ZStack 技术白皮书精选

ZStack--网络模型 1：L2 和 L3 网络

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。

ZSTACK--网络模型 1：L2 和 L3 网络

ZStack 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。

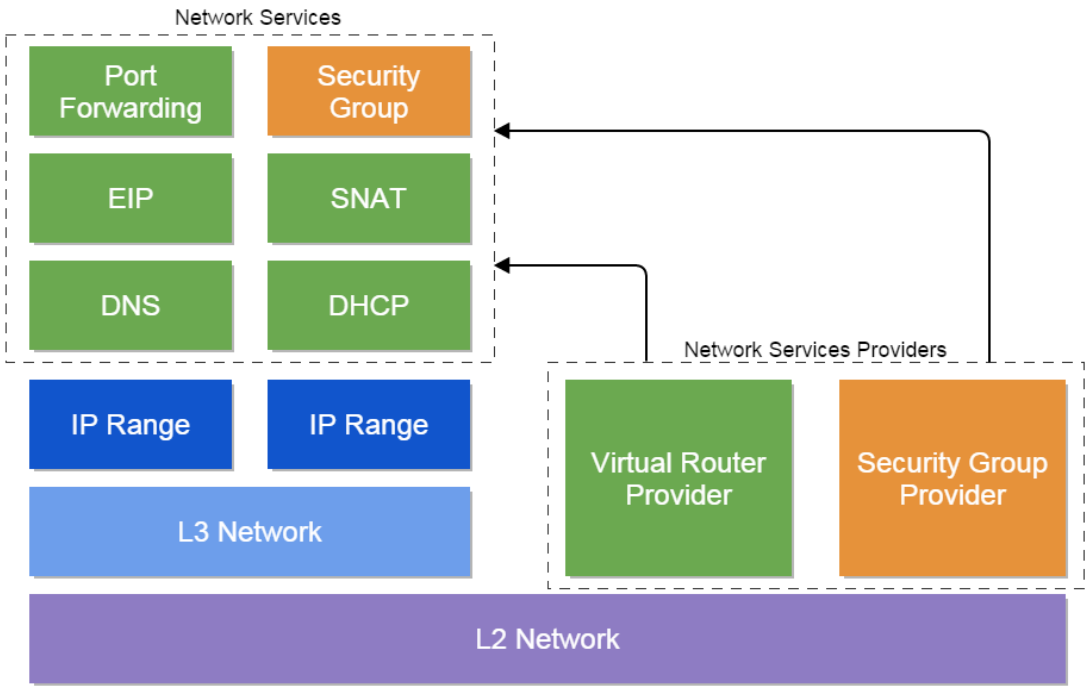
注：*我们将不涉及任何在 Hypervisor 端虚拟化技术的网络实现细节；例如，我们将不讨论 ZStack 如何在 Linux 操作系统中创造网桥或 VLAN 设备。这篇文章的目的是给你介绍 ZStack 网络模型的简要构想。如果你还没有阅读“通用插件系统”的话，我们强烈建议你去阅读一遍，因为许多和插件相关的术语将在下文被提到。*

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

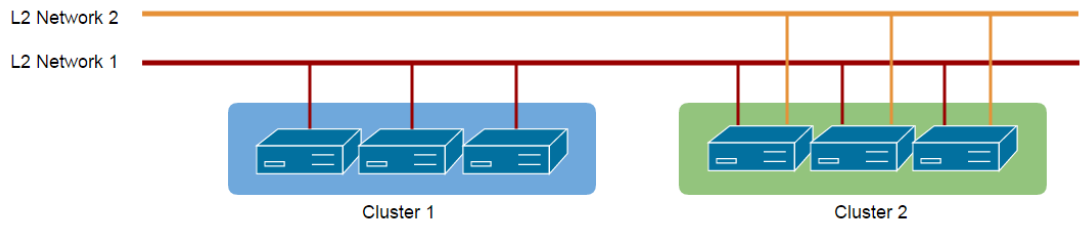
一个 L2 网络负责提供一种二层隔离方法，可以是一个纯粹的 L2 技术（如 VLAN），或一个网络覆层（overlay）技术（如 GRE 隧道，VxLAN）。ZStack 不关心 L2 网络在后端使用的技术细节，所以包含必要的 L2 信息的数据结构--L2NetworkInventory--是高度抽象的：

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

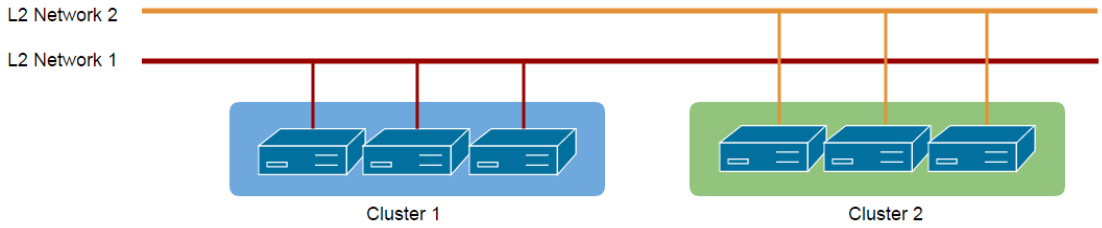
L2 网络的子类型可能有额外的属性，例如，L2VlanNetwork 有一个额外的字段的 `vlan`。

绑定策略

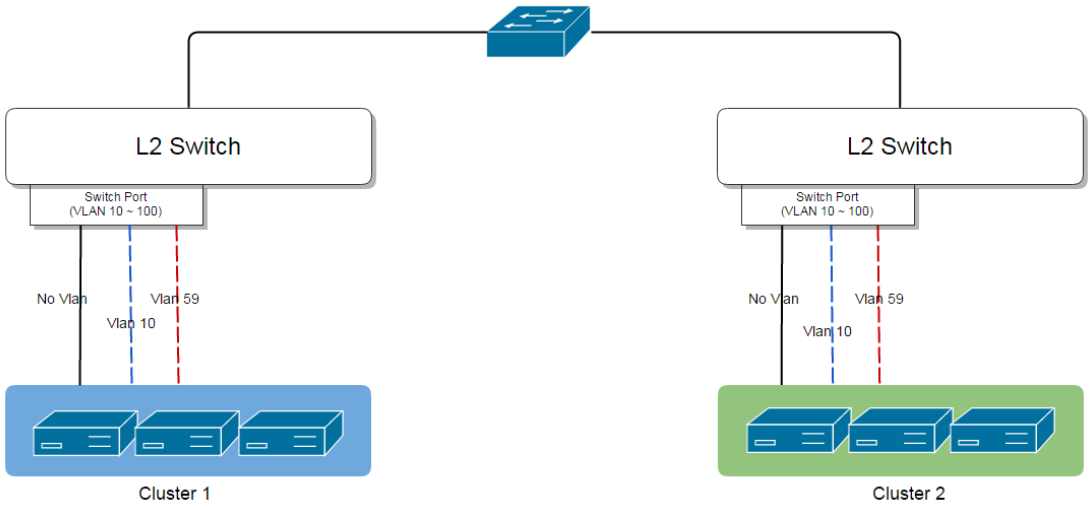
在真实的数据中心中，L2 网络通常代表主机之间一个的物理网络连接。例如，在同一 L2 交换机下的主机可能在同一个 L2 网络中。网络的连接不是一成不变的，它可能会在任何数据中心的物理设备改变的时候改变，例如管理员重新配置（re-wire）一个 L2 交换机。为了提供一种灵活的、描述主机和 L2 网络之间的关系的方式，ZStack 采用了一种所谓的*绑定策略*，允许一个 L2 网络连接从多个集群（主机的集合）中绑定/解绑。



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
completion);



    void check(L2NetworkInventory l2Network, String hostUuid, Completion
completion);
```

```
    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。

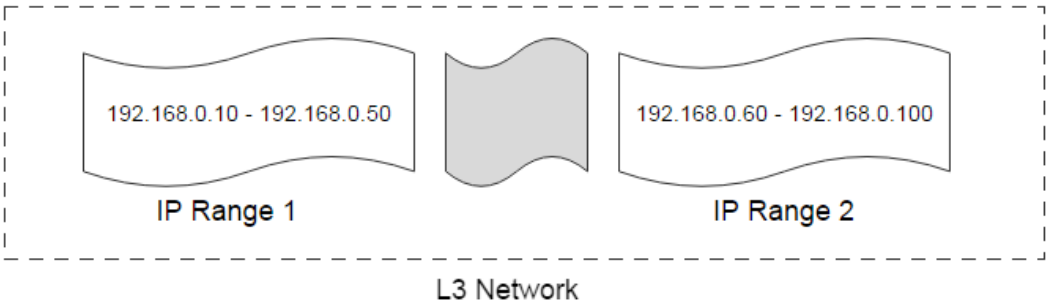
然而，一些 L2 网络可能只能在虚拟机被创建的时候实现，例如，一个 L2VxlanNetwork 可能需要查找虚拟机所有者帐户的 VID，为了建立一个 L2 广播域；在这种情况下，Hypervisor 插件可以实现另一个扩展点 `PreVmInstantiateResourceExtensionPoint`：

```
public interface PreVmInstantiateResourceExtensionPoint {

    void preBeforeInstantiateVmResource(VmInstanceSpec spec) throws
VmInstantiateResourceException;


    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。