[**Neutron 理解 (3): Open vSwitch + GRE/VxLAN 组网 [Netruon Open vSwitch + GRE/VxLAN Virutal Network]**](http://www.cnblogs.com/sammyliu/p/4627230.html)

学习 Neutron 系列文章：

（1）[Neutron 所实现的虚拟化网络](http://www.cnblogs.com/sammyliu/p/4622563.html%20)

（2）[Neutron OpenvSwitch + VLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4626419.html%20)

（3）[Neutron OpenvSwitch + GRE/VxLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4627230.html%20)

（4）[Neutron OVS OpenFlow 流表 和 L2 Population](http://www.cnblogs.com/sammyliu/p/4633814.html%20)

（5）[Neutron DHCP Agent](http://www.cnblogs.com/sammyliu/p/4419195.html)

（6）[Neutron L3 Agent](http://www.cnblogs.com/sammyliu/p/4636091.html%20)

（7）[Neutron LBaas](http://www.cnblogs.com/sammyliu/p/4656176.html%20)

（8）[Neutron Security Group](http://www.cnblogs.com/sammyliu/p/4658746.html%20)

（9）[Neutron FWaas 和 Nova Security Group](http://www.cnblogs.com/sammyliu/p/4675991.html)

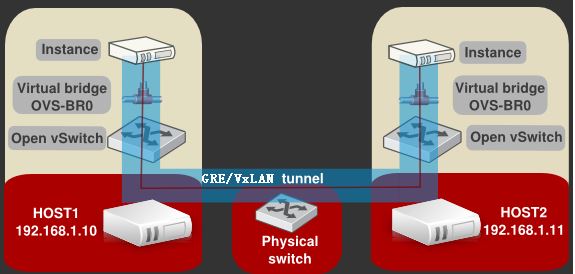
（10）[Neutron VPNaas](http://www.cnblogs.com/sammyliu/p/4677386.html%20)

（11）[Neutron DVR](http://www.cnblogs.com/sammyliu/p/4713562.html)

（12）[Neutron VRRP](http://www.cnblogs.com/sammyliu/p/4692081.html)

（13）[High Availability （HA）](http://www.cnblogs.com/sammyliu/p/4741967.html)

    目前，OpenStack Neutron 支持使用两种隧道网络技术 通用路由封装(GRE) 和 VxLAN 来实现虚拟的二层网络。这两种技术大致看起来非常相似，都是需要使用 OpenStack 在计算和网络节点上建立隧道来传输封装的虚机发出的数据帧：



在Neutron 中使用 GRE/VxLAN 时的配置也几乎完全相同。具体可以参考我已有的几篇文章：

* [探索 OpenStack 之（8）：Neutron 深入探索之 OVS + GRE 之 完整网络流程 篇](http://www.cnblogs.com/sammyliu/p/4204190.html)
* [探索 OpenStack 之（7）：Neutron 深入探索之 Open vSwitch (OVS) + GRE 之 Neutron节点篇](http://www.cnblogs.com/sammyliu/p/4201721.html)
* [学习OpenStack之（6）：Neutron 深入学习之 OVS + GRE 之 Compute node 篇](http://www.cnblogs.com/sammyliu/p/4201143.html)

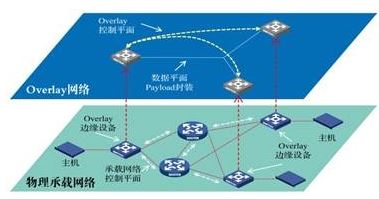
本文将不再重复说明这些配置细节。本文将试着分析两种技术本身的异同，以及Neutorn 中的代码过程。

**1. Overlay 网络**

**1.1 Overlay 技术概述**

    Overlay 在网络技术领域，指的是一种网络架构上叠加的虚拟化技术模式，其大体框架是对基础网络不进行大规模修改的条件下，实现应用在网络上的承载，并能与其它网络业务分离，并且以基于IP的基础网络技术为主。Overlay 技术是在现有的物理网络之上构建一个虚拟网络，上层应用只与虚拟网络相关。一个Overlay网络主要由三部分组成：

* 边缘设备：是指与虚拟机直接相连的设备
* 控制平面：主要负责虚拟隧道的建立维护以及主机可达性信息的通告
* 转发平面：承载 Overlay 报文的物理网络



    当前主流的 Overlay 技术主要有VXLAN, GRE/NVGRE和 STT。这三种二层 Overlay 技术，大体思路均是将以太网报文承载到某种隧道层面，差异性在于选择和构造隧道的不同，而底层均是 IP 转发。如下表所示为这三种技术关键特性的比较。其中VXLAN利用了现有通用的UDP传输，其成熟性高。总体比较，VLXAN技术相对具有优势。



**1. 1 GRE 技术**

     GRE（Generic Routing Encapsulation，通用路由协议封装）协议，是一种 IP-over-IP 的隧道，由 Cisco 和 Net-smiths 等公司于1994年提交给IETF，它对部分网络层协议（IP）的数据报进行封装，使这些被封装的数据报能够在 IPv4/IPv6 网络中传输。其协议格式见RFC2784（https://tools.ietf.org/html/rfc2784）。简单的说，GRE是一种协议封装的格式。它规定了如何用一种网络协议去封装另一种网络协议，很多网络设备都支持该协议。说得直白点，就是 GRE 将普通的包（如ip包）封装了，又按照普通的ip包的路由方式进行路由，相当于ip包外面再封装一层 GRE 包。在本例子中，在 GRE 包外围实际上又有一层公网的ip包。

   GRE 使用 tunnel（隧道）技术，数据报在 tunnel 的两端封装，并在这个通路上传输，到另外一端的时候解封装。你可以认为 tunnel 是一个虚拟的点对点的连接。（实际 Point To Point 连接之后，加上路由协议及nat技术，就可以把两个隔绝的局域网连接在一起，就实现了NetToNet的互联）。一般 GRE turnel 是在多个网络设备（一般为路由器）之间建立。因为 linux 服务具备路由转发功能，所以当您缺少专业的路由设备的时候，可使用 linux 服务器实现 gre turnel 的建立（实际上巨大多数网络设备都使用unix或linux系统）。

[这篇文章](http://h2ofly.blog.51cto.com/6834926/1544860) 很好地解释 GRE 的原理和一个应用场景。

**1.1.1 应用场景**

    办公网（LAN 192.168.1.0/24，并通过固定的公网IP上网）和 IDC （LAN 10.1.1.0/24，并有多个公网ip，两者通过公网ip互联）这两个局域网 (或者任意两个不同局域网) 相互隔离。但是在日常运维和研发过程中，需要在办公网访问IDC网络。如果都通过公网 IP 绕，既不方便，也不安全；拉专线是最稳定可靠的办法，但是成本高。

**1.1.2 使用 GRE 的配置**

如果公司有多余的固定的公网 ip 或者路由器本身支持GRE，建议使用 GRE 方案。

（1）办公网路由器（linux服务器实现）：局域网IP 192.168.1.254，公网 IP 180.1.1.1 配置。

cat /usr/local/admin/gre.sh #并把改脚本加入开机启动

[复制代码](javascript:void(0);)

#!/bin/bash

modprobe ip\_gre #加载gre模块

ip tunnel add office mode gre remote 110.2.2.2 local 180.1.1.1 ttl 255 #使用公网 IP 建立 tunnel 名字叫 ”office“ 的 device，使用gre mode。指定远端的ip是110.2.2.2，本地ip是180.1.1.1。这里为了提升安全性，你可以配置iptables，公网ip只接收来自110.2.2.2的包，其他的都drop掉。

ip link set office up #启动device office

ip link set office up mtu 1500 #设置 mtu 为1500

ip addr add 192.192.192.2/24 dev office #为 office 添加ip 192.192.192.2

echo 1 > /proc/sys/net/ipv4/ip\_forward #让服务器支持转发

ip route add 10.1.1.0/24 dev office #添加路由，含义是：到10.1.1.0/24的包，由office设备负责转发

iptables -t nat -A POSTROUTING -d 10.1.1.0/24 -j SNAT --to 192.192.192.2#否则 192.168.1.x 等机器访问 10.1.1.x网段不通

[复制代码](javascript:void(0);)

IDC路由器（linux服务器实现）：局域网 ip：10.1.1.1，公网 ip110.2.2.2配置  
cat /usr/local/admin/gre.sh#并把改脚本加入开机启动

[复制代码](javascript:void(0);)

#!/bin/bash

modprobe ip\_gre

ip tunnel add office mode gre remote 180.1.1.1 local 110.2.2.2 ttl 255

ip link set office up

ip link set office up mtu 1500

ip addr add 192.192.192.1/24 dev office #为office添加 ip 192.192.192.1

echo 1 > /proc/sys/net/ipv4/ip\_forward

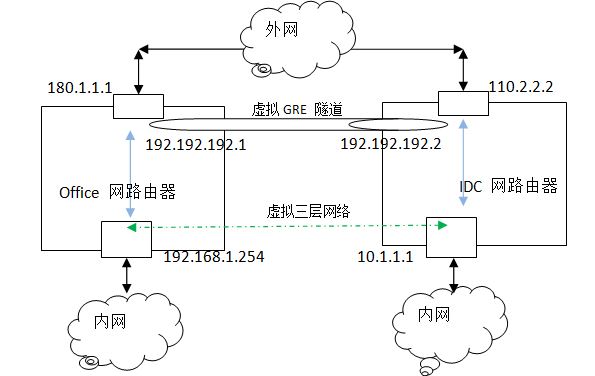
ip route add 192.168.1.0/24 dev office

iptables -t nat -A POSTROUTING -s 192.192.192.2 -d 10.1.0.0/16 -j SNAT --to 10.1.1.1 #否则192.168.1.X等机器访问10.1.1.x网段不通

iptables -A FORWARD -s 192.192.192.2 -m state --state NEW -m tcp -p tcp --dport 3306 -j DROP #禁止直接访问线上的3306，防止内网被破

[复制代码](javascript:void(0);)

配置示意图：



然后两个局域网中的机器就可以互通了。

**1.1.3 过程**

过程：

1. GRE turnel 的打通：这个过程就是双方建立turnel的过程。   
2. 局域网路由过程  
（1）主机 A 发送一个源为192.168.1.2，目的为 10.1.1.2 的包。  
（2）封装过程  
        1、根据内网路由，可能是你的默认路由网关将之路由至 192.168.1.254。  
        2、192.168.1.254 第一次封装包，增加 GRE 包头，说明包的目的地址 192.192.192.1 和源地址192.192.192.2。  
        3、192.168.1.254 第二次封装包，增加公网的包头（否则在公网上无法路由），说明包的目的地址 110.2.2.2 和源地址 180.1.1.1。  
        4、192.168.1.254 把所有到 10.1.1.0/24 的包，源地址都转换为从192.192.192.2 出（snat）。  
3. 公网路由过程：经过n个路由设备，该包最终路由到110.2.2.2。  
4. 拆包过程  
        1、B端的路由器检测到是到达自己的 IP，就开始拆包。  
        2、拆包之后发现有GRE协议，就进一步拆包。  
        3、拆包之后发现目的地不是自己的内网ip、发现自己本地做了snat，就将其源IP替换为10.1.1.1。  
5. 局域网路由  
        1、实际上从 10.1.1.1 出发的，到达目的地为 10.1.1.2 的包，无需路由，直接在局域网内广播。10.1.1.2 的机器确定是发送给自己的包，就接收。然后进一步处理了。

（注：以上文字皆摘自 <http://h2ofly.blog.51cto.com/6834926/1544860>）

**1.1.4 GRE 的不足**

GRE 技术本身还是存在一些不足之处：

（1）Tunnel 的数量问题

    GRE 是一种点对点（point to point）标准。Neutron 中，所有计算和网络节点之间都会建立 GRE Tunnel。当节点不多的时候，这种组网方法没什么问题。但是，当你在你的很大的数据中心中有 40000 个节点的时候，又会是怎样一种情形呢？使用标准 GRE的话，将会有 780 millions 个 tunnels。

（2）扩大的广播域

   GRE 不支持组播，因此一个网络（同一个 GRE Tunnel ID）中的一个虚机发出一个广播帧后，GRE 会将其广播到所有与该节点有隧道连接的节点。

（3）GRE 封装的IP包的过滤和负载均衡问题

    目前还是有很多的防火墙和三层网络设备无法解析 GRE Header，因此它们无法对 GRE 封装包做合适的过滤和负载均衡。

**1.2. VxLAN 技术**

    VxLAN 主要用于封装、转发2层报文。VXLAN 全称 Virtual eXtensible Local Area Network，简单的说就是扩充了的 VLAN，其使得多个通过三层连接的网络可以表现的和直接通过一台一台物理交换机连接配置而成的网络一样处在一个 LAN 中。

    它的实现机制是，将二层报文加上个 VxLAN header，封装在一个 UDP 包中进行传输。VxLAN header 会包括一个 24 位的 ID（称为VNI），含义类似于 VLAN id 或者 GRE 的 tunnel id。GRE 一般是通过路由器来进行 GRE 协议的封装和解封的，在 VXLAN 中这类封装和解封的组件有个专有的名字叫做 VTEP。相比起 VLAN 来说，好处在于其突破了VLAN只有 4000+ 子网的限制，同时架设在 UDP 协议上后其扩展性提高了不少（因为 UDP 是高层协议，屏蔽了底层的差异，换句话说屏蔽了二层的差异）。

**1.2.1 VxLAN 主要的网络设备与组网方案**

VXLAN网络设备主要有三种角色，分别是：

（1）VTEP(VXLAN Tunnel End Point)：直接与终端设备比如虚机连接的设备，负责原始以太报文的 VXLAN 封装和解封装，形态可以是虚拟交换机比如 Open vSwitch，也可以是物理交换机。

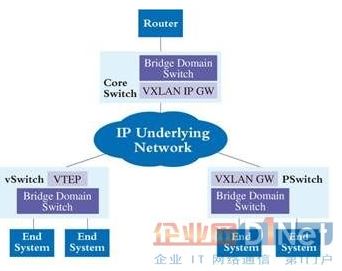
（2）VXLAN GW(VXLAN Gateway/二层网关)：用于终结VXLAN网络，将VXLAN报文转换成对应的传统二层网络送到传统以太网络，适用于VXLAN网络内服务器与远端终端或远端服务器的二层互联。如在不同网络中做虚拟机迁移时，当业务需要传统网络中服务器与VXLAN网络中服务器在同一个二层中，此时需要使用VXLAN二层网关打通VXLAN网络和二层网络。如下图所示，VXLAN 10网络中的服务器要和IP网络中VLAN100的业务二层互通，此时就需要通过VXLAN的二层网关进行互联。VXLAN10的报文进入IP网络的流量，剥掉VXLAN的报文头，根据VXLAN的标签查询对应的VLAN网络（此处对应的是VLAN100），并据此在二层报文中加入VLAN的802.1Q报文送入IP网络；相反VLAN100的业务流量进入VXLAN也需要根据VLAN获知对应的VXLAN网络编号，根据目的MAC获知远端VTEP的IP地址，基于以上信息进行VXLAN封装后送入对应的VXLAN网络。可见，它除了具备 VTEP 的功能外，还负责 VLAN 报文与 VXLAN 报文之间的映射和转发，主要以物理交换机为主。



（3）VXLAN IP GW(VXLAN IP Gateway/三层网关)：用于终结 VXLAN 网络，将 VXLAN 报文转换成传统三层报文送至 IP 网络，适用于 VXLAN 网络内服务器与远端终端之间的三层互访；同时也用作不同VXLAN网络互通。如下图所示，当服务器访问外部网络时，VXLAN三层网关剥离对应VXLAN报文封装，送入IP网络；当外部终端访问VXLAN内的服务器时，VXLAN根据目的IP地址确定所属VXLAN及所属的VTEP，加上对应的VXLAN报文头封装进入VXLAN网络。VXLAN之间的互访流量与此类似，VXLAN网关剥离VXLAN报文头，并基于目的IP地址确定所属VXLAN及所属的VTEP，重新封装后送入另外的VXLAN网络。可见，具有 VXLAN GW 的所有功能，此外，还负责处理不同 VXLAN 之间的报文通信，同时也是数据中心内部服务向往发布业务的出口，主要以高性能物理交换机为主。



可见，无论是二层还是三层网关，均涉及到查表转发、VXLAN报文的解封装和封装操作。从转发效率和执行性能来看，都只能在物理网络设备上实现，并且传统设备无法支持，必须通过新的硬件形式来实现。以上设备均是物理网络的边缘设备，而有三种边缘设备构成了VXLAN Overlay 网络，对于应用系统来说，只与这三种设备相关，而与底层物理网络无关。



Overlay 网络架构就纯大二层的实现来说，可分为 网络Overlay、主机Overlay以及两种方式同时部署的混合Overlay。 Overlay网络与外部网络数据连通也有多种实现模式，并且对于关键网络部件有不同的技术要求。

**（1）网络 Overlay 方案：使用物理交换机做VxLAN网络设备**

网络Overlay

    网络Overlay 方案如上图所示，所有的物理接入交换机支持VXLAN，物理服务器支持SR-IOV功能，使虚拟机通过SR-IOV技术直接与物理交换机相连，虚拟机的流量在接入交换机上进行VXLAN报文的封装和卸载，对于非虚拟化服务器，直接连接支持VXLAN的接入交换机，服务器流量在接入交换机上进行VXLAN报文封装和卸载;当VXLAN网络需要与VLAN网络通信时，采用物理交换机做VXLAN GW，实现VXLAN网络主机与VLAN网络主机的通信;采用高端交换机做VXLAN IP GW，实现VXLAN网络与WAN以及Internet的互连。

**（2）主机Overlay方案：使用服务器上的软件实现VxLAN网络设备**

主机Overlay

  在主机Overlay方案中(如上图所示)，VTEP、VXLAN GW、VXLAN IP GW 均通过安装在服务器上的软件实现，

* vSwitch 实现VTEP功能，完成VXLAN报文的封装解封装;
* vFW 等实现VXLAN GW功能，实现VXLAN网络与VLAN网络、物理服务器的互通;
* vRouter作为VXLAN IP GW，实现 VXLAN 网络与 Internet和WAN的互联。

在本组网中，由于所有VXLAN报文的封装卸载都通过软件实现，会占用部分服务器资源，当访问量大时，vRouter会成为系统瓶颈。

**（3）混合Overlay组网方案**

混合Overlay

    上述两种组网方案中，网络Overlay方案与虚拟机相连，需要通过一些特殊的要求或技术实现虚拟机与VTEP的对接，组网不够灵活，但是主机Overlay方案与传统网络互通时，连接也比较复杂，且通过软件实现 VXLAN IP GW 也会成为整个网络的瓶颈，所以最理想的组网方案应该是一个结合了网络Overlay与主机Overlay两种方案优势的混合Overlay方案。

    如上图所示，它通过 vSwitch 实现虚拟机的VTEP，通过物理交换机实现物理服务器的VTEP，通过物理交换机实现VXALN GW和VXLAN IP GW；混合式Overlay组网方案对虚拟机和物理服务器都能够很好的兼容，同时通过专业的硬件交换机实现VXLAN IP GW从而承载超大规模的流量转发，是目前应用比较广泛的组网方案。

VxLAN 网络中，虚机之间的三种互访形式：

1. 相同VXLAN内 VM之间互访：
   1. 单播报文在 VTEP 处查找目的MAC地址，确定对应的VTEP主机IP地址
   2. 根据目的和源VTEP主机IP地址封装VXLAN报文头后发送给IP核心网
   3. IP 核心内部根据路由转发该UDP报文给目的VTEP
   4. 目的VTEP解封装VXLAN报文头后按照目的MAC转发报文给目的VM
2. 不同VXLAN内 VM之间需要互访经过VXLAN IP GW 完成，
   * 1. 在VXLAN IP GW上匹配 VXLAN Maping 表项进行转发，报文封装模式同同一VXLAN内VM一致
3. VXLAN VM与VLAN VM之间互访，通过VXLAN GW来完成，
   1. VXLAN 报文先通过VXLAN 内部转发模式对报文进行封装，目的IP为VXLAN GW，
   2. 在VXLAN GW把VXLAN报文解封装后，匹配二层转发表项进行转发，VLAN到VXLAN的访问流程正好相反

**1.2.2 VXLAN 的实现**

**1.2.2.1 VxLAN 将二层数据帧封装为 UDP 包**

含义：

* Outer MAC destination address (MAC address of the tunnel endpoint VTEP)
* Outer MAC source address (MAC address of the tunnel source VTEP)
* Outer IP destination address (IP address of the tunnel endpoint VTEP)
* Outer IP source address (IP address of the tunnel source VTEP)
* Outer UDP header：Src port 往往用于 load balancing，下文有提到；Dst port 即 VXLAN Port，默认值为 4789.
* VNID：表示该帧的来源虚机所在的 VXLAN 网段的 ID

特点：

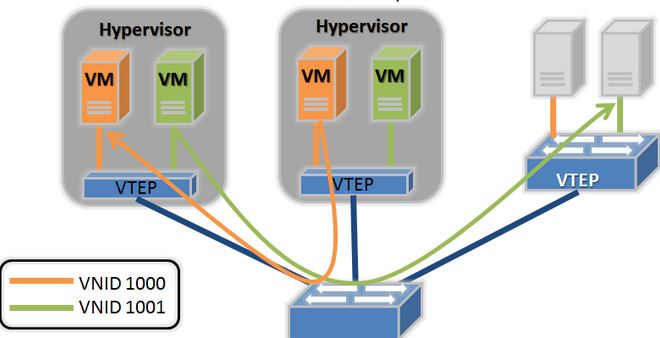
* VNID： 24-bits，最大 16777216。每个不同的 24-bits VNI 代表一个 VXLAN 网段。只有同一个网段中的虚机才能互相通信。
* VXLAN Port：目的 UDP 端口，默认使用 4789 端口。用户可以自己配置。
* 两个 VTEP 之间的 VXLAN tunnels 是无状态的。
* VTEP 可以在虚拟交换机上，物理交换机或者物理服务器上通过软件或者硬件实现。
* 使用多播来传送未知目的的、广播或者多播帧。
* VTEP 不可以对 VXLAN 包分段。

**1.2.2.2 VTEP 寻址**

    一个 VTEP 可以管理多个 VXLAN 隧道，每个隧道通向不同的目标。那 VTEP 接收到一个二层帧后，它怎么根据二层帧中的目的 MAC 地址找到对应的 VXLAN 隧道呢？VXLAN 利用了多播和 MAC 地址学习技术。如果它收到的帧是个广播帧比如 ARP 帧，也会经过同样的过程。

    以下图为例，每个 VTEP 包含两个 VXLAN 隧道。VTEP-1 收到二层 ARP 帧1（A 要查找 B 的 MAC） 后，发出一个 Dst IP 地址为VTEP多播组 239.1.1.1 的 VXLAN 封装 UDP 包。该包会达到 VTEP-2 和 VTEP-3。VTEP-3 收到后，因为目的 IP 地址不在它的范围内，丢弃该包，但是学习到了一条路径：MAC-A,VNI 10,VTEP-1-IP，它知道要到达 A 需要经过 VTEP-1 了。VTEP-2 收到后，发现目的 IP 地址是机器 B，交给 B，同时添加学习到的规则 MAC-A,VNI 10,VTEP-1-IP。B 发回响应帧后，VTEP-2 直接使用 VTEP-1 的 IP 直接将它封装成三层包，通过物理网络直接到达 VTEP-1，再由它交给 A。VTEP-1 也学习到了一条规则 MAC-B,VNI 10,VTEP-2-IP。

**1.2.2.3 VxLAN 组网**



* 逻辑 VxLAN Tunnel：建立在物理的 VxLAN 网络之上，向虚机提供虚拟的二层网络，以 VNID 做区分。
* VTEP （VxLAN Tunnel End Point）：对虚机的二层包封装和解封。

**1.2.2.4 数据流向**

发送端：

1. 计算目的地址：Linux 内核在发送之前会检查数据帧的目的MAC地址，需要选择目的 VTEP。
   1. 如果是广播或者多播地址，则使用其 VNI 对应的 VXLAN group 组播地址，该多播组内所有的 VTEP 将收到该多播包；
   2. 如果是单播地址，如果 Linux 的 MAC 表中包含该 MAC 地址对应的目的 VTEP 地址，则使用它；
   3. 如果是单播地址，但是 LInux 的 MAC 表中不包含该 MAC 地址对应的目的 VTEP IP，那么使用该 VNI 对应的组播地址。
2. 添加Headers：依次添加 VXLAN header，UDP header，IP header。

接收端：

1. UDP监听：因为 VXLAN 利用了 UDP，所以它在接收的时候势必须要有一个 UDP server 在监听某个端口，这个是在 VXLAN 初始化的时候完成的。
2. IP包剥离：一层一层剥离出原始的数据帧，交给 TCP/IP 栈，由它交给虚机。

实现代码在 [这里](http://openvswitch.org/pipermail/dev/2011-October/012051.html)。

**1.2.2.6 负载均衡**

    组成 VXLAN 隧道的三层路由器在有多条 ECMP （Equal-cost multi-path routing）路径通往目的 VTEP 时往往会使用基于每个包（ per-package） 的 负载均衡（load balancing）。因为两个 VTEP 之间的所有数据包具有相同的 outer source and destination IP addresses 和 UDP destination port，因此 ECMP 只能使用 soure UDP port。VTEP 往往将其设置为原始数据帧中的一些参数的 hash 值，这样 ECMP 就可以使用该 hash 值来区分 tunnels 直接的网络流量了。

[这里](http://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-729383.html)有更详细的 VXLAN 介绍。

**1.3. 相关技术对比**

**1.3.1 VLAN 和 VxLAN 的对比**

该图中的网络组件和Neutorn 中相应的组件一一对应：

|  |  |
| --- | --- |
| VLAN | VXLAN |
| cloudVirBrX 对应于 br-int  ethY.X 对应于 br-eth3 （物理bridge）  ethY 对应于 Hypervisor NIC | cloudVirBrX 对应于 br-int  vxlanX 对应于 Neutron 中的 br-tun。  cloudbrZ 对应于 Neutorn 中的 br-tun 上的 vxlan interface。  ethY 对应于Hypervisor NIC |

**1.3.2 GRE 和 VXLAN 对比**

| **Feature** | **VXLAN** | **GRE** |
| --- | --- | --- |
| Segmentation | 24-bit VNI (VXLAN Network Identifier) | Uses different Tunnel IDs |
| Theoretical scale limit | 16 million unique IDs | 16 million unique IDs |
| Transport | UDP (default port 4789) | IP Protocol 47 |
| Filtering | VXLAN uses UDP with a well-known destination port; firewalls and switch/router ACLs can be tailored to block only VXLAN traffic. | Firewalls and layer 3 switches and routers with ACLs will typically not parse deeply enough into the GRE header to distinguish tunnel traffic types; all GRE would need to be blocked indiscriminately. |
| Protocol overhead | 50 bytes over IPv4 (8 bytes VXLAN header, 20 bytes IPv4 header, 8 bytes UDP header, 14 bytes Ethernet). | 42 bytes over IPv4 (8 bytes GRE header, 20 bytes IPv4 header, 14 bytes Ethernet). |
| Handling unknown destination packets, broadcasts, and multicast | VXLAN uses IP multicast to manage flooding from these traffic types. Ideally, one logical Layer 2 network (VNI) is associated with one multicast group address on the physical network. This requires end-to-end IP multicast support on the physical data center network. | GRE has no built-in protocol mechanism to address these. This type of traffic is simply replicated to all nodes. |
| IETF specification | http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-01 | http://tools.ietf.org/html/rfc2784.html |

**2. Neutron 通过 OVS 对 GRE 和 VXLAN 的支持**

    因为 OVS 对两种协议的实现机制，Neutron 对两个协议的支持的代码和配置几乎是完全一致的，除了一些细小差别，比如名称，以及个别配置比如VXLAN的UDP端口等。OVS 在计算或者网络节点上的 br-tun 上建立多个 tunnel port，和别的节点上的 tunnel port 之间建立点对点的 GRE/VXLAN Tunnel。Neutron GRE/VXLAN network 使用 segmentation\_id （VNI 或者 GRE tunnel id） 作为其网络标识，类似于 VLAN ID，来实现不同网络内网络流量之间的隔离。

**2.1 Open vSwitch 实现的 VxLAN VTEP**

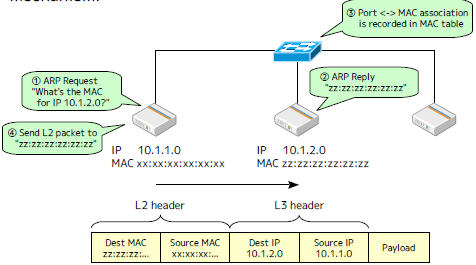
  从上面的基础知识部分，我们知道 VTEP 不只是实现包的封装和解包，还包括：

（1）ARP 解析：需要尽量高效的方式响应本地虚机的 ARP 请求

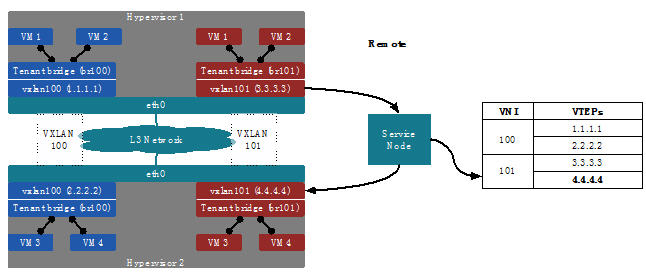
（2）目的 VTEP 地址搜索：根据目的虚机的 MAC 地址，找到它所在机器的 VTEP 的 IP 地址

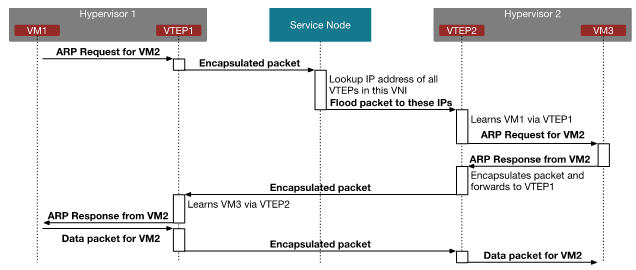
通常的实现方式包括：

（1）使用 L3 多播



（2）使用 SDN 控制器（controller）来提供集中式的 MAC/IP 对照表





（一个基于 Linuxbrige + VxLAN + Service Node 的集中式 Controller node 解决 VNI-VTEP\_IPs 映射的提议，替代L3多播和广播，来源： 20140520-dlapsley-openstack-summit-vancouver-vxlan-v0-150520174345-lva1-app6891.pptx）

（3）在VTEP本地运行一个代理（agent），接收（MAC, IP, VTEP IP）数据，并提供给 VTEP

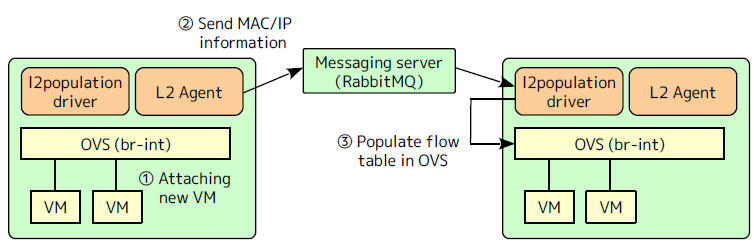
那 Open vSwitch 是如何实现这些功能需求的呢？

（1）在没有启用 l2population 的情况下，配置了多播就使用多播，没的话就使用广播

（2）在启用 l2population 的情况下，在虚机 boot 以后，通过 MQ 向用于同网络虚机的节点上的 l2population driver 发送两种数据，再将数据加入到 OVS 流表

    （2.1）FDB (forwarding database): 目的地址-所在 VTEP IP 地址的对照表，用于查找目的虚机所在的 VTEP 的 IP 地址

     （2.2）虚机 IP 地址 - MAC 地址的对照表，用于响应本地虚机的 ARP 请求



详细情况，请参考 [Neutron 理解 (4): Neutron OVS OpenFlow 流表 和 L2 Population [Netruon OVS OpenFlow tables + L2 Population]](http://www.cnblogs.com/sammyliu/default.html?page=2)

**2.2 隧道端口 （tunnel port）**

    下面是一个实例。该例子中，10.0.1.31 Hypervisor上的 br-tun 上分别有两个 GRE tunnel 端口和两个 VXLAN tunnel 端口，分别连接 目标 Hypervisor 10.0.1.39 和 21。

[复制代码](javascript:void(0);)

Bridge br-tun

Port "vxlan-0a000127"

Interface "vxlan-0a000127"

type: vxlan

options: {df\_default="true", in\_key=flow, local\_ip="10.0.1.31", out\_key=flow, remote\_ip="10.0.1.39"}

Port "vxlan-0a000115"

Interface "vxlan-0a000115"

type: vxlan

options: {df\_default="true", in\_key=flow, local\_ip="10.0.1.31", out\_key=flow, remote\_ip="10.0.1.21"}

Port "gre-0a000127"

Interface "gre-0a000127"

type: gre

options: {df\_default="true", in\_key=flow, local\_ip="10.0.1.31", out\_key=flow, remote\_ip="10.0.1.39"}

Port "gre-0a000115"

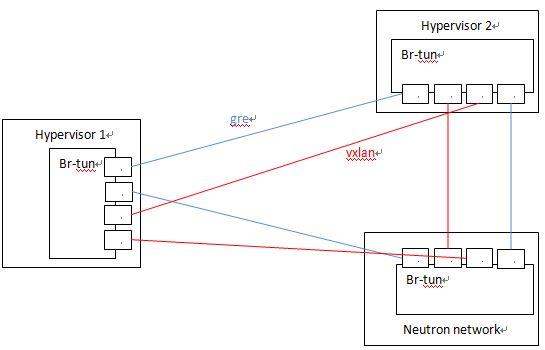
Interface "gre-0a000115"

type: gre

options: {df\_default="true", in\_key=flow, local\_ip="10.0.1.31", out\_key=flow, remote\_ip="10.0.1.21"}

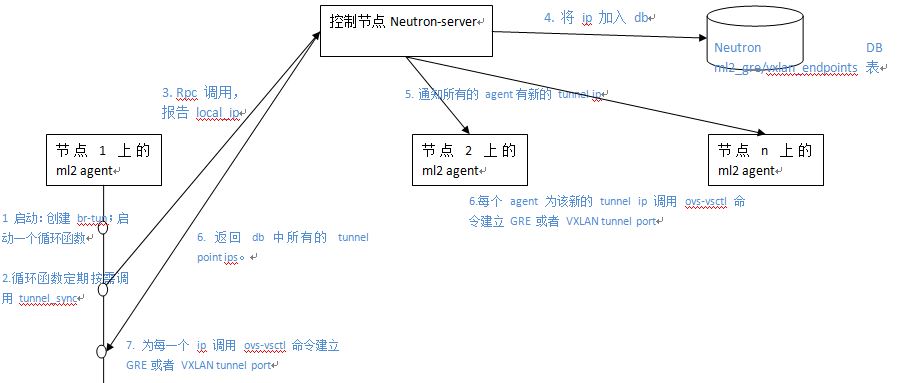
[复制代码](javascript:void(0);)

当有多个节点时，Neutron 会建立一个 tunnel 网：



**2.3 在不使用 l2population 时的隧道建立过程**

    要使用 GRE 和 VXLAN，管理员需要在 ml2 配置文件中配置 local\_ip（比如该物理服务器的公网 IP），并使用配置项 tunnel\_types 指定要使用的隧道类型，即 GRE 或者 VXLAN。当 enable\_tunneling = true 时，Neutorn ML2 Agent 在启动时会建立 tunnel bridge，默认为 br-tun。接着，ML2 Agent 会在 br-tun 上建立 tunnel ports，作为 GRE/VXLAN tunnel 的一端。具体过程如图所示：



OVS 默认使用 4789 作为 VXLAN port。下表中可以看出 Neutron 节点在该端口上监听来自所有源的UDP包：

root@compute1:/home/s1# netstat -lnup

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

udp 3840 0 0.0.0.0:4789 0.0.0.0:\* -

**2.4 Neutron tunnel 数据流向**

    Neutron 中的数据流向是受 Neutron 添加在 integration bridge 和 tunnel bridge 中的 OpenFlow rules 控制的，而且和 L2 population 直接相关。具体内容在下一篇文章中会仔细分析。

**2.5 MTU 问题（**[**来源**](http://notelinux.org/openstack-gre-vxlan-mtu-%E7%BD%91%E9%80%9F%E9%97%AE%E9%A2%98/)**）**

 VXLAN 模式下虚拟机中的 mtu 最大值为1450，也就是只能小于1450，大于这个值会导致 openvswitch 传输分片，进而导致虚拟机中数据包数据重传，从而导致网络性能下降。GRE 模式下虚拟机 mtu 最大为1462。

计算方法如下：

* vxlan mtu = 1450 = 1500 – 20(ip头) – 8(udp头) – 8(vxlan头) – 14(以太网头)
* gre mtu = 1458 = 1500 – 20(ip头) – 8(gre头) – 14(以太网头)

可以配置 Neutron DHCP 组件，让虚拟机自动配置 mtu，官方文档链接 <http://docs.openstack.org/juno/install-guide/install/yum/content/neutron-network-node.html>

#/etc/neutron/dhcp\_agent.ini

[DEFAULT]

dnsmasq\_config\_file = /etc/neutron/dnsmasq-neutron.conf

#/etc/neutron/dnsmasq-neutron.conf

dhcp-option-force=26,1450或1458

重启 DHCP Agent，让虚拟机重新获取 IP，然后使用 ifconfig 查看是否正确配置 mtu。

**3. H3C 选择 VxLAN 的理由**

   H3C 是国内领先的网络设备供应商之一，在其 [一篇文章](http://www.h3c.com.cn/About_H3C/Company_Publication/IP_Lh/2014/07/Home/Catalog/201501/852548_30008_0.htm) 中，谈到了他们为什么选择 VxLAN 技术。这对别的用户具有一定的参考性。

**3.1 为什么选择 VxLAN**

从Overlay网络出现开始，业界陆续定义了多种实现Overlay网络的技术，主流技术包括：VXLAN、NVGRE、STT、Dove等（如图3所示）。

Overlay主流技术概览（没找到清晰图。。）

从标准化程度进行分析，DOVE和STT到目前为止，标准化进展缓慢，基本上可以看作是IBM和VMware的私有协议。因此，从H3C的角度来看无法选择这两种技术。

从技术的实用性来看，XLAN和NVGRE两种技术基本相当。其主要的差别在于链路Hash能力。由于NVGRE采用了GRE的封装报头，需要在标准GRE报头中修改部分字节来进行Hash实现链路负载分担。这就需要对物理网络上的设备进行升级改造，以支持基于GRE的负载分担。这种改造大部分客户很难接受。相对而言，VXLAN技术是基于UDP报头的封装，传统网络设备可以根据UDP的源端口号进行Hash实现链路负载分担。这样VXLAN网络的部署就对物理网络没有特殊要求。这是最符合客户要求的部署方案，所以VXLAN技术是目前业界主流的实现方式。

**3.2 VXLAN为什么选择SDN**

    VXLAN的标准协议目前只定义了转发平面流程，对于控制平面目前还没有协议规范，所以目前业界有三种定义VXLAN控制平面的方式。

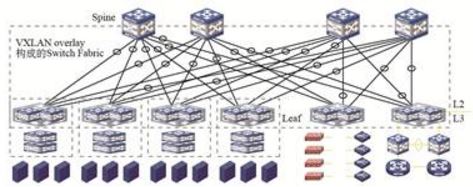
**方式1：组播。**由物理网络的组播协议形成组播表项，通过手工将不同的VXLAN与组播组一一绑定。VXLAN的报文通过绑定的组播组在组播对应的范围内进行泛洪。简单来说，和VLAN方式的组播泛洪和MAC地址自学习基本一致。区别只是前者在三层网络中预定义的组播范围内泛洪，而后者是在二层网络中指定VLAN范围内泛洪。这种方式的优点是非常简单，不需要做协议扩展。但缺点也是显而易见的，需要大量的三层组播表项，需要复杂的组播协议控制。显然，这两者对于传统物理网络的交换机而言，都是巨大的负荷和挑战，基本很难实现。同时，这种方式还给网络带来大量的组播泛洪流量，对网络性能有很大的影响。

**方式2：自定义协议。**通过自定义的邻居发现协议学习Overlay网络的拓扑结构并建立隧道管理机制。通过自定义（或扩展）的路由协议透传Overlay网络的MAC地址（或IP地址）。通过这些自定义的协议可以实现VXLAN控制平面转发表项的学习机制。这种方式的优点是不依赖组播，不存在大量的组播泛洪报文，对网络性能影响很小。缺点是通过邻居发现协议和路由协议控制所有网络节点，这样网络节点的数量就受到协议的限制。换句话说，如果网络节点的数量超过一定范围，就会导致对应的协议（例如路由协议）运行出现异常。这一点在互联网行业更加明显，因为互联网行业云计算的基本特征就是大规模甚至超大规模。尤其是在vSwitch上运行VXLAN自定义路由协议，其网络节点数量可以达到几千甚至上万个，没有路由协议可以支持这种规模的网络。

**方式3：SDN控制器。**通过SDN控制器集中控制VXLAN的转发，经由Openflow协议下发表项是目前业界的主流方式。这种方式的优点是不依赖组播，不对网络造成负荷；另外，控制器通过集群技术可以实现动态的扩容，所以可以支持大规模甚至超大规模的VXLAN网络。当然，SDN控制器本身的性能和可靠性决定了全网的性能和可靠性，所以如何能够提高控制器的性能和可靠性就是核心要素。

**3.3 VXLAN Fabric网络架构的优势**

    云计算网络一直都有一个理念：Network as a Fabric，即整网可以看作是一个交换机。通过VXLAN Overlay可以很好地实现这一理念——VXLAN Fabric（如下图所示）。



VXLAN Fabric网络架构

Spine和Leaf节点共同构建了Fabric的两层网络架构，通过VXLAN实现Spine和Leaf之间的互联，可以看做是交换机的背板交换链路。Spine节点数量可以扩容，最大可以达到16台。Leaf节点数量也可以平滑扩容。理论上只要Spine节点的端口密度足够高，这个Fabric可以接入数万台物理服务器。

另外，通过VXLAN隧道可以实现安全服务器节点的灵活接入。这些安全服务节点可以集中部署在一个指定区域，也可以灵活部署在任意Leaf节点下。通过Service Chain技术实现任意两个虚拟机之间可以通过任意安全服务节点互联。保证网络中虚拟机业务的安全隔离和控制访问。同理，Fabric的出口节点也可以部署在任意位置，可以灵活扩展。

简而言之，VXLAN Fabric构建了一个灵活的、稳定的、可扩展的Overlay网络。这个网络可以有效地解决云计算对网络的挑战，是云计算网络发展的趋势。

参考链接：

1. [GRE和VXLAN](http://bingotree.cn/?p=654)

2. <https://plus.google.com/+KennethDuda/posts/2tnVCHkeVyZ>

3. [百家：云计算网络，没那么简单](http://www.h3c.com.cn/About_H3C/Company_Publication/IP_Lh/2014/07/Home/Catalog/201501/852548_30008_0.htm)

4. [全融合网络虚拟化技术](http://www.d1net.com/virtual/news/375158.html)

5. [Overlay网络与物理网络的关系](http://www.h3c.com.cn/About_H3C/Company_Publication/IP_Lh/2014/07/Home/Catalog/201501/852551_30008_0.htm)

本文中大部分图片来自互联网。