**Linux下的vlan实践**

编 写 人：孙自翔

统 稿 人：

编写时间：2014-03-05

部 门 名：

审 核 人：

审核时间：

修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订内容简述 | 修订日期 | 版本 | 修订人 |
| 1 | 初稿 | 2014-03-05 | 0.1 | 孙自翔 |
|  |  |  |  |  |
|  |  |  |  |  |

正文目录

[1 文档说明 4](#_Toc382401510)

[1.1 本文档目的 4](#_Toc382401511)

[1.2 本文档要达成的目标 4](#_Toc382401512)

[1.3 本文档不涉及 4](#_Toc382401513)

[1.4 更进一步（最终目的） 4](#_Toc382401514)

[2 以太网 4](#_Toc382401515)

[2.1 以太网概念 4](#_Toc382401516)

[2.2 以太网分类 4](#_Toc382401517)

[3 vlan概述 4](#_Toc382401518)

[3.1 VLAN特性总结 5](#_Toc382401519)

[3.2 VLAN实现方式 5](#_Toc382401520)

[3.3 VLAN实现需要的外围设施 5](#_Toc382401521)

[3.4 Trunk链路和access链路 6](#_Toc382401522)

[3.5 802.1q协议封包格式 7](#_Toc382401523)

[3.5.1 EEE 802.1Q协议介绍。 7](#_Toc382401524)

[3.5.2 EEE 802.1Q封包格式。 7](#_Toc382401525)

[3.5.1 VLAN标签格式 8](#_Toc382401526)

[4 实验场景 8](#_Toc382401527)

[4.1 网络结构图 8](#_Toc382401528)

[4.2 进一步简化 9](#_Toc382401529)

[4.3 简单的抓包演示。 10](#_Toc382401530)

[4.3.1 不同网桥下的虚拟机之间的单播通信。 10](#_Toc382401531)

[**4.3.2** 同一网桥下的虚拟机之间的单播通信**。** 11](#_Toc382401532)

[5 linux下的vlan 11](#_Toc382401533)

[5.1 vlan标签tagged和detagged 11](#_Toc382401534)

[5.1.1 VLAN标签格式 11](#_Toc382401535)

[5.1.2 VLAN标签的tagged和detagged 12](#_Toc382401536)

[5.2 需要的内核模块和工具 13](#_Toc382401537)

[5.3 创建vlan的实验环境说明 14](#_Toc382401538)

[5.4 命令行创建vlan 14](#_Toc382401539)

[5.4.1 通过vconfig命令 14](#_Toc382401540)

[5.4.2 通过ip命令 15](#_Toc382401541)

[5.5 Debian系统下持久化配置vlan 15](#_Toc382401542)

[5.5.1 通过配置文件 15](#_Toc382401543)

[5.5.2 配置的另一种方式 16](#_Toc382401544)

# 文档说明

## 本文档目的

云计算的网络环境比较复杂，其中涉及的概念比较多，某个概念理解不透彻就可能影响对整个环境的理解。

其中某些概念尤其重要，包括vlan、网桥。本文档详细介绍linux下vlan的知识，并对vlan进行抓包分析，以深刻理解vlan的概念.

理解vlan和网桥的概念之后，可以对云网络的数据包走向做深入理解。

## 本文档要达成的目标

* 深入理解vlan概念。
* Linux下如何创建vlan软设备。并通过抓包进一步理解vlan概念。其中涉及到的网桥的概念，后续会进一步阐述。

## 本文档不涉及

* 文档不涉及如何在cisco路由器中配置vlan，只关心linux下的软设备

## 更进一步（最终目的）

* 理解在各种情况下云网络的数据包走向.

# 以太网

<http://zh.wikipedia.org/wiki/%E4%BB%A5%E5%A4%AA%E7%BD%91>

不论是vlan还是网桥，都需要深入理解以太网的概念。比如共享以太网和交换以太网的概念.

## 以太网概念

<http://zh.wikipedia.org/wiki/%E4%BB%A5%E5%A4%AA%E7%BD%91>

<http://en.wikipedia.org/wiki/Ethernet>

以太网是一种计算机局域网组网技术。其他局域网技术还有令牌环网(token ring)，FDDI等

## 以太网分类

* CSMA/CD共享介质以太网(传统以太网，已经不适用了)
* 交换式以太网

# vlan概述

关于VLAN的资料很多，典型的资料如下链接。

<http://www.firewall.cx/networking-topics/vlan-networks/214-vlan-concept.html>

<http://www.firewall.cx/networking-topics/vlan-networks/218-vlan-access-trunk-links.html>

<http://www.firewall.cx/networking-topics/vlan-netorks/219-vlan-tagging.html>

<http://www.microhowto.info/tutorials/802.1q.html>

<http://www.microhowto.info/howto/configure_an_ethernet_interface_as_a_vlan_trunk_on_debian.html>

## VLAN特性总结

<http://www.rhexpo.com/wang-luo-an-quan/wang-luo-an-quan-27.htm>

VLAN的特性有：

* 有效控制网络广播域，如图所示，一个交换机分割成两个广播域。

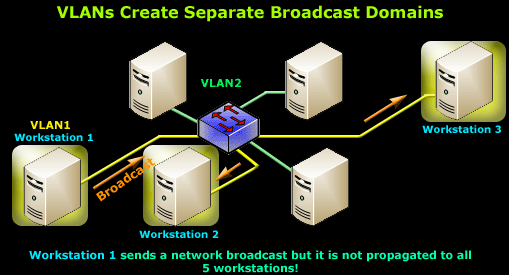


图3-1 vlan示意图

每个广播域都是一个独立的网络，所以VLAN可以减少网络部署成本。比如使用VLAN 可以在不增加交换机的同时，把现存的网络分割成若干个独立的网络，而且网络之间互相不能访问。

* 有效的网络监控 流量管理
* 提高网络的安全性，如限制网络中的计算机相互访问权限。
* 实现不同地域部门内的局域网通信

在云计算的网络环境中主要关注前两点。

## VLAN实现方式

基于交换式的以太网要实现虚VLAN主要有以下几种途径,具体可以查相关资料.

* 基于端口的虚拟局域网
* 基于MAC地址(网卡的硬件地址)的虚拟局域网
* 基于IP地址的虚拟局域网。
* **802.1Q tag-based （本文重点介绍）**

## VLAN实现需要的外围设施

<https://wiki.ubuntu.com/vlan>

为了创建并使用VLAN，还需要一些软设施或硬件的支持

* 以太网帧格式符合802.1Q协议定义的格式.
* 支持VLAN的二层交换机.
* NIC (Network Interface Card)

## Trunk链路和access链路

<http://www.firewall.cx/networking-topics/vlan-networks/218-vlan-access-trunk-links.html>

<http://baike.baidu.com/link?url=gjhDopcKQ_3GuKJcg2kPv1QXbNvmOi_bIim0l-Ib6oxKECPQjywjYruI9x0louuRVQ8MEmLkTkvzALwgEzy-0q>

在交换网络中，链路有两种类型,接入(access)链路和中继(trunk)链路。

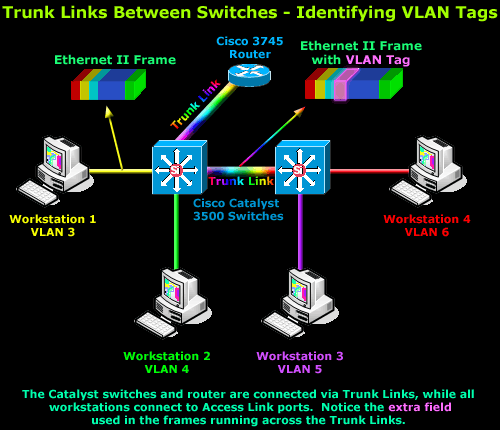


图3-2 vlan trunk 和access 示意图

* **(trunk)中继链路可以承载多个VLAN。即可以让连接在不同交换机上的同一vlan互通。**

在实际的企业环境中，如果各个vlan分布在不同交换机的端口上，同一vlan之间相互通信，需要经过trunk链路。如果交换机1的VLAN1中的机器要访问交换机2的VLAN1中的，可以把两台交换机的级联端口设置为Trunk端口，这样，当交换机把数据包从级联口发出去的时候，会在数据包中做一个标记（TAG），以使其它交换机识别该数据包属于哪一个VLAN，这样，其它交换机收到这样一个数据包后，只会将该数据包转发到标记中指定的VLAN，从而完成了跨越交换机的VLAN内部数据传输。VLAN Trunk目前有两种标准，ISL和802.1q，前者是Cisco专有技术，后者则是IEEE的国际标准，除了Cisco两者都支持外，其它厂商都只支持后者。

* **接入链路是隶属于同一交换机的VLAN成员。**

每个access link分配给一个vlan，同一vlan的不同端口相互之间通信没有vlan tag。不同vlan之间不能通信，除非通过三层交换机或者路由器。

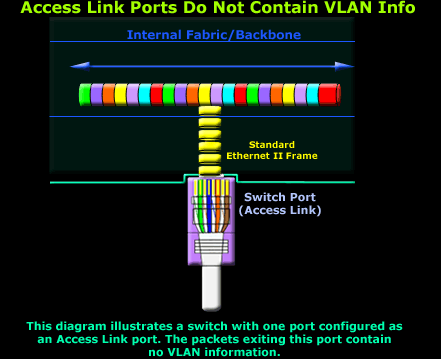


图3-3 vlan access数据单元示意图

## 802.1q协议封包格式

### EEE 802.1Q协议介绍。

<http://www.firewall.cx/networking-topics/vlan-networks/219-vlan-tagging.html>

<http://zh.wikipedia.org/wiki/IEEE_802.1Q>

vlan tag的类型主要有IEEE 802.1q、InterSwitch Link (ISL思科专用)

IEEE 802.1Q以及VLAN Tagging属于互联网下IEEE 802.1的标准规范，允许多个网桥(Bridge)在信息不被外泄的情况下公开的共用同一个实体网络。IEEE 802.1Q -英文缩写写为dot1q.

IEEE 802.1Q定义一个关于VLAN连接介质访问控制层和IEEE 802.1D生成树协议的具体概念模型。这个模型允许各个独立的VLAN以以太网交换机的数据链路层或路由器互相连接。

### EEE 802.1Q封包格式。

802.3定义的以太网的帧格式如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 前导码 | 帧开始符 | MAC 目标地址 | MAC 源地址 | [**802.1Q**](http://en.wikipedia.org/wiki/802.1Q)**标签 (可选)** | [以太类型](http://en.wikipedia.org/wiki/Ethertype)或长度 | 负载 | [冗余校验](http://zh.wikipedia.org/wiki/%E5%86%97%E4%BD%99%E6%A0%A1%E9%AA%8C) | [帧间距](http://en.wikipedia.org/wiki/Interframe_gap) |
| 10101010 7个octet | 10101011 1个octet | 6 octets | 6 octets | (4 octets) | 2 octets | 46–1500 octets | 4 octets | 12 octets |
|  | | 64–1522 octets | | | | | |  |
| 72–1530 octets | | | | | | | |  |
| 84–1542 octets | | | | | | | | |

可以看到协议中有一个可选的[802.1Q](http://en.wikipedia.org/wiki/802.1Q" \o "en:802.1Q) 标签，这个就是专门用来支持VLAN的。

### VLAN标签格式

上面显示的[802.1Q](http://en.wikipedia.org/wiki/802.1Q" \o "en:802.1Q) 标签占用4 octets，每个octets8个bit。在后面会具体抓包显示这些字段（链接）.它的格式为

|  |  |  |  |
| --- | --- | --- | --- |
| 16 bits | 3 bits | 1 bit | 12 bits |
| TPID | PCP | CFI | VID |

* **标签协议识别符**(Tag Protocal Identifier, TPID): 一组16bits的域,其数值被设定为0x8100，用来辨别某个IEEE 802.1Q的帧为已被标签的，而这个域所被标定位置与乙太形式/长度在未标签帧的域相同，这是为了用来区别未标签的帧。
* **优先权代码点**(Priority Code Point, PCP): 以一组3位元的域当作[IEEE 802.1p](http://zh.wikipedia.org/w/index.php?title=IEEE_802.1p&action=edit&redlink=1" \o "IEEE 802.1p（页面不存在）)优先权的参考，从0(最低)到7(最高)，用来对资料流(音讯、影像、档案等等)作传输的优先级。
* **标准格式指示**(Canonical Format Indicator, CFI): 1位元的域。若是这个域的值为1，则MAC地指则为非标准格式；若为0，则为标准格式；在乙太交换器中他通常默认为0。在乙太和令牌环中，CFI用来做为两者的相容。若帧在乙太端中接收资料则CFI的值须设为1，且这个端口不能与未标签的其他端口桥接。
* **虚拟局域网识别符**(VLAN Identifier, VID): 12位元的域，用来具体指出帧是属于哪个特定VLAN。值为0时，表示帧不属于任何一个VLAN；此时，802.1Q标签代表优先权。16位元的值0x000和0xFFF为保留值，其他的值都可用来做为共4094个VLAN的识别符。

# 实验场景

本文档在云计算联调环境下的云主机中做实验。

## 网络结构图

下图为简化了的部分网络结构图。只显示出一个vlan.



图4-1 云计算环境下的vlan示意图

## 进一步简化

为了理解vlan的trunk和access口,进一步简化如下。



图4-2 简化示意图

几点说明:

* 图中只显示了两个计算节点10-120-120-11和10-120-120-13
* 每个节点上有两个虚拟机，分别为vlan11-1,vlan11-2 和 vlan13-1，vlan13-2

其固定ip地址分别为

|  |  |
| --- | --- |
| 主机名 | 固定IP |
| vlan11-1 | 10.160.161.30/22 |
| vlan11-2 | 10.160.161.49/22 |
| vlan13-1 | 10.160.163.94/22 |

* 每个虚拟机接的是网桥(交换机)的access口，同一个网桥下的虚拟机之间通信时没有tag的。
* Linux下的vlan设备eth1.201相当于网桥的trunk口,经过trunk口的物理接口是有tag的。

## 简单的抓包演示。

本节给出简单的抓包演示，后面会详细介绍vlan标签,何时添加vlan标签， 以及何时去除vlan标签。

<http://forum.ivorde.ro/linux-adding-and-removing-vlan-tagged-interfaces-t15231.html>

### 不同网桥下的虚拟机之间的单播通信。

不通网桥下的虚拟机之间的通信要经过trunk口，必然要经过接口eth1。

以vlan11-1 ping vlan13-1为例

root@vlan11-1:~# ping vlan13-1

PING vlan13-1.novalocal (10.160.163.94) 56(84) bytes of data.

64 bytes from vlan13-1.novalocal (10.160.163.94): icmp\_req=1 ttl=64 time=4.59 ms

64 bytes from vlan13-1.novalocal (10.160.163.94): icmp\_req=2 ttl=64 time=0.748 ms

……

nbs@10-120-120-11:~$ sudo tcpdump -nn -vvv -e -i eth1 -c 1 icmp

tcpdump: WARNING: eth1: no IPv4 address assigned

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

20:44:35.527831 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype **802.1Q (0x8100),** length 102: **vlan 201**, p 0, ethertype IPv4, (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 9981, seq 83, length 64

1 packet captured

1 packets received by filter

0 packets dropped by kernel

### 同一网桥下的虚拟机之间的单播通信**。**

同一网桥下的虚拟机之间的单播通信是不会经过trunk口的,所以流量不会经过eth1，只在acces口之间转发数据，没有vlan tag，此时只能在br100上抓包。

以vlan11-1 ping vlan11-2为例

root@vlan11-1:~# ping vlan11-2

PING vlan11-2.novalocal (10.160.161.49) 56(84) bytes of data.

64 bytes from vlan11-2.novalocal (10.160.161.49): icmp\_req=1 ttl=64 time=1.73 ms

64 bytes from vlan11-2.novalocal (10.160.161.49): icmp\_req=2 ttl=64 time=0.436 ms

……

nbs@10-120-120-11:~$ sudo tcpdump -nn -vvv -e -i br100 -c 1 icmp

tcpdump: listening on br100, link-type EN10MB (Ethernet), capture size 65535 bytes

20:51:12.810273 fa:16:3e:79:fc:d5 > fa:16:3e:6d:2d:c5, ethertype **IPv4 (0x0800)**, length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.161.49: ICMP echo request, id 10269, seq 9, length 64

1 packet captured

2 packets received by filter

0 packets dropped by kernel

# linux下的vlan

## vlan标签tagged和detagged

### VLAN标签格式

抓包查看vlan标签的帧格式

**root@vlan11-1:~# ping vlan11-2**

PING vlan11-2.novalocal (10.160.161.49) 56(84) bytes of data.

64 bytes from vlan11-2.novalocal (10.160.161.49): icmp\_req=1 ttl=64 time=0.263 ms

64 bytes from vlan11-2.novalocal (10.160.161.49): icmp\_req=2 ttl=64 time=0.346 ms

**nbs@10-120-120-11:~$ sudo tcpdump -nn -vvv -e -i eth1 -c 1 icmp**

tcpdump: WARNING: eth1: no IPv4 address assigned

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

11:52:32.763839 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype **802.1Q (0x8100)**, length 102: vlan 201, p 0, ethertype IPv4, (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 11018, seq 548, length 64  
**nbs@10-120-120-11:~$ sudo tcpdump -nn -vvv -e -i eth1 -c 1 icmp -w -|hexdump**

tcpdump: WARNING: eth1: no IPv4 address assigned

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

1 packet captured

13 packets received by filter

0 packets dropped by kernel

0000000 c3d4 a1b2 0002 0004 0000 0000 0000 0000

0000010 ffff 0000 0001 0000 9d32 5316 25cc 000c

0000020 0066 0000 0066 0000 16fa 263e d96b 16fa

0000030 793e d5fc **0081 c900** 0008 0045 5400 0000

0000040 0040 0140 ece0 a00a 1ea1 a00a 5ea3 0008

0000050 490f e72a 0800 9d31 5316 0000 0000 04ad

0000060 000a 0000 0000 1110 1312 1514 1716 1918

0000070 1b1a 1d1c 1f1e 2120 2322 2524 2726 2928

0000080 2b2a 2d2c 2f2e 3130 3332 3534 3736

000008e

其中输出结果中标红的4个字节即为vlan[802.1Q](http://en.wikipedia.org/wiki/802.1Q) 标签。含义如下

|  |  |  |
| --- | --- | --- |
| 网络字节序 | 0x0081 | 0xc900 |
| 实际表示 | 0x8100 | 0x00c9 |
| 二进制 | 略 | 0000.0000.1100.1001 |
| 含义 | TPID（固定值0x8100） | 前3位(000)为优先级，其值为0  接着一位是‘标准格式指示’,其值为0  剩下的12位(0000.0110.0100)为vlan id，即为十进制的201 |

### VLAN标签的tagged和detagged

通过抓包分析vlan标签在linux下何时tagged及detagged。

**实验一：何时添加标签(tagged)**

当vlan11-1 (11上的虚拟机)ping vlan13-1(13上的虚拟机)时，在宿主机11上其数据包走向为

br100 -> eth1.201 -> eth1

**root@vlan11-1:~# ping vlan13-1**

PING vlan13-1.novalocal (10.160.163.94) 56(84) bytes of data.

64 bytes from vlan13-1.novalocal (10.160.163.94): icmp\_req=1 ttl=64 time=0.780 ms

64 bytes from vlan13-1.novalocal (10.160.163.94): icmp\_req=2 ttl=64 time=1.28 ms

**nbs@10-120-120-11:~$ sudo tcpdump -nn -vvv -e -i br100 -c 1 icmp**

tcpdump: listening on br100, link-type EN10MB (Ethernet), capture size 65535 bytes

13:26:45.107824 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype IPv4 **(0x0800),** length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 14836, seq 15, length 64

**nbs@10-120-120-11:~$ sudo tcpdump -nn -vvv -e -i eth1.201 -c 1 icmp**

tcpdump: WARNING: eth1.201: no IPv4 address assigned

tcpdump: listening on eth1.201, link-type EN10MB (Ethernet), capture size 65535 bytes

13:26:53.107815 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype IPv4 **(0x0800)**, length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 14836, seq 23, length 64

**nbs@10-120-120-11:~$ sudo tcpdump -nn -vvv -e -i eth1 -c 1 icmp**

tcpdump: WARNING: eth1: no IPv4 address assigned

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

13:26:57.107873 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype **802.1Q (0x8100)**, length 102: vlan 201, p 0, ethertype IPv4, (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 14836, seq 27, length 64

可以看出数据包是在eth1接口上tagged的。在vlan接口上仍然是未打标签的。

**实验二：何时去除标签标签(detagged)**

当vlan11-1 (11上的虚拟机)ping vlan13-1(13上的虚拟机)时，在宿主机13上接受数据包，其数据包走向为

eth1-> eth1.201 -> br100

**root@vlan11-1:~# ping vlan13-1**

PING vlan13-1.novalocal (10.160.163.94) 56(84) bytes of data.

64 bytes from vlan13-1.novalocal (10.160.163.94): icmp\_req=1 ttl=64 time=0.450 ms

64 bytes from vlan13-1.novalocal (10.160.163.94): icmp\_req=2 ttl=64 time=0.424 ms

**nbs@10-120-120-13:~$ sudo tcpdump -n -vvv -e -i eth1 -c 1 icmp**

tcpdump: WARNING: eth1: no IPv4 address assigned

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

13:34:26.211565 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype **802.1Q (0x8100)**, length 102: vlan 201, p 0, ethertype IPv4, (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 15134, seq 11, length 64

**nbs@10-120-120-13:~$ sudo tcpdump –n -vvv -e -i eth1.201 -c 1 icmp**

tcpdump: WARNING: eth1.201: no IPv4 address assigned

tcpdump: listening on eth1.201, link-type EN10MB (Ethernet), capture size 65535 bytes

13:34:33.212294 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype IPv4 **(0x0800),** length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 15134, seq 18, length 64

**nbs@10-120-120-13:~$ sudo tcpdump -n -vvv -e -i br100 -c 1 icmp**

tcpdump: listening on br100, link-type EN10MB (Ethernet), capture size 65535 bytes

13:34:41.213710 fa:16:3e:79:fc:d5 > fa:16:3e:26:6b:d9, ethertype IPv4 **(0x0800)**, length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)

10.160.161.30 > 10.160.163.94: ICMP echo request, id 15134, seq 26, length 64

可以看出数据包是在eth1.201接口上detagged的。

## 需要的内核模块和工具

<https://wiki.ubuntu.com/vlan>

<http://docs.oracle.com/cd/E19862-01/820-1606-12/vlan.html>

* vlan的创建需要内核模块8021q支持的。如果不显式加载，在创建vlan时会报错。

root@debian-1:~# lsmod |grep 802

8021q 18747 0

garp 13025 1 8021q

* ip命令或vconfig命令是用来配置vlan的用户空间的工具，其安装包分别为

root@debian-1:~# aptitude install iproute

root@debian-1:~# aptitude install vlan

## 创建vlan的实验环境说明

无法在云计算环境的虚拟机中演示vlan的创建，后续演示vlan的创建在个人电脑的虚拟机上进行，桥接两台虚拟机(debian-1、debian-2)为例。（能在虚拟机中创建vlan，但无法使不同虚拟机上的vlan通信。）

每台虚拟机挂载两块网卡：

* eth0，10.240.152.0/21 桥接两台虚拟机，充当外网(WAN)
* eth1，VMnet2,分配ip段192.168.1.0/24，充当内网(LAN)，配置vlan在这个接口上进行。

## 命令行创建vlan

以创建Id为100的vlan为例。

### 通过vconfig命令

运行命令vconfig –h查看帮助

* 在debian-1上创建vlan并配置IP(192.168.1.100)

root@debian-1:~# vconfig add eth1 100 # 配置vlan

Added VLAN with VID == 100 to IF -:eth1:-

root@debian-1:~# ip link show eth1.100 # 显示接口

6: eth1.100@eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFA

link/ether 00:0c:29:60:a7:7b brd ff:ff:ff:ff:ff:ff

root@debian-1:~# ip addr add 192.168.1.100/24 dev eth1.100 # 为接口配置IP

root@debian-1:~# ip addr show eth1.100 # 显示接口IP

6: eth1.100@eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN

link/ether 00:0c:29:60:a7:7b brd ff:ff:ff:ff:ff:ff

inet 192.168.1.100/24 scope global eth1.100

* 同样在debian-2上创建vlan并配置IP(192.168.1.200)
* 验证vlan互通性。

为了证明只有vlan之间才能互通，给eth1配置一笔名IP(192.168.1.150)

root@debian-1:~# ifconfig eth1:1 192.168.1.150 netmask 255.255.255.0 #(配置别名)

root@debian-1:~# ifconfig eth1:1 # 查看别名

eth1:1 Link encap:Ethernet HWaddr 00:0c:29:60:a7:7b

inet addr:192.168.1.150 Bcast:192.168.1.255 Mask:255.255.255.0

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

Interrupt:19 Base address:0x2080

root@debian-2:~# ping 192.168.1.200 #ping vlan中的另一个节点

PING 192.168.1.200 (192.168.1.200) 56(84) bytes of data.

64 bytes from 192.168.1.200: icmp\_req=1 ttl=64 time=0.105 ms

root@debian-2:~# ping 192.168.1.150 # ping非vlan节点

PING 192.168.1.150 (192.168.1.150) 56(84) bytes of data.

From 10.9.8.70 icmp\_seq=1 Destination Host Unreachable

* 删除vlan

root@debian-1:~# vconfig rem eth1.100

Removed VLAN -:eth1.100:-

root@debian-2:~# vconfig rem eth1.100

Removed VLAN -:eth1.100:-

### 通过ip命令

<http://www.microhowto.info/tutorials/802.1q.html>

运行命令ip –h查看帮助。

* 在debian-1上创建vlan并配置IP(192.168.1.100)

root@debian-1:~# ip link add link eth1 name eth1.100 type vlan id 100

root@debian-1:~# ip addr add 192.168.1.100/24 dev eth1.100

root@debian-1:~# ip addr show eth1.100

12: eth1.100@eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN

link/ether 00:0c:29:60:a7:7b brd ff:ff:ff:ff:ff:ff

inet 192.168.1.100/24 scope global eth1.100

* 同样在debian-2上创建vlan并配置IP(192.168.1.200)，并验证vlan的互通性

## Debian系统下持久化配置vlan

<http://www.microhowto.info/howto/configure_an_ethernet_interface_as_a_vlan_trunk_on_debian.html>

### 通过配置文件

用命令行配置vlan，在网络重启时会丢失。可以通过文件配置进行持久化。Debian的配置文件为/etc/network/interface.

#在/etc/network/interface中添加如下几行

auto eth1.100

iface eth1.100 inet static

address 192.168.1.100

netmask 255.255.255.0

然后执行

root@debian-2:~# ifup eth1.100

root@debian-2:~# ip addr show eth1.100

6: eth1.100@eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP

link/ether 00:0c:29:93:25:0c brd ff:ff:ff:ff:ff:ff

inet 192.168.1.200/24 brd 192.168.1.255 scope global eth1.100

inet6 fe80::20c:29ff:fe93:250c/64 scope link

valid\_lft forever preferred\_lft forever

### 配置的另一种方式

auto vlan2

iface vlan2 inet static

address 192.168.2.1

netmask 255.255.255.0

vlan-raw-device eth0

不提倡用这种方式。