ZStack 技术白皮书精选

网络篇

扫一扫二维码，获取更多技术干货吧

版权声明

摘要

大道至简·极速部署，ZStack 致力于产品化私有云和混合云。

ZStack 是新一代创新开源的云计算 IaaS 软件，由英特尔、微软、CloudStack 等世界上最早一批虚拟化工程师创建，拥有 KVM、Xen、Hyper-V 等成熟的技术背景。

ZStack 创新提出了云计算 4S 理念，即 Simple （简单）、Strong （健壮）、Smart （智能）、Scalable （弹性），通过全异步架构，无状态服务架构，无锁架构等核心技术，完美解决云计算执行效率低，系统不稳定，不能支撑高并发等问题，实现 HA 和轻量化管理。

ZStack 发起并维护着国内最大的自主开源 IaaS 社区——zstack.io，吸引了 6000 多名社区用户，对外公开的 API 超过 1000 个。基于这 1000 多个 API，用户可以自由组装出自己的私有云、混合云，甚至利用 ZStack 搭建公有云对外提供服务。

ZStack 拥有充足的知识产权储备，积极申报多项软著和专利，参与业内标准、白皮书的撰写，入选云计算行业方案目录，还通过了工信部云服务能力认证和信通院可信云认证。ZStack 面向企业用户提供基于 IaaS 的私有云和混合云，是业内唯一一家实现产品化，并领先业内首家推出同时打通数据面和控制面无缝混合云的云服务商。选择 ZStack，用户可以官网直接下载、1 台 PC 也可上云、30 分钟完成从裸机的安装部署。

目前已有 1000 多家企业用户选择了 ZStack 云平台。

目录

ZSTACK--网络模型 1：L2 和 L3 网络

Stack 将网络模型抽象为 L2 和 L3 网络。L2 网络提供一种二层网络隔离的方式，而 L3 网络主要和 OSI 七层模型中第 4 层~第 7 层网络服务相对应。我们的想法是使用管理员熟悉的术语和概念，来形容 ZStack 的网络模型，使得管理员可以方便快捷的创建网络拓扑。

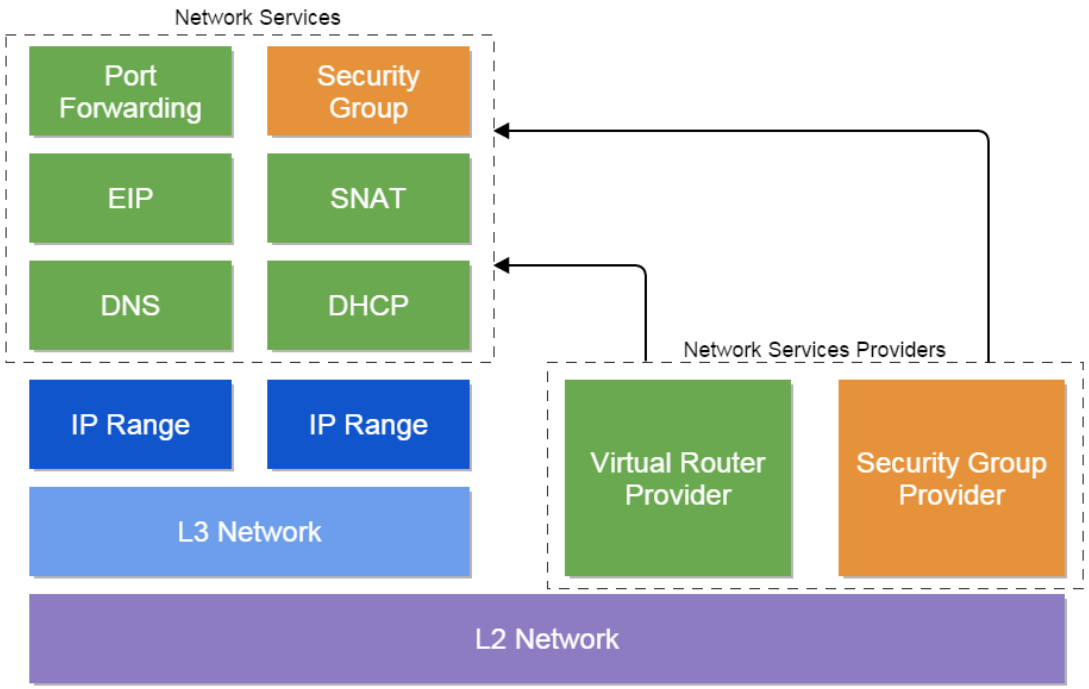
注：我们将不涉及任何在 Hypervisor 端虚拟化技术的网络实现细节；例如，我们将不讨论 ZStack 如何在 Linux 操作系统中创造网桥或 VLAN 设备。这篇文章的目的是给你介绍 ZStack 网络模型的简要构想。如果你还没有阅读“通用插件系统”的话，我们强烈建议你去阅读一遍，因为许多和插件相关的术语将在下文被提到。

概述

云计算中最令人兴奋和最困难的部分应该是网络模型。云技术给传统的数据中心带来的最大的变革是，管理员不需要花费几天甚至几周的时间去创建或改变网络的拓扑结构，相反，他们可以几分钟就能完成以前很艰巨的任务，通过点击在 IaaS 软件用户界面上的一些按钮。

为了达到这种简单性，IaaS 软件必须有一个清晰、灵活的网络模型，可以帮助管理员在云中建立大多数的，传统数据中心里的典型的网络拓扑。而且，更重要的是，它必须允许管理员改变已经构建好的网络，在任何必要的时候，而无需重新部署整个云。

ZStack 的网络模型的整体画面就像：



一个 L2 网络，精确地表示了一个二层网络广播域的，是所有网络元素的基础。在 L2 网络之上，有各种 L3 网络和网络服务提供模块；一个 L3 网络是一个与网络服务相关的子网；尽管一个 L2 网络通常只包含一个 L3 网络，只要 L3 网络的 IP 段不冲突，多个 L3 网络可以并存于同一 L2 网络。一个 L3 网络可能有一个或多个属于同一子网的 IP 段，IP 地址分段的目的是为了让用户保留一部分来自子网的 IP。网络服务，类似于 DHCP、DNS，由绑定到一个 L2 网络上的提供器提供给 L3 网络。

> ***注：****由于虚拟私有云（VPC）尚未在这个 ZStack 版本（0.6）支持，上述网络模型不显示 VPC 将如何工作。然而，概念是类似的，VPC 只是一个为多个 L3 网络设计的，有编程选路功能的调度器。我们将在未来的 ZStack 版本中引入 VPC，不久之后。*

L2 网络

一个 L2 网络负责提供一种二层隔离方法，可以是一个纯粹的 L2 技术（如 VLAN），或一个网络覆层（overlay）技术（如 GRE 隧道，VxLAN）。ZStack 不关心 L2 网络在后端使用的技术细节，所以包含必要的 L2 信息的数据结构--L2NetworkInventory--是高度抽象的：

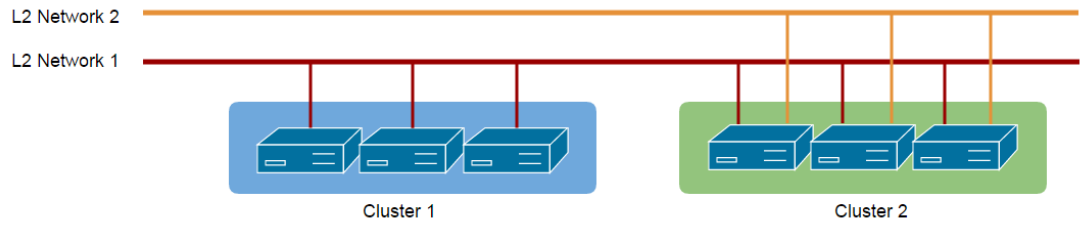
FIELD	DESCRIPTION
uuid	L2 network UUID
name	a short name
description	a long description
zoneUuid	uuid of zone the L2 network belongs to
physicalInterface	a string containing information necessary to implement the L2 network at the backend. for example,
type	L2 network type
attachedClusterUuids	a list of cluster uuid the L2 network has attached to

L2 网络的子类型可能有额外的属性，例如，L2VlanNetwork 有一个额外的字段的 vlan。

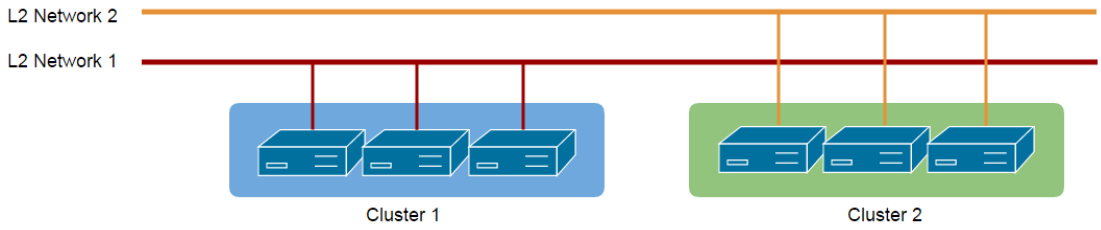
绑定策略

在真实的数据中心中，L2 网络通常代表主机之间一个的物理网络连接。例如，在同一 L2 交换机下的主机可能在同一个 L2 网络中。网络的连接不是一成不变的，它可能会在任何数据中心的物理设备改变的时候改变，例如管理员重新配置（re-wire）一个 L2 交换

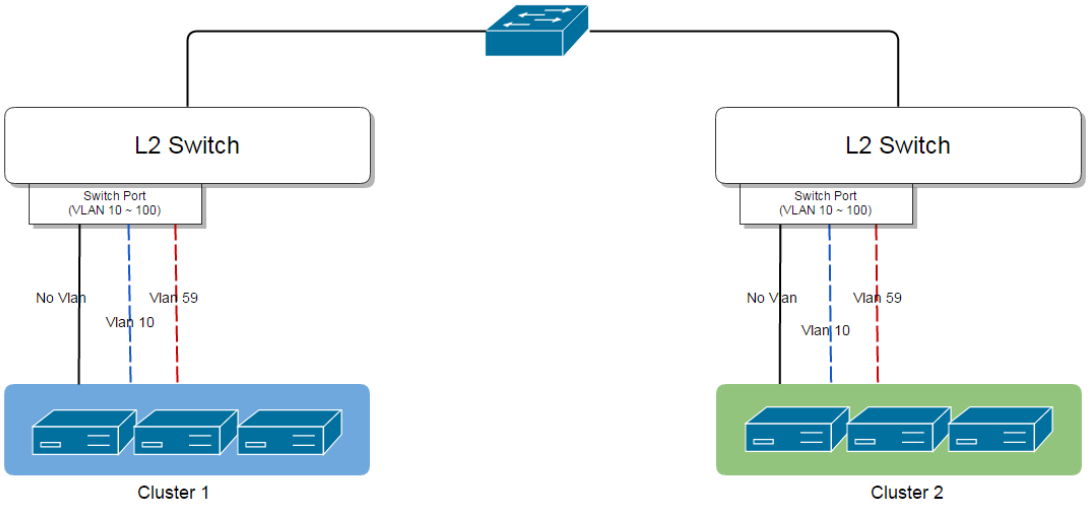
机。为了提供一种灵活的、描述主机和 L2 网络之间的关系的方式，ZStack 采用了一种所谓的绑定策略，允许一个 L2 网络连接从多个集群（主机的集合）中绑定/解绑。



上图，Cluster1 和 Cluster2 中的主机都是挂载在 L2 Network1 上，同时 Cluster2 上的主机也挂载在 L2 Network2 上，管理员可以同时将 L2 Network1 绑定两个集群，却不能仅仅把 L2 Network2 绑定在 Cluster2 上。一段时间后，如果管理员为了删除 L2 Network1 上的连接，重新配置在 Cluster2 上的主机，他们可以从 Cluster2 中解绑 L2 Network1 去反映当前的网络连接。



集群和 L2 网络之间的挂载关系，展示了在这些集群内的主机之间建立 L2 广播域的行为，这并不总是涉及到物理连接的变化。例如，连接到标记的交换机端口的主机，可以在以太网设备上使用操作系统中相同的 VLAN 创建网桥，用来为连接到这些网桥的虚拟机建立一个 L2 广播域；在这种情况下，绑定或解绑 L2 网络并不意味着任何物理基础设施的变化，但意味着创建或删除一个 L2 广播域的行为。



上图所示，一旦管理员创建一个包含 VLAN 10 的 L2VlanNetwork，并把它挂载到 cluster1 和 cluster2 上，一个广播域在这些集群中的主机之间被创建。虚拟机管理程序可以通过各种方式来实现 L2 广播域，例如，KVM 主机可以在它们的 Linux 操作系统上通过 VLAN 设备（VLAN 10）创建网桥；如果 L2VlanNetwork 解绑集群 cluster2 后，被解绑的集群中的主机将通过删除它们的 VLAN(10)网桥的方式，从广播域中被移除。这种创建/销毁广播域的概念适用于所有 L2 网络类型；例如，绑定一个 OvsGreL2Network 到 KVM 集群上可能导致 GRE 隧道在这些主机中被创建，而将一个 OvsGreL2Network 解绑可能导致 GRE 隧道被删除。

这种绑定策略有一个额外的好处是，考虑到了限制虚拟机可以运行的主机。因为虚拟机总是和 L3 网络一起被创建，这些 L3 网络属于一些 L2 网络，虚拟机将只被分配给已经绑定这些 L2 网络的集群中的主机。通过这种方式，管理员可以通过 L2 网络把主机划分到不同的池中，例如，一个连接了高带宽的 L2 网络的集群，一个连接了公有 L2 网络的集群。如果管理员想把所有的主机都放在一个单一的池中，他们可以让所有的 L2 网络绑定所有的集群。

后端实现

通过虚拟化技术，L2 网络的后端实现是高度依赖 Hypervisor 的。例如，在 KVM 主机上实现 L2VlanNetwork 就是创建一个 VLAN 设备的网桥，但对于 VMWare ESXi 主机则是配置 vSwitch。为了让 L2 网络的实现和 Hypervisor 解耦，ZStack 将实现某种类型 L2 网络的责

任委托给 Hypervisor 插件。为了实现一个 L2 网络，定义了两个扩展点。第一个是 L2NetworkRealizationExtensionPoint：

```
public interface L2NetworkRealizationExtensionPoint {

    void realize(L2NetworkInventory l2Network, String hostUuid, Completion completio
n);


    void check(L2NetworkInventory l2Network, String hostUuid, Completion completio
n);


    L2NetworkType getSupportedL2NetworkType();


    HypervisorType getSupportedHypervisorType();

}
```

当一个 L2 网络被绑定到一个集群，这个拓展点将被集群中的每个主机所调用，这个 Hypervisor 插件可以借此机会在后端主机实现网络；例如，KVM 的插件同时有 `KVMRealizeL2NoVlanNetworkBackend` 和 `KVMRealizeL2VlanNetworkBackend`，后者拓展了 `L2NetworkRealizationExtensionPoint`，为了在 Linux 操作系统创造网桥。这个扩展点是非常有用的，对于不需要知道虚拟机信息的 L2 网络而言。L2NoVlanNetwork 和 L2VlanNetwork 都属于这一类。

然而，一些 L2 网络可能只能在虚拟机被创建的时候实现，例如，一个 L2VxlanNetwork 可能需要查找虚拟机所有者帐户的 VID，为了建立一个 L2 广播域；在这种情况下，Hypervisor 插件可以实现另一个扩展点 `PreVmInstantiateResourceExtensionPoint`：

```
public interface PreVmInstantiateResourceExtensionPoint {

    void preBeforeInstantiateVmResource(VmInstanceSpec spec) throws VmInstantiateRes
ourceException;
```

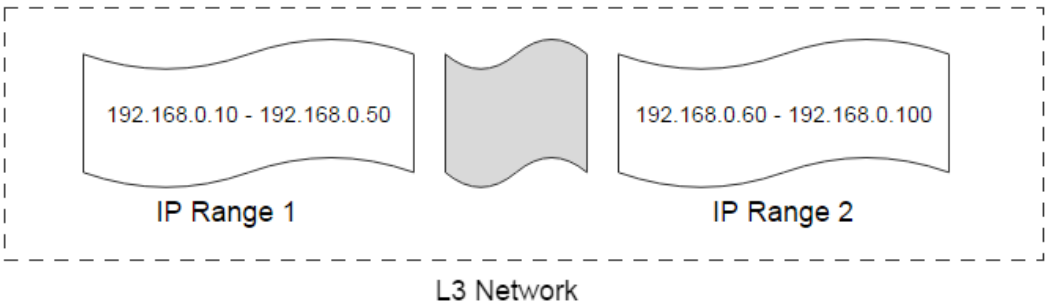
```
    void preInstantiateVmResource(VmInstanceSpec spec, Completion completion);

    void preReleaseVmResource(VmInstanceSpec spec, Completion completion);
}
```

插件可以从 `VmInstanceSpec` 中获取获取目标主机和虚拟机的信息，然后在目标主机创建虚拟机之前实现一个 L2 网络。

L3 网络

一个 L3 网络是创建在 L2 网络上的一个子网，与网络服务相关联；它可以有多个 IP 地址范围，只要它们属于同一个 L3 网络且彼此并不冲突。



在上面的图片中有两个 IP 范围（192.168.0.10 - 192.168.0.50）和（192.168.0.60 - 192.168.0.100），从 192.168.0.51 到 192.168.0.59 的 IP 被保留，这样管理员可以把它们分配给不被 ZStack 管理的设备。

如果没有由网络服务提供模块提供的、和底层的 L2 网络服务相关的网络服务，L3 网络没有任何用处。网络服务提供模块可以提供一个或多个网络服务，例如，ZStack 的默认虚拟路由提供模块能够提供几乎所有常见的网络服务如 DHCP、DNS、NAT 等，而 F5 提供模块可能只提供负载均衡服务。在 ZStack 版本（0.6）中，网络服务提供模块只能在 L2 网络被创建的时候和 L2 网络关联；例如，实现了 `L2NetworkCreateExtensionPoint` 的虚拟路由，将在任何 L2 网络创建后与之关联。

管理员可以将网络服务绑定到一个 L3 网络；对于一类服务，只有一个网络服务提供模块提供的服务可被绑定到这个 L3 网络；例如，你不能将来自不同提供模块的两个 DHCP 服务绑定到同一 L3 网络。在 ZStack 版本（0.6）中，定义了六种网络服务类型：DHCP、

DNS、NAT、EIP、端口转发和安全组，提供模块只需要实现相应的后端：`NetworkServiceDhcpBackend`, `NetworkServiceDnsBackend`,`NetworkServiceSnatBackend`,`EipBackend`,`PortForwardingBackend`, 和 `SecurityGroupHypervisorBackend` 来提供这些服务。在“网络模型 2：虚拟路由器的网络服务提供模块”，我们将讨论我们引用到的提供模块——虚拟路由，你可以探索更多的细节。

总结

在这片文章中，我们简要地解释了 ZStack 的网络模型。在没有挖掘后台 Hypervisor 的细节的情况下，我们演示了 ZStack 是如何将 OSI 模型抽象为 L2 网络（layer 2），L3 网络（layer 3）以及网络服务（layer 4～7）。在下一篇文章中，我们将详细阐述网络服务提供模块的参考实现，关于它如何在虚拟机中实现 DHCP、DNS、NAT、EIP 和端口转发。

ZSTACK--虚拟路由网络服务提供模块

在 ZStack 的网络模型中，OSI 第 4～7 层网络服务被实现为来自不同服务提供模块的小插件。默认提供模块，称为虚拟路由，采用定制的 Linux 虚拟机作为虚拟设备，为每一个 L3 网络提供包括 DHCP、DNS、NAT、EIP 和端口转发在内的网络服务。使用虚拟机作为虚拟路由器的方式的优点有：没有单点故障、对物理设备没有特殊要求，因此用户无需购买昂贵的硬件，就可以在商用设备上实现各种网络服务。

概述

正如“ZStack--网络模型 1：L2 和 L3 网络”中提到的，ZStack 以小插件的方式设计网络服务，供应商可以通过创建网络服务提供模块的方式，选择性地实现他们的硬件或软件支持的网络服务。默认情况下，ZStack 带有一个虚拟路由，它负责使用一个应用虚拟机（Virtual Router VM）实现所有网络服务。

注：事实上 ZStack 有另一个提供模块称为安全组提供模块，它提供了分布式防火墙功能。我们称虚拟路由为默认的提供模块，因为它提供了最常见的，一个云需要的网络服务。

在 IaaS 软件中，实现网络服务有几种方法。

一种方式是使用中心的、功能强大的网络节点，它们通常是一些物理服务器；通过聚集来自不同租户的流量，网络节点将负责流量隔离并使用类似 Linux 网络命名空间的技术来提供网络服务。

另一种方法是使用专用的网络硬件，例如，可编程的物理交换机、物理防火墙、物理负载平衡器，它会要求用户去购买特定的硬件。

最后一个方法是使用网络功能虚拟化（NFV）技术，像 ZStack 的虚拟路由虚拟机，就是在商用 x86 服务器上虚拟化网络服务。

每种方法都有优点和缺点；我们选择 NFV 作为我们的解决方案是出于如下考虑：

1. **需要最少的基础设施**：解决方案应该对用户的物理基础设施需求很少甚至为零；也就是说，用户不应该改变现有的基础设施或购买特殊的基础设施来迎合 IaaS 软件的网络模型。我们不想强迫用户购买特定的硬件或要求他们在一组主机前放置一些特殊的功能服务器。

2. **没有单点故障**：解决方案应该是采用没有单点故障的分布式的方式。一个网络节点崩溃应该只能影响拥有它的租户，不应该影响任何其他租户。

3. **无状态**：网络节点应该是无状态的，这样 IaaS 软件在发生无法预料的错误后，可以轻易摧毁并重新创建它们。

4. **利于高可用性（HA）**：解决方案应该易于实现高可用，这样租户可以要求部署富余的网络节点。

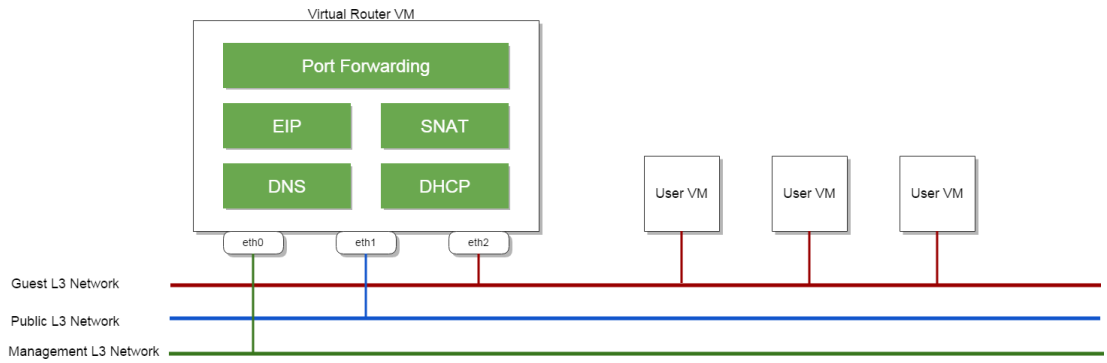
5. **不依赖虚拟机管理程序**：解决方案不应该依赖于 Hypervisor，并且应该和主流的 Hypervisor 完美结合工作，包括 KVM、XEN、VMWare 和 Hyper-V。

6. **较好的性能**：解决方案应该为大多数使用场景提供合理的网络性能。

基于虚拟路由的 NFV 解决方案满足上述所有考虑。我们选择它作为默认的实现，同时也为开发人员提供了采用其他解决方案的可能。

虚拟路由

应用虚拟机（Appliance VMs）是一种特别的虚拟机，运行着定制的 Linux 操作系统，以及特别的帮助管理云的 agents。虚拟路由虚拟机是应用虚拟机概念的第一个实现。这个想法，简单的说，在用户虚拟机第一次被创建的时候，去创建一个为某个 L3 网络提供全部网络服务的虚拟路由虚拟机，只要这个 L3 网络上开启了虚拟路由提供的网络服务。每个虚拟路由虚拟机包含一个 Python agent，它通过 HTTP 协议接收来自 ZStack 管理节点的命令，并在同一 L3 网络给用户虚拟机提供包括 DHCP、DNS、NAT、EIP 和端口转发的网络服务。



上图，显示了在客户 L3 网络上启用了所有网络服务的网络拓扑结构。一个虚拟路由通常有三种 L3 网络：

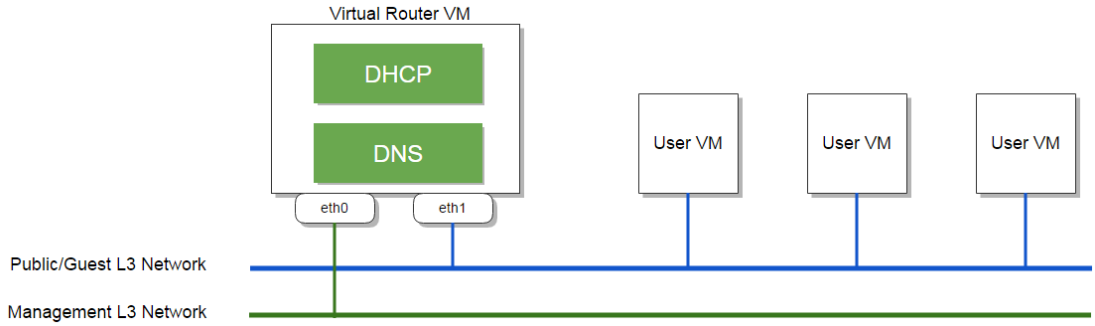
1. **一个 L3 管理网络**指的是 ZStack 管理节点和在虚拟路由虚拟机内的 Python agent 通过 HTTP 协议进行通信的网络，是一个必须的网络，每个虚拟路由都有。

2. **一个公有 L3 网络**是一个可以连接互联网的可选网络，它在虚拟路由虚拟机内提供了默认的路由。如果省略，L3 管理网络将同时被作为管理网络和公有网络使用。

公有网络不需要允许公开访问：把用户虚拟机和外部世界（数据中心的其他网络或互联网）相连的公有网络不需要允许公开访问。例如，当桥接被 VLAN 和数据中心的其他网络(10.x.x.x/x)隔离的客户 L3 网络（192.168.1.0 / 24）时，网络 10.0.1.0/24 可以是一个公有网络，即使它不能被互联网访问。

3. **一个客户 L3 网络**是用户虚拟机连接的网络；和网络服务相关的流量在用户虚拟机和虚拟路由虚拟机内流动。

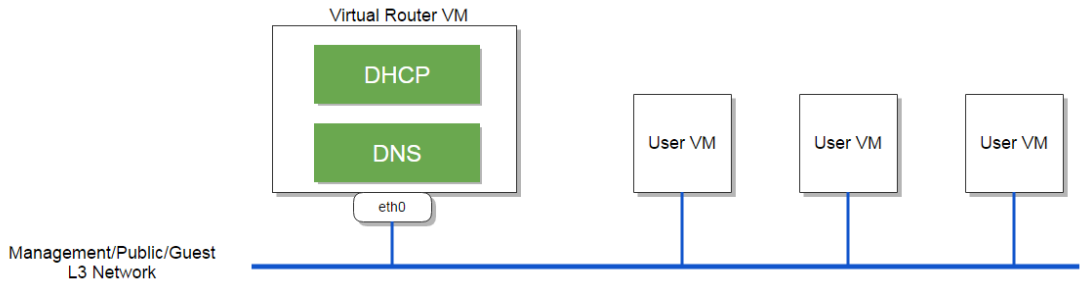
对不同的网络服务组合，L3 网络数量是可变的。例如，如果 DHCP 和 DNS 被启用，网络的拓扑结构变为：



因为没有 NAT 相关的服务（例如 SNAT，EIP），用户的虚拟机不需要一个单独的、隔离的客户 L3 网络，但可以直接连接到公共网络。

***注意**：当然，你可以创建一个只有 DHCP 和 DNS 服务的、隔离的客户 L3 网络，该网络上的虚拟机可以获得 IP，但不能访问外部网络，因为缺失了 SNAT 服务。*

如果我们省略了上图中的 L3 公有网络，那么网络拓扑将变成:



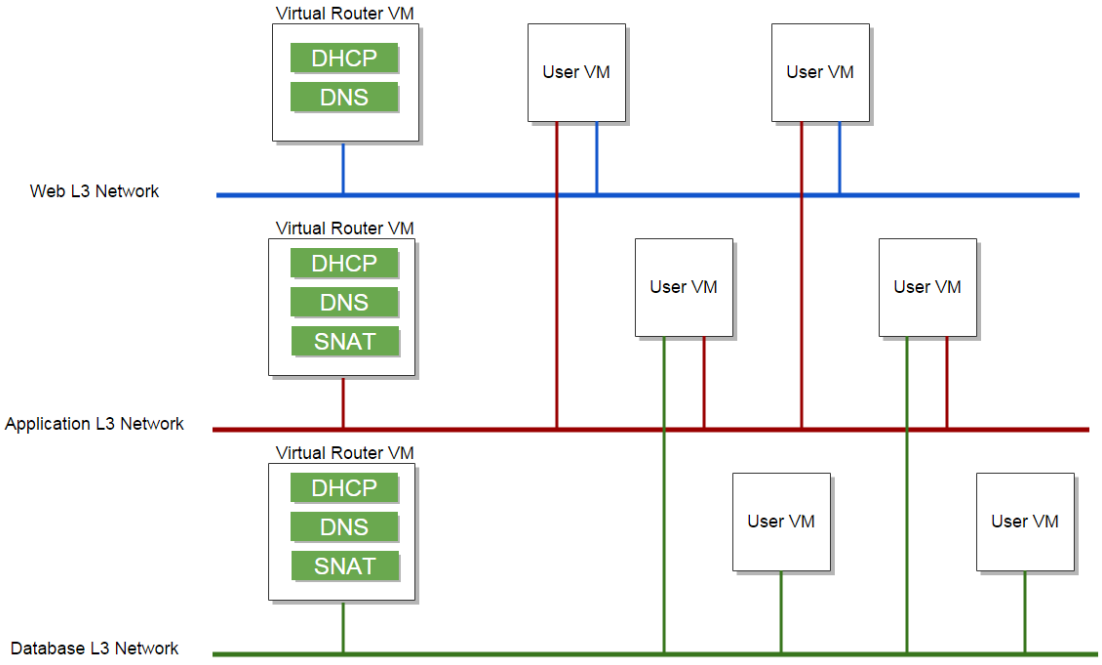
用户可以使用一个虚拟路由计算规格来配置一个虚拟路由虚拟机的 L3 管理网络、L3 公有网络、CPU 速度、以及内存大小。当创建一个虚拟路由虚拟机的时候，ZStack 将会尝试去找到一个合适的虚拟路由计算规格。一个系统标签

`guestL3Network::{l3NetworkUuid},`

可以被用于为一个 L3 客户网络指定一个虚拟路由计算规格，如果没有指定的规格被找到，将会使用一个默认的规格。

注意：*关于系统标签，请参阅 The Tag System。*

在这个 ZStack 版本中（0.6），一个 L3 客户网络可以含有并只能含有一个虚拟路由虚拟机，对于一个多层网络的环境，不同的虚拟路由虚拟机将会服务不同的层：



ZStack 管理节点将会向处于一个虚拟路由虚拟机内部的 Python agent 发送命令，当用户虚拟机启动会停止的时候。以通过 dnsmasq 和 iptables 实现网络服务。Iptables 规则的一小段像这样：

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [1828:141919]
:appliancevm - [0:0]
:snat-fwd-eth1 - [0:0]
-A INPUT -p tcp -m tcp --dport 7272 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 7759 -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -d 192.168.0.236/32 -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -j appliancevm
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -o eth1 -j snat-fwd-eth1
-A FORWARD -i eth1 -o eth0 -j snat-fwd-eth1
-A FORWARD -i eth1 -o eth1 -j snat-fwd-eth1
-A appliancevm -i eth0 -p tcp -m state --state NEW -m tcp --dport 7272 -j ACCEPT
-A appliancevm -i eth0 -p tcp -m state --state NEW -m tcp --dport 9393 -j ACCEPT
-A appliancevm -i eth1 -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A appliancevm -i eth1 -p udp -m state --state NEW -m udp --dport 67:68 -j ACCEPT
-A snat-fwd-eth1 -j ACCEPT
COMMIT
```

注意：*在未来的 ZStack 版本中，网络服务：负载均衡，VPN，GRE 隧道，也将会通过虚拟路由虚拟机来实现。另外虚拟路由虚拟机也将会成为虚拟专用云 VPC 的核心实现元素。*

虚拟路由虚拟机是怎样满足以下考虑的

让我们回顾下我们先前提到的一些考虑，然后看下虚拟路由虚拟机如何能满足它们。

最小的基础设施需求：虚拟路由虚拟机，对数据中心的物理设备没有任何特别需要。它们只是一些类似于用户虚拟机的虚拟机，可以在物理机上被创建。因为使用它们，管理员不必去为复杂的硬件互联做规划。

没有单点故障：对每一个 L3 网络都有一个虚拟路由虚拟机，如果它因为某种原因崩溃了，只有对应的 L3 网络上的用户虚拟机会被影响，而不会对其他 L3 网络产生任何影响。在大多数的使用场景中，一个 L3 网络只属于一个租户，这就是说，只有一个租户会遭受到一个虚拟路由虚拟机的失败。当 L3 网络遭到恶意工具的时候，这特别有用。例如，DDOS。攻击者不能通过攻击一个租户而摧毁整个云内的网络。

无状态：虚拟路由虚拟机是无状态的，所有的配置，来自于 ZStack 管理节点，可以在任何时间被重建。用户有各种各样的选择，用于重建虚拟路由虚拟机中的配置。例如，关闭、启动它们，删除、重建他们，或调用重连 API（ReconnectVirtualRouter API）。

易于实现高可用（HA）：可以部署两个虚拟路由虚拟机，使用虚拟路由冗余协议在主备模式下工作，以实现 HA。一旦主要的虚拟路由失效了，备用的会自动接管，使得网络的宕机时间微不足道。

注意：*这个冗余虚拟路由虚拟机的特性在当前版本中不支持（0.6）.*

1. **Hypervisor 无关**：虚拟路由虚拟机不依赖于 Hypervisor。ZStack 有一个脚本，用于为主流的 Hypervisor 构建虚拟路由虚拟机的模板。
2. **合理的性能**：因为使用了 Linux，虚拟路由虚拟机能够实现该 Liunx 能够提供的合理的性能。用户可以配置一个虚拟路由计算规格，通过更多的 CPU 和更大的内存来为虚拟路由虚拟机提供足够的计算能力，以应对沉重的网络流量。性能上主要的关注点在于，虚拟路由虚拟机和用户虚拟机上公有网卡间的流量，在虚拟路由虚拟机提供 NAT 相关的服务，包括 SNAT、EIP、和端口转发时。在大多数场景中，由于一个公网 IP 通常有几十 MB 的带宽，虚拟路由虚拟机足以胜任一个不错的性能。
   然而，当通过虚拟路由虚拟机的流量需要一个极高的带宽的时候，由于虚拟化导致的显著的网络性能下降时不可避免的；然而，有两种技术可以缓解这个问题：
   a. **LXC/Docker**:由于 ZStack 支持多种 Hypervisor，一旦 LXC 或 Docker 被支持，作为一种轻量级的虚拟化技术，作为容器运行的虚拟路由虚拟机可以近乎原生的性能。
   b. **SR-IOV**:虚拟路由虚拟机可以通过 SR-IOV 被分配物理网卡，以达到原生的网络性能。

   注意：*LXC/Docker 和 SR-IOV 在当前版本中不支持（0.6）.*

另外，用户可以使用系统标签和虚拟路由计算规格来为虚拟路由虚拟机控制物理主机的分配；更进一步，用户甚至可以指派一个物理服务器给一个虚拟路由虚拟机；在 LXC/Docker 或 SR-IOV 的帮助下，虚拟路由虚拟机能接近一个 Linux 服务器能够提供的原生的网络性能。

不管怎样，软件的解决方案有着天生的性能缺陷；用户可以选择为了网络的高性能而选择混合的解决方案；例如，仅为 DHCP 和 DNS 使用虚拟路由虚拟机，将性能敏感的服务留给使用了物理设备的服务提供器。

总结

在这篇文章中，我们演示了 ZStack 的默认网络服务提供器：虚拟路由提供器。解释了它怎么工作并详细阐述了它是怎样满足了我们关于网络服务的考虑。借助虚拟路由虚拟机，ZStack 取得了一个理想的平衡，在灵活性和性能之间。我们相信 90%的用户可以轻松明确地在商业硬件上构建他们的网络服务。

如何在私有云语境下定义 VPC

日前，ZStack 推出了新一版网络架构 VPC 2.0，作为 ZStack 2.3.0 版本中的重磅功能，VPC 2.0 提供全面而强大的网络功能支持，包括：灵活的网络配置，安全可靠的逻辑隔离，以及开启分布式路由功能，优化东西向网络流量、不同子网之间不通过云路由器直接完成互联互通等高级策略，可满足丰富的网络场景实现。

VPC 经过多年的市场教育和实践，大部分公有云用户已经接受了公有云中 VPC 是必不可少的基础组件。但是对于私有云和混合云领域来说，如何保障业务安全？如何充分利用云的潜力？是否还是沿用过去的虚拟化的设计经验，目前尚无定数。

ZStack 在私有云和混合云技术和产品上又做了哪些尝试？VPC2.0 是如何解决网络灵活性、扩展性业务需求的？如何在私有云语境下定义 VPC？

ZStack 网络研发总监 王为 对 VPC2.0 技术架构与应用场景等常见问题进行了深度解答。

人物介绍

王为

ZStack 网络研发总监

前有云 OpenStack 研发负责人，在有云、中科院计算所参与过多个国内大型云计算建设项目，带领网络研发团队向 OpenStack 社区贡献了网络诊断项目 Steth，向 OpenStack、MidoNet、buildbot 等多个开源项目做过贡献，2016 年加入 ZStack，主要负责 SDN、NFV 在 ZStack 上的实践和探索。

Q1：作为网络领域资深专家，请您简单介绍下 VPC2.0，它在私有云和混合云建设中，解决什么样的问题和痛点，以及其最佳应用场景？

在 ZStack VPC1.0 里，其 VPC 的概念和设计主要来源于公有云对隔离性的设计，在现网接入、网络灵活性等诸多方面有所欠缺，在面向复杂的私有云、混合云场景不能完全满足我们的需求，因此 ZStack 对网络模型进行了大量改造，着重提高了灵活性、易用性和性能。

目前 IaaS 产品 VPC 大多来自公有云，着重于安全和隔离性，ZStack VPC 2.0 在安全性和隔离性的基础上大量考虑了用户的实际使用场景、软硬结合的使用场景、利旧的场景等，在现有的 VPC1.0 基础上带来了功能丰富还易用的 VPC2.0，它能够满足企业从几十台虚拟机的小规模应用到几万台虚拟机的大规模部署，并将完整对接混合云 VPC，完成私有云 VPC 的一次全新定义。

Q2：VPC2.0 给用户带来的主要价值是？

- ✧ 灵活的网络配置
- ✧ 安全可靠的隔离
- ✧ 丰富的网络场景
- ✧ 更低的管理成本
- ✧ 更高的网络性能

Q3：在您看来，VPC 技术会朝着什么样的方向发展，下一步产品在网络方面会进行哪些优化？

下一步首先是在更加开放的网络功能上。我们希望能将网络功能做成一个平台开放出去，不只是有我们提供的网络服务，来自各家厂商的无论 NFV、防火墙、监控等产品都能无缝接入到 ZStack 云平台之中，这样才能更进一步的提升效率、降低用户的使用、运维成

本。目前 OPNFV 等组织、OpenDaylight 等项目都做过一些工作，但距离可交付给用户还有相当的距离。

其次是更加的自动化。目前混合云开通、隧道建立不可避免的还存在一些手工的过程，这是目前用户需要等待多的步骤，也是最容易出错的步骤，将来我们希望能够通过引入例如 SDWAN 等技术将这个过程更加高效化，并能够够细致、直观的展示给客户当前网络的状态、质量，能够让用户“傻瓜化”的进行查错、管理。

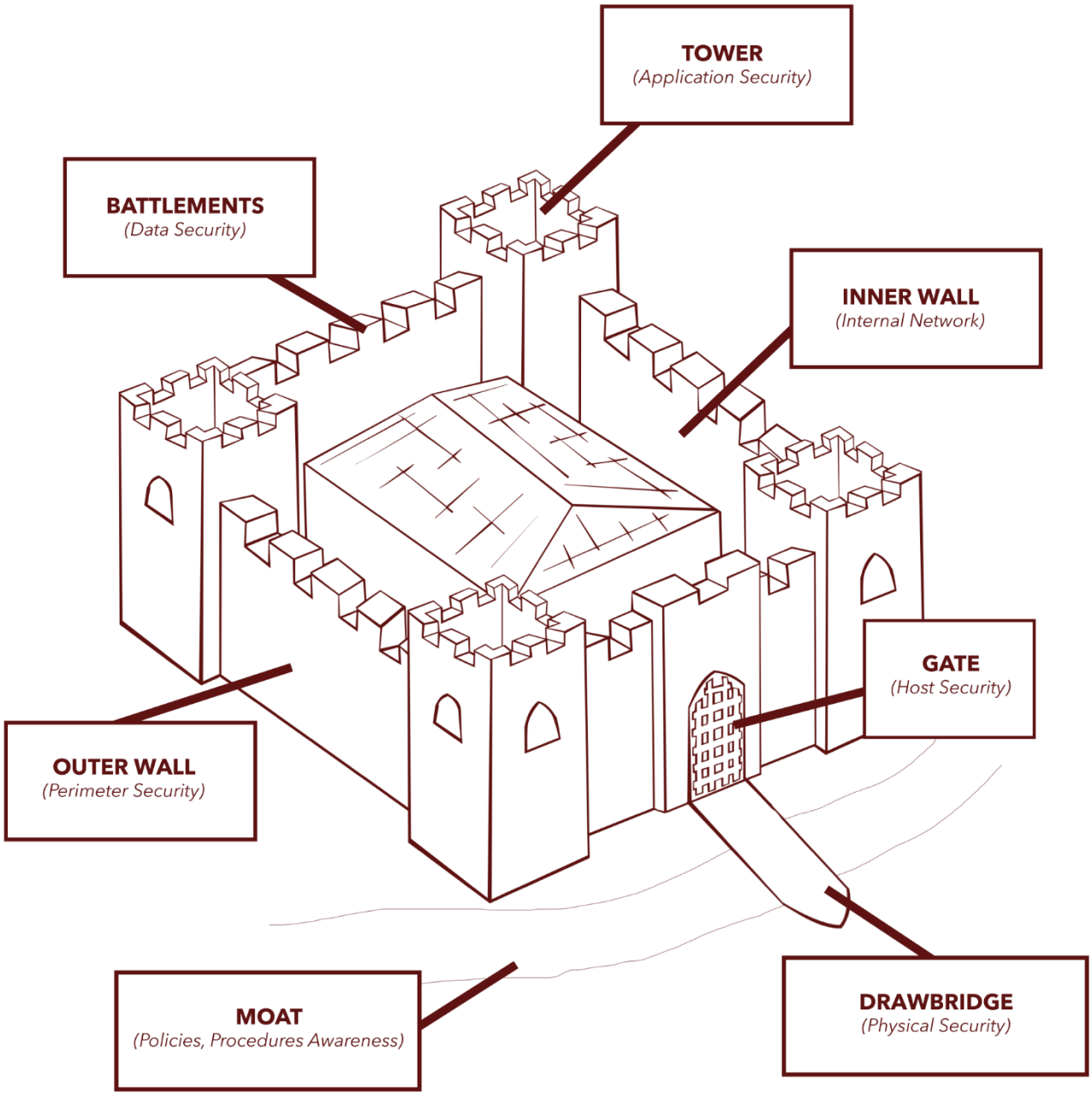
接下来，我们为您详细解读 ZStack VPC 2.0 设计思路与架构。

安全

当用户使用公有云时，其第一考虑的往往是安全。去年一年来不断的发生了各类安全事件，使得用户心生踌躇，这样的问题在私有云中是不存在的——从机柜到电源都完全在企业自己控制下，而且往往在出口部署了高昂的网络安全设备，相比将数据走在和别人共用的网线上，这一做法自然更好保证企业的专属网络和数据，但也给用户带来不小的运维成本。

所以如何在保证安全的前提下减少用户的安全运维成本是我们的第一个考量。要知道网络隔离和弹性计算天然是对立的——隔离的越细致，快速扩容、服务编排、资源回收往往越难，而且这个难度会随业务的数量指数增长。

以往我们可能认为出口部署防火墙足以抵挡一切攻击——如同在一片暴风雨中间设一个城堡，外墙越修越高，试图抵挡一切攻击。

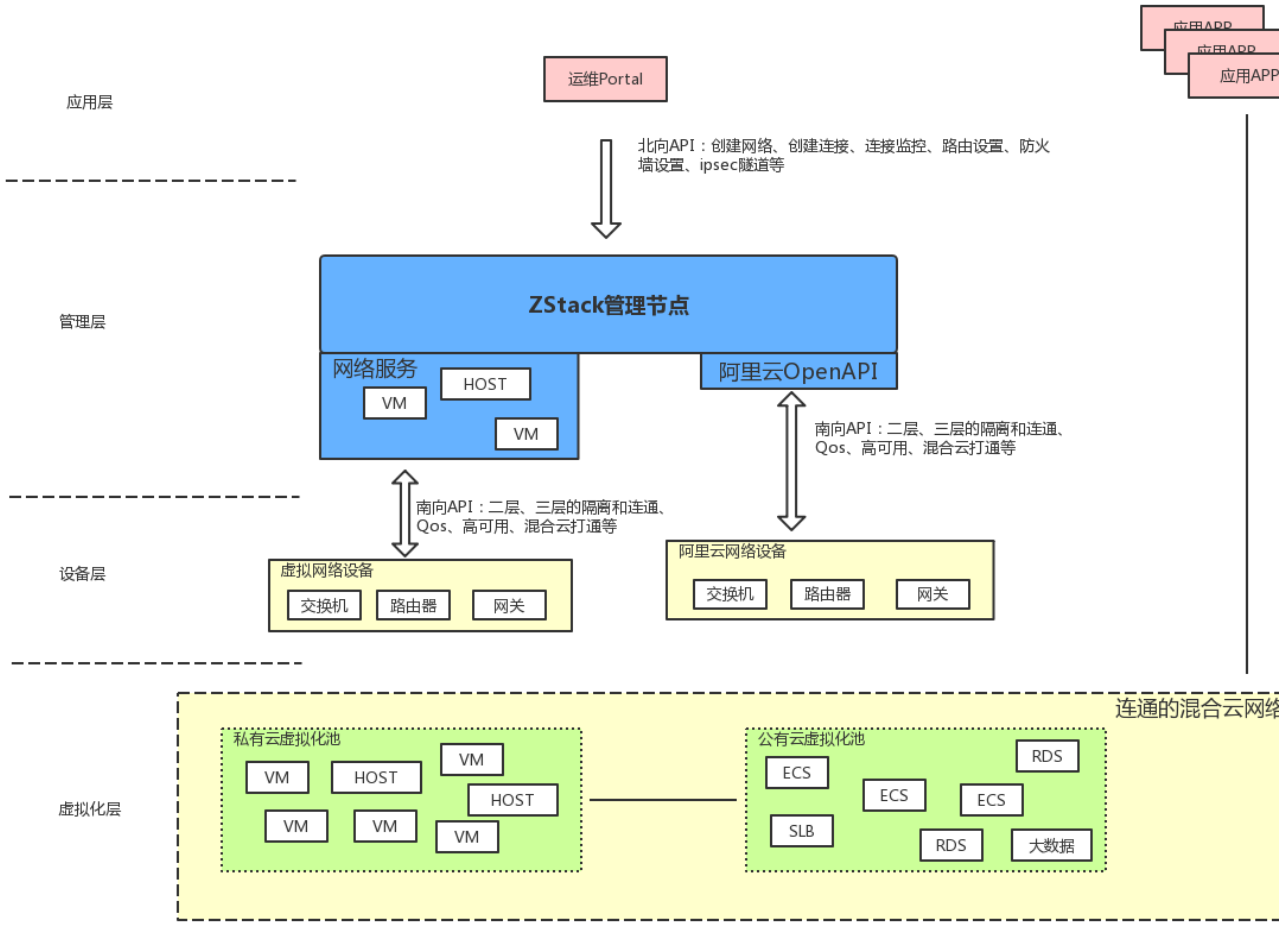


但是实际上，随着 SaaS 服务的兴起、企业互联网以及介入企业内网的设备越来越多，这种设计越来越难以难以维系：

企业对外的数据和网络入口越来越多，比如网站、员工 VPN、购买的 SaaS 服务、对外的 API 服务、用于所开发的网站 App 或手机 App 而提供的数据通道，甚至员工自行连入的手机都可能存在安全隐患，这样企业需要设置的外墙越来越多也越来越高；

应用的爆发式增长，过去，一个应用的扩容可能是以月为单位甚至季度为单位计的，一次上架可能在数月内都足够支撑业务增长，而现在互联网的快速爆发下，应用一旦获得好评，可能会飞速传播，这样对计算能力扩容提出极高要求，而人为配合网络变更极易出错出漏；

数据的爆发式增长带来的混合云需求，与业务的增长并行的，是存储需求的增长，以及数据价值的不断提高，越来越多的企业对存储有更为丰富的需求——除了传统的 SAN、NAS，还有一些业务可能需要超高性能，一些业务需要对象存储，还有冷热分离，大量的日志存储等等要求。此时完全自购设备是很不划算的，对接公有云组成混合云无疑是哥可行选择，但对网络要求更复杂了。

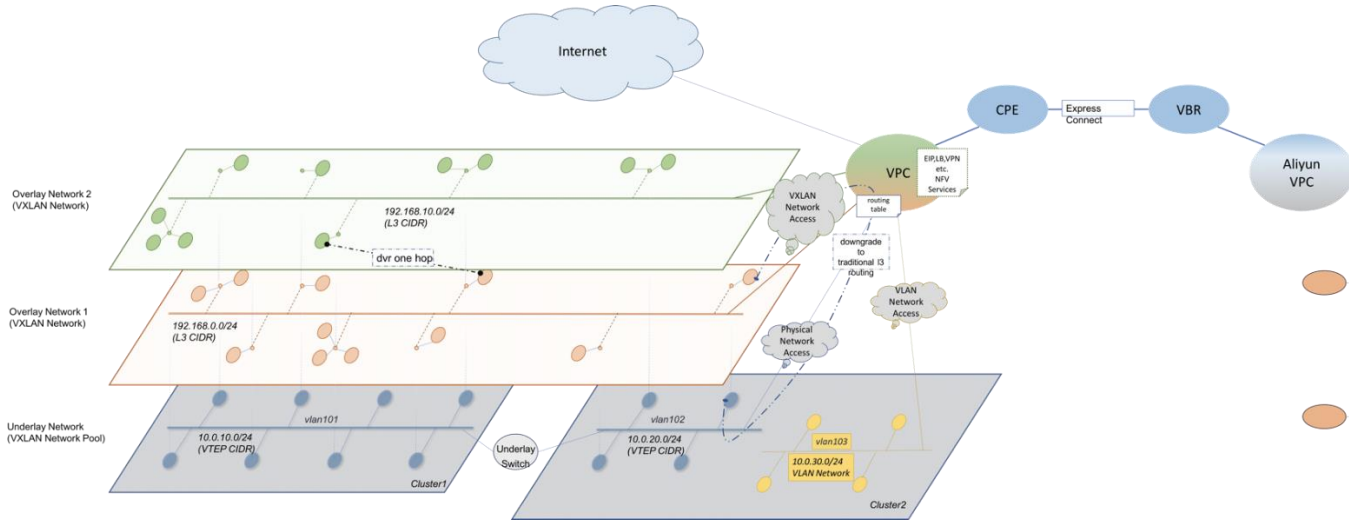


在这种现状下，Google 提出了 BeyondCorp 的安全架构，其基本思想是：安全隔离不再基于 IP、MAC 而是应用、账户访问控制不再是静态的，而是动态的、自动跟随的限制

业务间相互访问的权限甚至协议，但这种先进的安全架构如何应用到企业呢？私有云如何帮助客户完成这些安全目标呢？我们看一个 ZStack 架构下的网络设计。

二层隔离

对于所有网络隔离来说，隔离最彻底的，莫过于二层网络隔离，然而传统二层网络隔离基于 VLAN 或 802.1ad，前者可供使用的域少，而后者配置过于麻烦。还好 ZStack 从 2.0 就提供了 VXLAN 作为网络隔离技术。



与市面上大多数哦 IaaS 不同，ZStack 提供了极为灵活丰富的方法来配置 VXLAN 网络，ZStack 将 VXLAN 的 Underlay 网络 Overlay 网络记为 VXLAN Pool 和 VXLAN Network，前者用于加载到集群、加载 VNI Range 等等，并且在加载时可以选择 VTEP 的地址段，ZStack 就会自动寻找到合适的 VTEP 地址并加载。

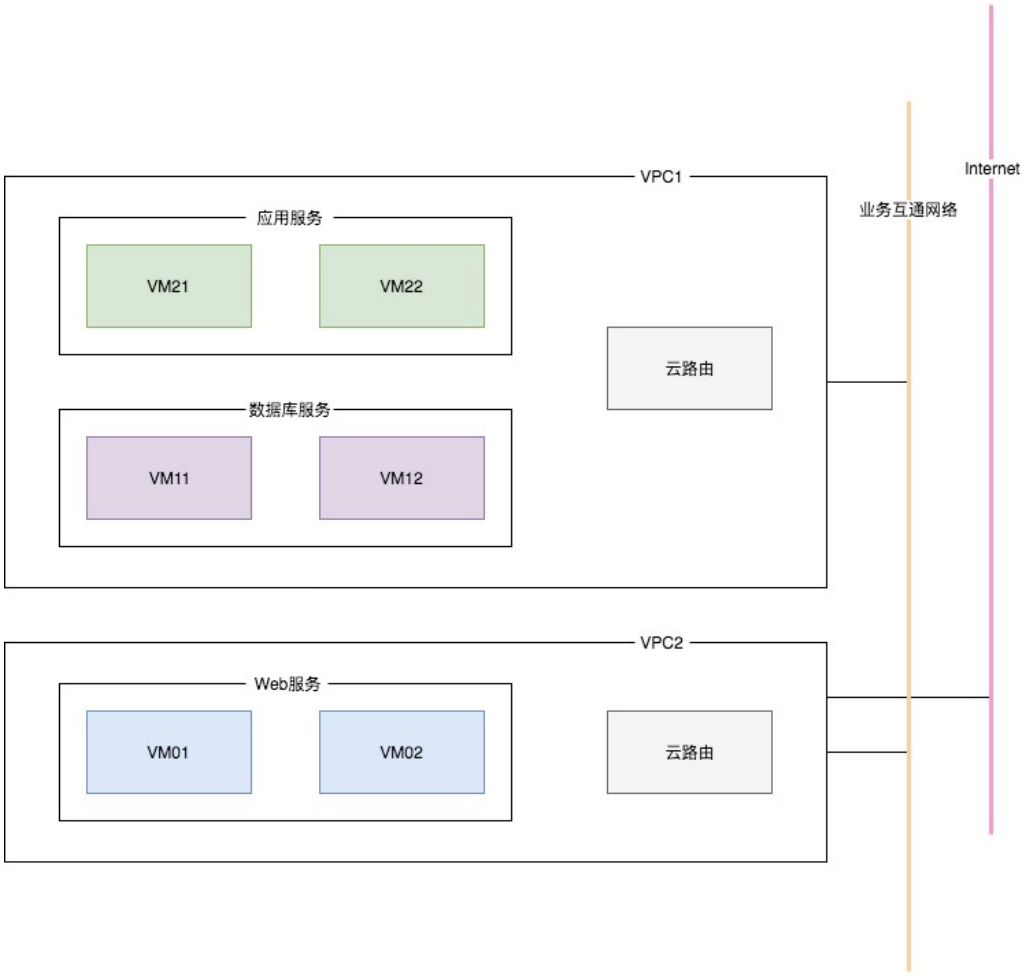
每一个集群可以使用不同的 VTEP 地址段，管理员可以任意划定 VNI 范围，并将这个 Pool 共享给最终用户或部门的运维管理人员，再在之上创建 VXLAN 网络。整个过程高度自动化和灵活——甚至服务器上 VTEP 所需要的地址由于某些原因被修改，只需要在 ZStack 界面上重连物理机即可自动同步！

在生产实践中，可以将需要的业务单独部署 VXLAN，也可以将 VXLAN 划分给业务部门供其自由配置，保证其最底层的网络隔离——无论底层的 ARP 欺骗抑或高层的服务嗅探都不可能突破这一关卡。

自定义路由

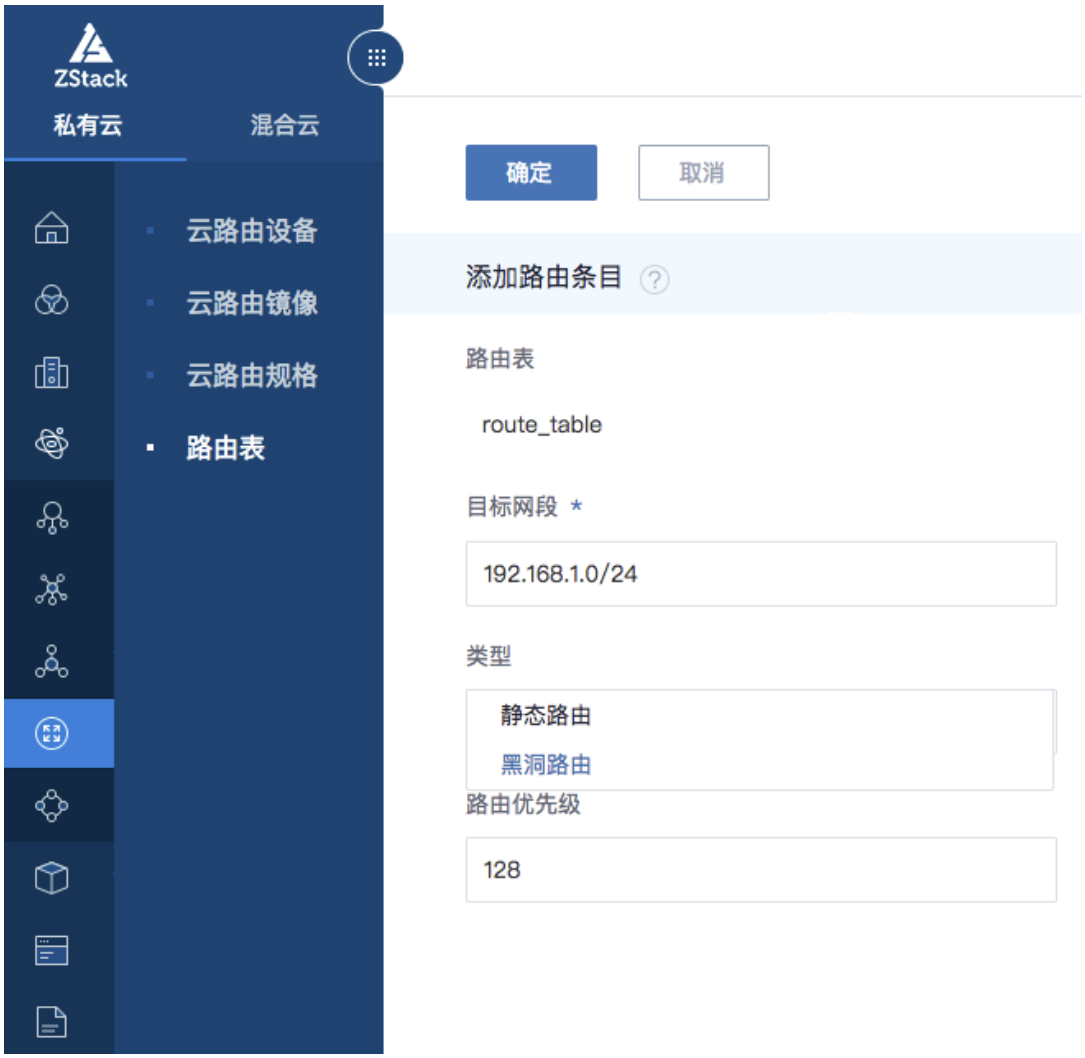
从 ZStack 2.1 开始我们提供路自定义路由这一功能，这一功能咋看之下与安全毫无关系，实则不然。

首先，在公有云中公网往往比较好定义——一定就是 Internet，唯一区别是可能是电信网络或联通网络或 BGP 网络等等。而私有云则不然，私有云中内网一定是部署业务的网络而公网却不一定是 Internet，例如下图中的架构：



假设我们有两个 VPC 部署了三个业务，其中应用和数据库为了性能考虑放在了一个 VPC，而 Web 服务是一个公共服务在一个单独的 VPC 中，此时应用服务可以通过云路由，走自定义路由走到 VPC2，这里 VPC2 运用了我们的云路由支持多公网的功能，这一功能一定是和自定义路由相互配合使用的。

此外，如何保障数据库的安全？不被渗透或探测？安全组是一方面，另一方面为了防止探测流量和意外的路由导致数据库被外部发现，我们还可以在 ZStack 中指定黑洞路由：



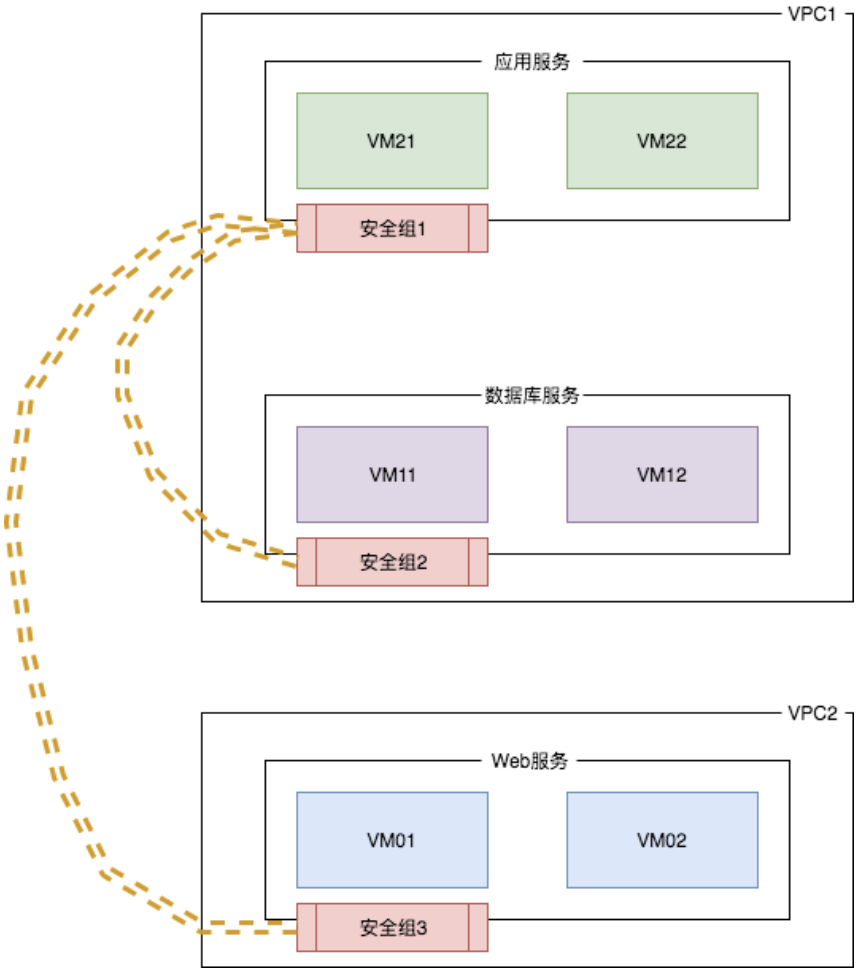
这样可以彻底的保证数据库服务的地址段不会泄漏到 VPC 之外了。

考虑到复杂网络的路由复杂性，ZStack 还提供了 API 供管理员查看云路由中哪些路由条目正在生效，哪些路由条目虽然配置但实际上是不生效的。

基于源的安全组

传统安全组基于 IP、协议和端口，这样的安全组除了策略跟随（虚拟机任意漂移不影响其安全策略的生效）之外很难说比传统 ACL 有多少优势，与 zone based firewall 都略显单薄。

但是 ZStack 自 2.1 起提供了基于源的安全组，也就是安全组可以当作一个安全域来使用，ZStack 可以自动识别来自不同安全组的流量，从而做到灵活的访问控制而无需反复设置 IP 或网段，减少出错，提高效率。



如上图，我们可以针对源安全组来设置安全组规则，例如安全组 1 可以配置当源来自安全组 2 时开放哪些端口，来自安全组 3 时开放那些端口。

其他

此外，ZStack 还有其他很多安全技术例如源地址过滤以防止 IP、MAC 地址伪造，DHCP 服务器防伪造避免用户从错误的服务器上获取地址等等。通过这些技术的有效运用，管理员可以根据自己的需求和实际环境配置做够安全、可信的网络。

私有云和公有云不同，这些系统的设置和使用都是可以由用户自己配置和使用的，用户与此同时可以接入一些既有的安全设备例如 WAF、防火墙、流量清洗等等，可以参考下面的内容。

灵活

对于公有云，一切规则是由供应商决定的，供应商往往会根据自身的考虑或者技术的局限而定义这些规则，一方面会导致云上的网络使用和传统存在很多不同，一些管理操作带来不便，例如公有云的数据库带来了 scale 的方便，但也带来了调试的不便；另一方面容易形成对其资源的使用形成 all in，例如可能我们只想使用公有云上的数据库，但为此将业务虚拟机也只好迁到云上，随之而来的还要迁移存储系统、备份系统甚至一套相配套的内容，带来了很多不便。

而当我们设计一款私有云时，这一切都需要考虑，而不是以一句“客户需要设计 Cloud Native”的理由认为客户的需求不合理。特别是 ZStack 是一个产品化的私有云，产品会不断升级，既要添加新的功能和场景，又要兼容旧的设计和使用，甚至是用户不合理的使用。

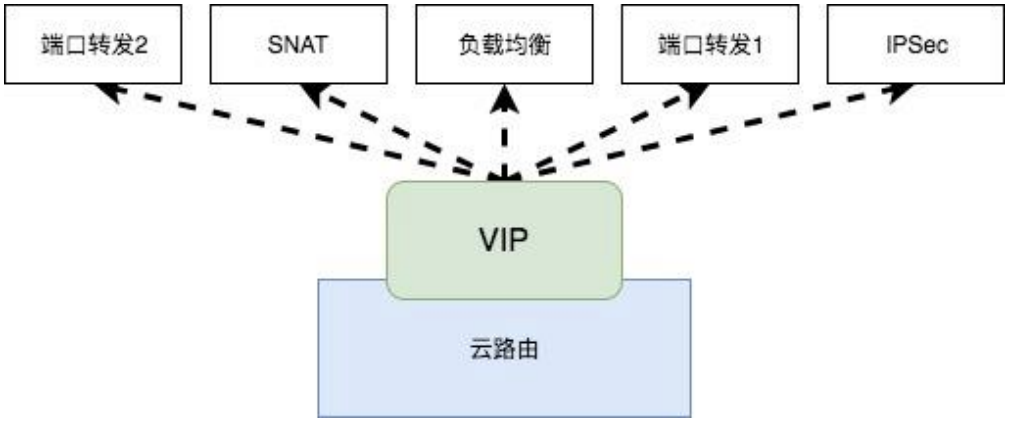
ZStack VPC 2.0 在灵活性上带来了下面这些亮点：

多网络服务复用地址

对于中小型的私有云，一个常见需求是——公有网络的 IP 地址不够用，特别是这个共有网络是一个真的 Internet 上 IPv4 地址时。

ZStack 为用户提供了丰富的网络服务，包括但不限于弹性 IP、负载均衡、IPSec VPN、端口转发、SNAT 等等，而与此同时用户的 IP 地址却可能捉襟见肘，比如用户只有一个地址但既想要用 SNAT 开放虚拟机到公网的访问，又想用其作为 IPSec 的地址连接混合云。

这个需求在数据平面是可行的，但对云平台的控制平面却做出很大挑战，云平台要很好的协调每个服务对 IP 的需求，将其合理的配置到云路由上。从 ZStack 2.3.0 开始，我们支持了一个 IP 同时用在负载均衡、IPSec、端口转发、SNAT 等所有不要求独占 IP 的服务上！而且用户可以任意划定其端口的使用，从此将 VIP 从一个网络属性，彻底开放成为一个可以池化的资源，大大解放了私有云对 IP 的使用。



此外，考虑到每个服务使用不同的端口（SNAT 外），ZStack 里还可以对每个端口做不同的 QoS，达到既对 VIP 本身可以做 QoS，同时可以对不同服务做 QoS 的能力。

物理网络接入

在私有云中，既有设备或既有网络的接入同样是无法绕开的问题。一来企业要避免投资浪费，二来短时间内很多软件的实现在性能或功能上都无法与硬件相媲美，例如应用交付控制器 ADC、流量清洗设备等等。

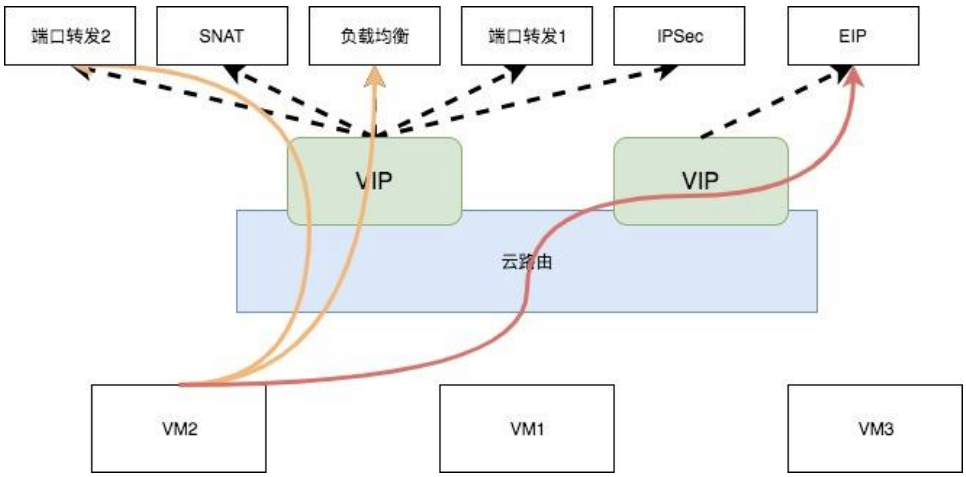
为了解决物理设备的接入，ZStack 提供了多公网、自定义路由、多网卡、VXLAN 网关等多种方案。其中前两者前面有所介绍，多网卡很好理解。基于 VXLAN 网关的方案则是因为 ZStack 采用标准的 VXLAN 协议，因此其他使用 VXLAN 协议的网络设备理论上都可以与 ZStack 的 VTEP 协商构成同一个 VXLAN Underlay 网络，达到虚拟网络与物理网络混合的效果。

很多时候，云网技术会通过 DVR 来优化流量，造成伪造网关带来的一系列问题，从而造成运维复杂度上升甚至部分需求无法实现，这个问题在 ZStack 中是不存在的，具体将在下面分布式路由中介绍。

网络服务跟随

在很多云中，网络服务定义在 VPC 中，实现在云路由之上，这样云路由成为 VPC 的关键角色，公有云中往往避免直接暴露给用户云路由，而像私有云也往往将云路由的相关操作列为一个关键操作——用户不能随意删除或卸载，否则可能会导致网络服务的消失。

而 ZStack 中却有所不同，我们认为用户的网络服务创建是面向业务的，也是服务于业务的（特别是弹性 IP、端口转发等），此时如果我们想将虚拟机转移到另一个 VPC 或者删除掉云路由就丢失网络服务是不可取的。我们通过实现网络服务跟随，达到了，网络服务一旦创建，将跟随这个虚拟机而非云路由，如果这个云路由被删除，一旦重新创建或加载，所有网络服务将自动重新应用——这就像从此的 NAT 规则不再依赖一个实际的防火墙，而是直接于你的业务绑定，即使换一个防火墙、重建一个防火墙，系统总会尝试帮你恢复业务，达到资源的真正池化、无依赖、完全灵活。



如上图，虽然网络服务定义在云路由上，但最终其用于 VM2，所以 VM2 不消失，无论云路由发生什么变化，网络服务最终会应用在 VM2 上。

性能

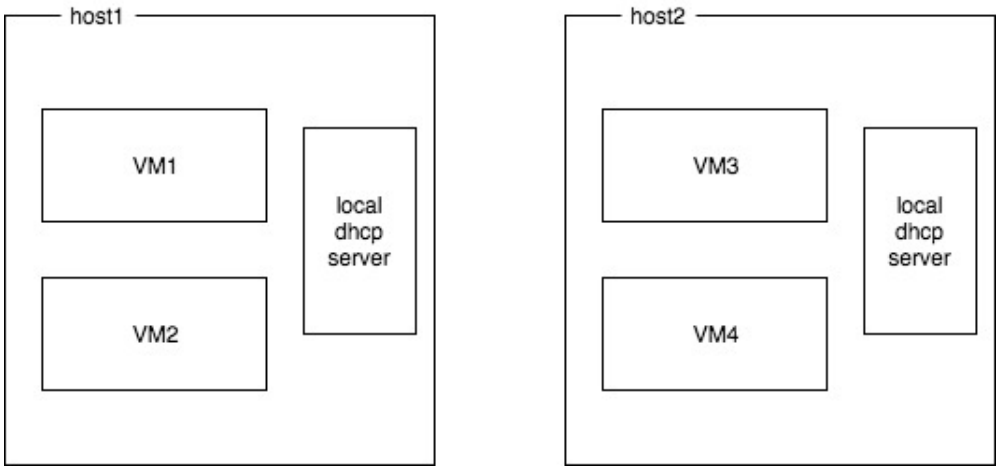
虽然不像公有云那样单区域承载百万台甚至更多虚拟机，但私有云也并非没有性能要求，甚至随着用户的需求变化，我们需要能够适应用户从小到大的规模要求，小规模时不浪费额外资源，大规模时也要有方案与之应对。

特别是云路由接入多个子网后，其上的流量由于企业对东西向流量的需求可能会变得很大，如果一个云路由成为整个网络的性能瓶颈是不被用户接受的，除了单点问题之外，性能上的占用也很可观。

传统的网络协议将整个过程一般都定义为集中式的，例如 DHCP 服务器、例如路由网关，而且集中控制也带来实现上的便利性——特别是 SNAT、路由表这些服务，而一些服务我们发现它其实可以通过一些手段做分布式的优化，而且其重要性也值得我们去这样做，例如 DHCP 和网关。

分布式 DHCP

ZStack 从最早的 Flat 网络即支持了分布式 DHCP，但云路由网络是集中式的 DHCP，从 2.1 开始，我们开始将云路由网络 DHCP 改为分布式的设计，并继承到了 VPC 网络之中。



分布式路由

分布式路由是 ZStack VPC 2.0 中一个极为重点的功能。与业界一些 DVR 的实现对规模有限制甚至很脆弱难以应用到生产不同，ZStack VPC 2.0 在设计之初就经过了精心的考虑，因此它有着以下的特点：

无先验知识

无消息队列

无网关欺骗

无集中式控制器

即时开关

首先，无先验知识是指整个流量优化系统无需从 ZStack 数据库拉取信息。为什么这个很重要呢？因为 ZStack 设计原则里有很重要的一点，就是管控面与数据面的分离，管控面的故障不会扩散到数据面上，任何情况下优先保证用户业务的不受影响。

在这种设计原则下，我们自然不能去拉取 ZStack 数据库的信息——即使所有网络信息都记录在了 ZStack 数据库上，但一来不能保证数据库的永久不宕机，二来数据库的数据其实是不能真实的反应网络状态的，用户随时可能修改自己的 IP、创建虚拟网卡等等。

其次，无消息队列。OpenStack 中 DVR 的控制数据通过消息队列执行最终被认为是一个欠考虑的设计，因为消息队列既有着容量限制还存在自身的可靠性问题，因此 ZStack DVR 抛弃了消息队列，实现了一种私有协议为 ZSNP（ZStack Network Protocol），通过这个协议直接在 IP 层上传递消息，并实现了无需知道虚拟机所在 Host 的信息的前提下将信令直接投递到 Host 上所在 Agent 的功能。

第三，无网关欺骗。传统的 DVR 都是基于网关欺骗实现的，这种想法非常直接——既然我要做网关，那肯定要实现一个伪网关，与此同时带来的，就是防止网关泄漏、伪网关与真网关的判断等等一系列问题，而且很容易造成用户自定义的路由难以实现、物理设备不好接入。

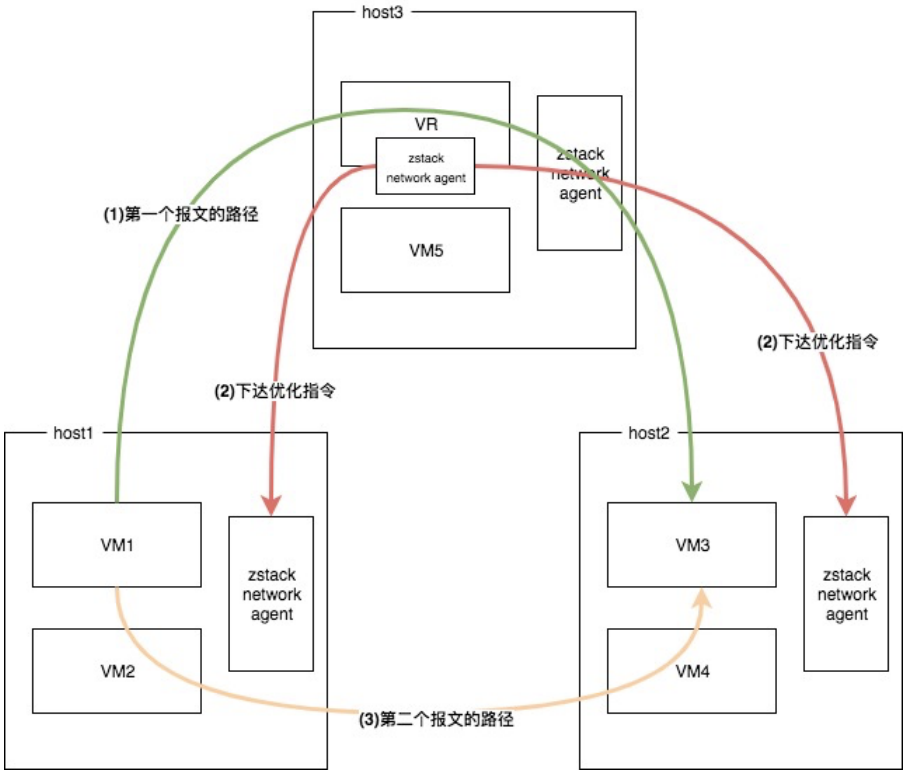
而 ZStack 希望在做分布式路由时，一要可靠，二要尽量避免影响用户的使用习惯，因此实现了无网关欺骗的无状态流量优化，从而完整兼容过去的所有功能，用户也可以透明的享受分布式路由带来的好处。

第四是没有集中控制器。在传统的 DVR 设计中除了与 CMS 数据库交互外，还会往往有集中化控制器这一问题，那么一旦控制器失效，轻则无法优化流量，重则网络中断，也是用户所万万无法接受的。

在 ZStack VPC 中，流量优化由在云路由中的 Agent 发起，每个 Agent 只关心自己所在云路由上的流量，因此不存在系统单点，而且这样的 Agent 类似于一种旁挂机制而非直连，那么即使 Agent 挂掉，也不会导致网络失效。而是已优化流量会逐渐回退到传统集中式路径上。

最后，一个很有意思的功能是 ZStack VPC 的分布式路由可以由用户自行决定是否开启，而且通过前面原理的介绍我们可以知道这种开关是分布在每个云路由上的，用户可以选择部分云路由开启而部分关闭，而且这种开启即时生效，关闭也会在规则老化后彻底回退到传统路径。

此外 ZStack VPC 2.0 还有着大量潜力尚待挖掘——例如基于优化数据的网络连接探测、优化流量统计等等，将在未来的版本逐步推出。



总结

上面简单分析了 ZStack VPC 2.0 架构的特点——安全、灵活、高性能。实际上 ZStack VPC 里还有大量的特性和功能待用户挖掘，特别是有很多结合混合云的场景在这篇文章里没有介绍，欢迎大家试用、讨论。