须有容纳原始的物理链路的接口。

对于物理链路上承载的伪线[7]，通过**逐跳**的方式传输，即**任何节点的伪线封装后的报文只能通过单播的方式传输到同一个广播域中的邻接目的节点**，目的节点再进行其他伪线的路由传输。本文中采用 802.1ah PBB 封装传输，下面详述封装过程。

在 PBB 规范中，用户接入的桥被称为 Backbone Edge Bridge，简称 BEB,服务提供商网络传输桥称为 Backbone Core Bridge，简称 BCB。BCB 桥在 PBBN(PBB network)中应该对客户实例(I-Tag)做出转发策略和 QoS 策略，但在本文实现中，基础设施物理网络采用标准的物理交换机，因此不用考虑物理交换机对 PBB 报文的转发影响。

在 PBBN 中，BEB 由两个组件构成，即 B-Component 和 I-Component。其中 I-Component 负责客服实例接入并封装好 I-Tag，I-Tag 会有至少两种方式接入客户:基于端口映射和基于 c-vlan 来映射。B-Tag 会把 I-Tag 封装后的报文根据配置和映射添加 B-VLAN 和 B_DA 以及 B-SA mac 地址。其模型如图 15 所示：

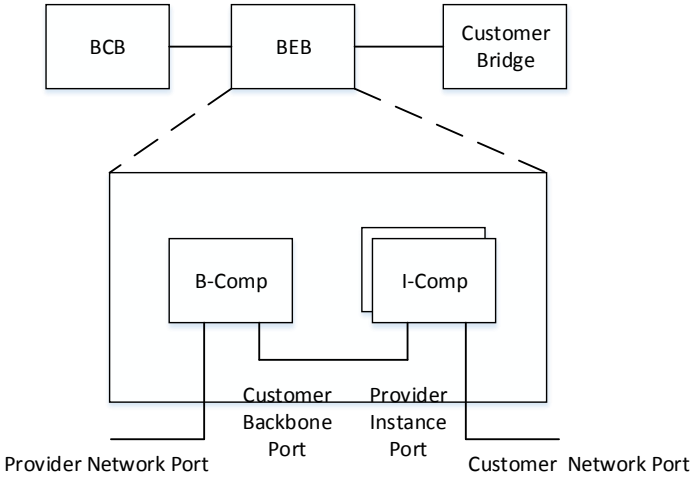

图 15 PBB BEB 组件图示

其中多个 I-Component 可能映射到一个 B-Component，通过 I-Tag 来辨别实例。在本文实现中，不再区分两个组件，统一对于任何一个要物理链路承载的伪线报文，经过 I-Tag 和链路层头(包括 B-VLAN)封装后，交给发送队列。同样，通过物链路承载的伪线报文经过解析后，交给 vSwitch 来根据 E-Line/E-LAN 配置进行下一步的转发。

在外层 MAC 封装的时候，源 MAC 地址为传输物理链路的的端口 MAC，目的 MAC 地址为同一个广播域内的目的节点(接入节点或者连接节点的 MAC 地址)，所以目的 MAC 地址也

[7] 除了物理链路承载伪线，guest/host 之间也承载伪线，[Vn-Nf]/N 中详述。

只能是单播地址。B-VLAN 作为可选的 Tag 用来进一步在物理基础设施上隔离属于不同 I-Tag 的流量。

下面简述 E-LAN 服务中的广播模型。由于本文中提出对伪线进行了完全的单播封装，且假定物理基础设施不会对 802.1ah 报文做出广播/多播复制行为，因此该工作由 vSwitch 实现，如图 16 所示：

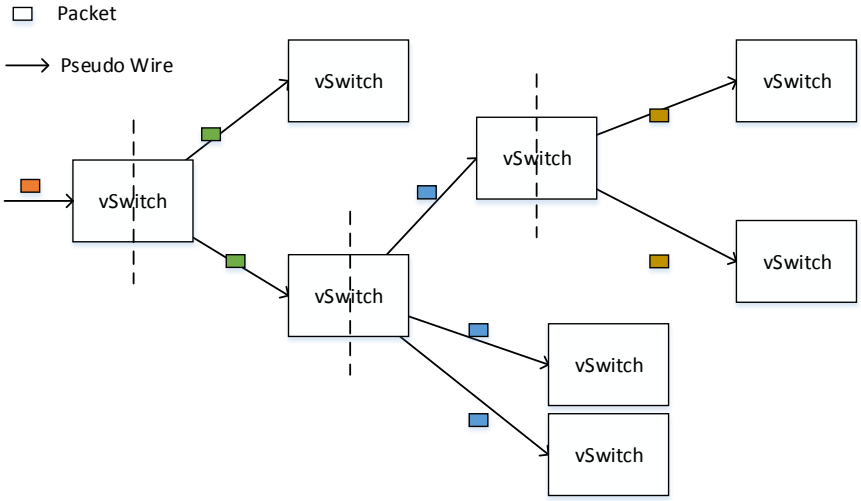

图 16 报文广播模型

在报文转发过程中，对于客户的广播(Broadcast)、未知地址的单播(Unknown Unicast)和多播(Multicast),下面简称 BUM 流量，物理基础设施不能感知客户流量类型，但是 vSwitch 必须能正确的转发该流量到所有的 E-LAN 中的客户中，这个过程由 vSwitch 来完成: vSwitch 解析客户目的 mac 地址，查询转发表，若为 BUM 类型，在根据 I-Tag 在本地除了接收伪线的所有伪线上复制一份报文并发送。因此，这种多目的端的报文最终能够到达所有的具有相同客户实例 I-Tag 的端点。

由于等价多路径是在伪线开辟前已经计算好，并且配置到伪线上，因此在 BUM 转发的时候不会出现路径不确定问题。

5.3.3 虚拟链路接入([Vn-Nf]/N)

虚拟链路是虚拟机与宿主机之间的报文通信线路，其有异于物理链路对报文链路的传输方式，虚拟链路是一种内存交换模型。但和物理链路一样，都对其接口做出了抽象，其接口包括但不限于发送、接收、承载伪线。下面以 qemu/KVM 为例简述虚拟链路的实现方式。

在 KVM 中，典型的网路分为网络前端(Frontend)和后端(Backend),前端网络主要给 Guest 呈现出网络设备接口以及传输通道，后端网络主要衔接来自前端网络的数据以及外部网络。

典型的前端网络实现方式有各厂商芯片模拟(如 Intel e1000,rtl8139)，virtio-net。其中厂商芯模拟的由 Hypervisor 来完成，正式出于模拟硬件设备的缘由，这种方法相对比较低效。因此出现了 virtio 通信方式，其采用了通用队列的方法来传输报文，不再模拟传统网卡的硬件特性，但是 Guest 前端网络需要有驱动来支持，所幸的是 virtio 已经纳入了 Linux 源码中，而 Guest 为其他的系统如 Windows 时，必须手动安装版本相关 virtio 网络驱动。

后端网络实现较为灵活，最常用的是 tap 设备，tap 设备通常和 Linux Bridge 桥[8]结合起来

[8] 为了更为高效的实现 tap 和 Linux Bridge 的功能，Linux 中引入了 macvtap，在 macvtap 中，转发不再根据 mac 地址表，而是依赖于其依附的设备，因此转发更为高效。

使用，如图 17 所示：

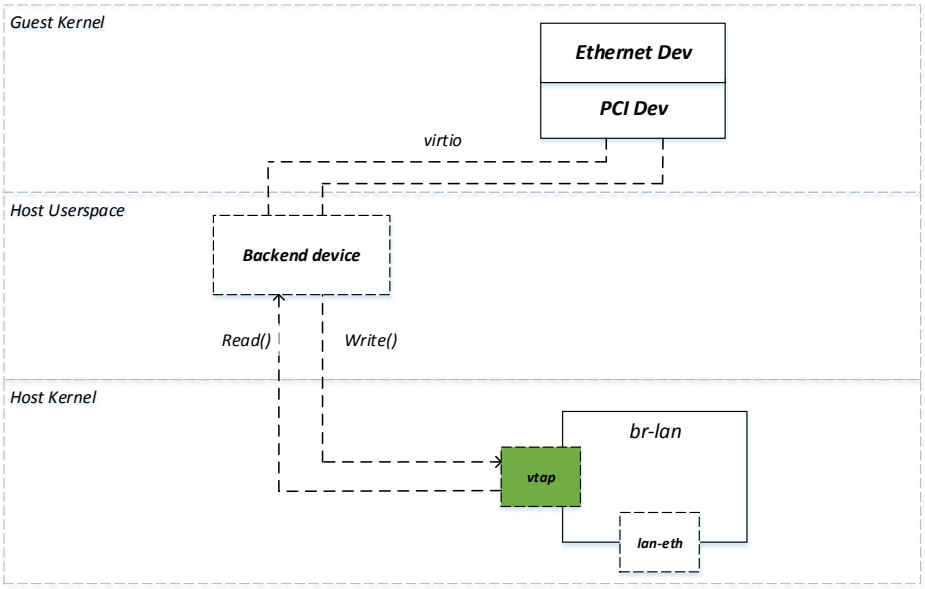

图 17 virtio-net 后端设备报文流程

其中在 Linux 中可以通过 tuntap 设备的方式从内核态提到用户态，其通过字符设备的方式 ioctl 创建虚拟网络设备 vtap，字符设备的 read/write 方法映射为虚拟网络设备的写和读方法。另一方面，使 vtap 作为桥的接口，使得流量能够正确接收。

宿主机用户态中，Qemu 实现了后端设备，其主要功能为从 vtap 中读取和发送报文，并通过 qemu 提供的 virtio API 传输该报文。关于 virtio 的传输机制这里不再介绍。这种方式报文经过一次拷贝到用户态，为了减少这种开销，出现了 vHost，使得报文不经过 Host 用户态经过拷贝送到 Guest，其流程如图 18 所示：

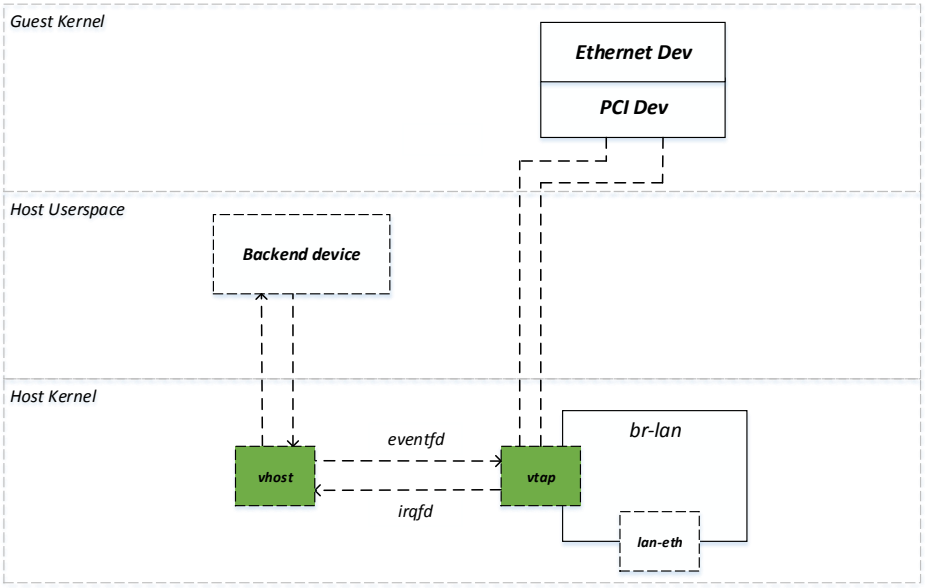

图 18 vHost 后端设备报文流程

在 vHost 环境下，报文直接从 vtap 映射到 Guest 的地址空间中，因此不再拷贝，极大减少拷贝开心，另外，在 vHost 中，Qemu 中的事件和中断句柄均由 kernel 中的 vHost 代理。

以上所述的两种方法，均是一种通用的包处理方法。基于一个事实：Host 中数据面处理能够在用户在用户态中高效的实施，如 Intel DPDK，其减少报文拷贝的方法是使用内存映射。

因此如果能使 Kernel 的内存通过映射为 Userspace 的地址，这部分地址再映射为 Guest 的物理地址，那么 Guest 用户态对这部分地址的访问将完全是一个零内存拷贝。

所幸在 Qemu 中，ivshmem(Inter-VM Shared Memory)机制使得这一切都称为现实，在 ivshmem 中，多个 Guest 共同使用一片 Host 内存空间，并且使用了 Guest 之间的 eventfd 和 irqfd 来通知，其原理稍后剖析。所以 ivshmem 通常用来作为一种 Guest 之间的零数据拷贝和通信手段。

Nahanni 是一个基于 ivshmem 的接口，于 2010 年 KVM 论坛大会上提出，其实现了 Host 的通信服务器，以及 Guest 的 PCI 设备和字符设备接口，因此 Guest 可以更容易访问共享内存。Nahanni 的传输性能由于较少的封装，其性能远比 virtio 和其他几个商业软件更高。详情参见：http://www.linux-kvm.org/page/KVM_Forum_2010。

Intel DPDK 专门针对 Qemu/KVM 做出了一套 guest 优化代码，其重写了 ivshmem 中的内存映射部分，将 DPDK 的大内存页映射到了 Guest 中，映射后的内存由 Guest 中的 ivshmem EAL 检测，详细参见：https://github.com/chillancezen/ivshmem-guest-code。

下面介绍一般 Guest/Host 零拷贝内存通信方法，由于 Guest 直接操纵 Host 的内存，因此在 Guest 必须是可信的，在 NFV 场景下，Guest 满足此要求。如图 19 所示:

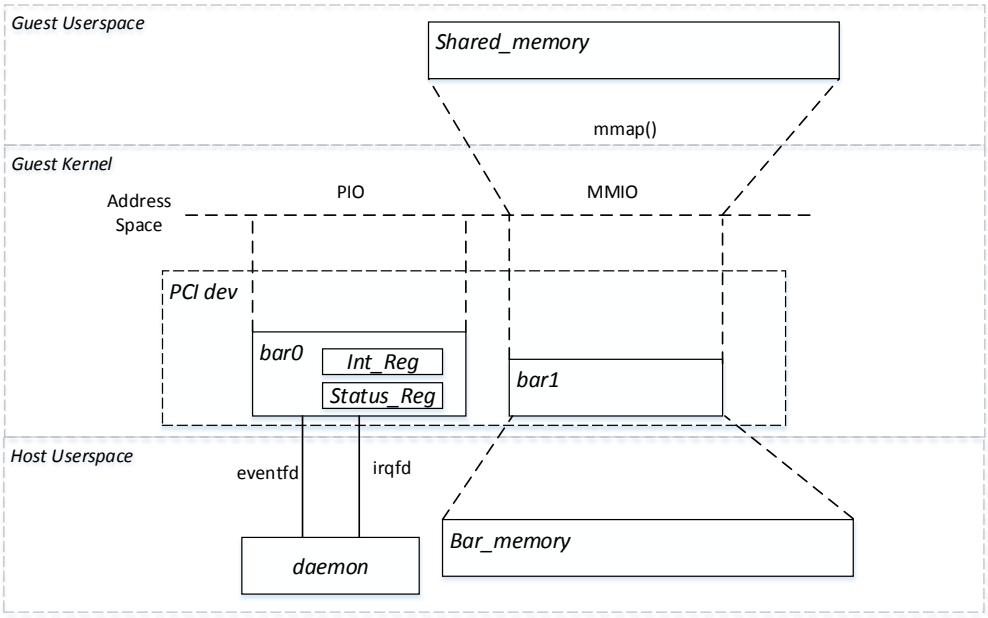

图 19 Qemu Host/Huest 内存交换示例

现做如下几点说明:

1. 共享内存的方法总体来说是模拟 PCI 设备实现，即 Hypervisor 来生成这个 PCI 设备。
2. 其中 PCI 设备中注册两个 region，简称 Bar，其中 bar0 中为寄存器状态，Hypervisor 负责对这些寄存器的读写操作。最基本的寄存器有两种:int_reg 中断和 status_reg 状态寄存器。其中中断寄存器指明是否发生中断(只有在共享中短线模型中使用，在消息信令中断(Message Signaling Interrupt,MSI)模式下不用到)，状态寄存器指示中断发生的理由；bar1 为内存映射区域，Hypervisor 负责把 Host 进程空间内存地址映射为 PCI 地址。
3. 对 bar0 和 bar1 的地址访问有区别的对待，在访问 bar0 前，PCI 驱动程序应该有把这部分地址映射为端口内存(Port based IO Memory,PIO)，对于 bar1，驱动程序应该这部分地址映射为基于内存的 IO 地址(MMIO)。
4. Irqfd 由 Host Userspace 中的 Hypervisor 触发，Qemu 提供触发 PCI 设备中断的 API，在此前应该设置好相应的寄存器。后端应用程序应该获取 Qemu 设备的事件通知，

这个过程由寄存器写操作来完成，在 Qemu 中，提供了写操作值和掩码的匹配以及基于 Unix Domain Socket 的 eventfd 接口。

5. Guest 用户态可以通过内存映射的方法把 bar1 中的地址空间映射到 Guest 用户态中，因此 Guest 用户态完全能够零内存拷贝访问 Host 内存。

本文中，虚拟链路接入中的链路组织和抽象留待详细设计中介绍。

5.3.4 虚拟交换机 OAM 接口([Nf-Vi]/N)

首先，规定 vSwitch 只能容纳三种链路类型：

1) 物理链路(Physical Link)。
2) 物理承载伪线(Physical PW)。
3) 虚拟承载伪线(Physical PW)。

下面对这三种链路不再区分，统一为虚拟链路，简称 vlink。

因此，vSwitch 模型很容易出来了，如图 20 所示：

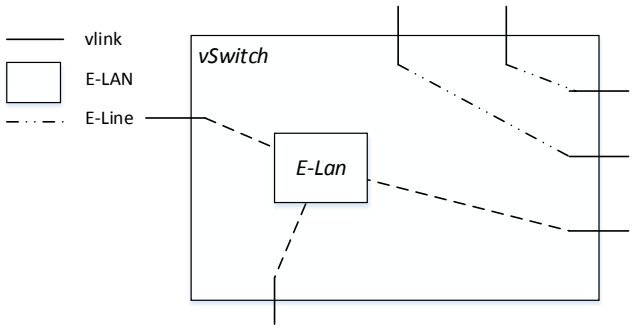

图 20 vSWitch 链路模型

其中 E-Line 服务模型，在 vSwitch 中直接关联两个链路。直接转发单播和 BUM 报文。E-LAN 模式中，根据客户 C-MAC 和客户 C-VLAN 来进行转发。因此，对于每一个 E-LAN 实例，vSwitch 必须维护一个转发数据库 FDB，这部分设计和文档留待详细设计中详述，下面说明 vSwitch 必须满足的管理面的功能。

a) 虚拟链路分配。
   如上所述，虚拟链路有三种类型，因此该管理功能实际上涉及到物理链路接入 vSwitch，物理和虚拟承载伪线的接入。对于伪线接入，必须要使得指明伪线所依赖的物理链路，链路端点 MAC 地址和实例标识符 I-SID。
b) 交换实例指定。其中交换实例分为 E-LAN 和 E-Line 两种。对于 E-Line，关联要进行两个虚拟链路 vlink。对于 E-LAN，对每个交换实例建立相应的转发表。
c) 转发表项问题。对于 E-LAN 的转发表项，可以通过中央控制面来配置，也可以通过自学习的方法。其中央控制集中配置下方保证进入网络端点的 MAC 地址和 VLAN 表项，具有较强的安全性，与此同时也限制了进入 E-LAN 的客户端点具有严格的地址类型，因此不能兼容标准的二层交换规范。
d) 链路迁移问题。在云或者数据中心网络中，虚拟机很可能会迁移，因此在虚拟机迁移或者数据链路失败后，虚拟链路应当能够正确释放和重新开辟。 其具体流程留待详细设计中解决。
e) 定位链路错误。当链路发生错误使，正确回收资源重新分配资源。

虚拟资源管理器能够全局的掌握虚拟资源，并分配和定向资源，因此尽可能的向链路可视化设计。

最后一个管理面的接口是[Nf-Vi]/H，该接口主要规定了虚拟机对链路元数据的查询。

6 网络设计原则

本文中提出的虚拟二层网络在伪线实施的时候应当尽可能考虑两个原则：负载均衡和量化原则。下面分别作出简单介绍。

6.1 负载均衡原则

本文所考虑的基础设施伪线层面的流量负载均衡，并不考虑承载的实际应用。在开辟伪线的时候，考虑两方面的内容：基础设施拓扑和当前链路可承载容量。如图 21 所示：

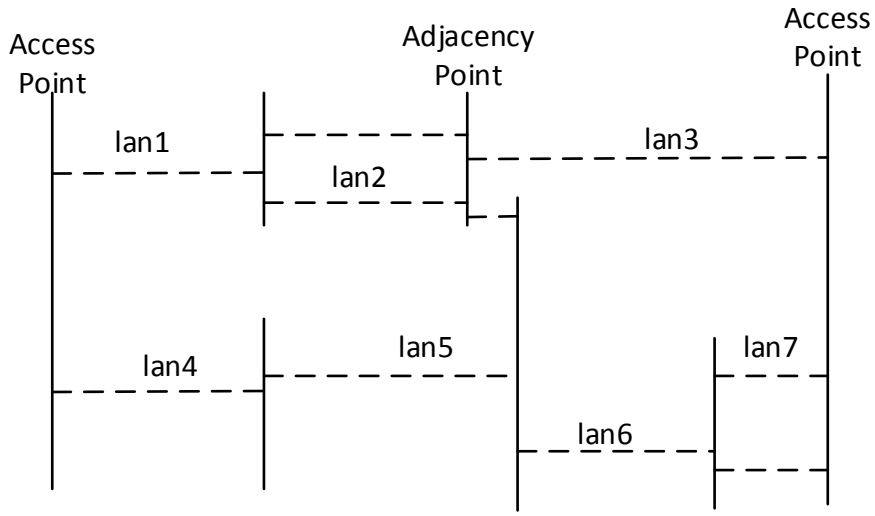

图 21 拓扑负载均衡示例

图中两个接入节点(Access Point)和 5 个连接节点。两个连接点之间经历的简单路径有 4 条:

a) (lan1,lan2,lan3).
b) (lan4,lan5,lan3).
c) (lan1,lan2,lan6,lan7).
d) (lan4,lan5,lan6,lan7).

其中 a)和 b)路径长度为 3，c)和 d)路径长度为 4。因此仅以 lan 的跳数作为路径代价的时候，a)和 b)属于等价路径，c)和 d)属于等价路径。

在实际实施伪线的时候，应当充分考虑路径上各个连接节点当前负载情况。其因素包括但不限于如下：伪线数，伪线需要带宽，伪线物理链路带宽。其具体的衡量原则和计算模型留待详细设计文档介绍。

6.2 量化原则

量化原则一方面要求在基础设施分配上通过统计的方法合理利用基础设施部署虚拟网络，如 6.1 章节中的负载均衡。另一方面对租户而言，由于计费的需求使得对网络比较有明确的量化指标，如最小带宽保证，最大带宽限制等。

本文主要通过流量整形的方法来约定伪线带宽。如图 22 所示：

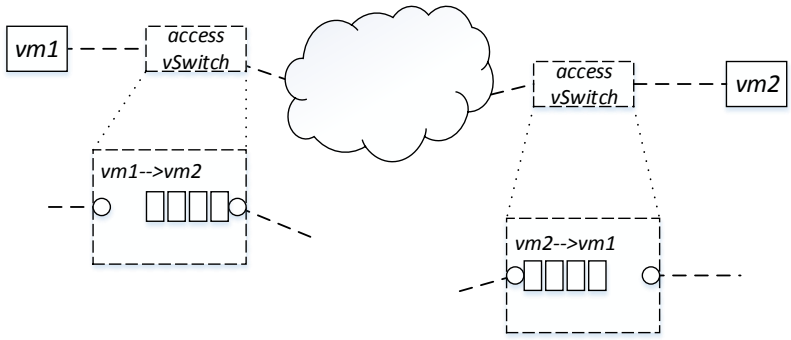

图 22 流量整形实施点

在云或者数据中心网络中，流量分为东西向流量和南北向流量，在南北向流量中容易定义流量方向(inbound/outbound)，正如 5.3.2 节所述，外部流量也通过 vSwitch 接入虚拟网络，因此本文接下来不再以 in/outbound 来区分流量。

在图 22 中，接入节点的 vSwitch 约定为 Access vSwitch，在两个虚拟承载伪线链路(也可能是物理链路)之间接入 vSwitch 后， Access vSwitch 实施了流量整形相关的工作，约定了流量进入核心虚拟网络的形式。至于量化实施细则和方法留待详细设计文档完成。

附录：

虚拟带宽接入