

Linux

Linux VLANs

Andy Fletcher

andy@x31.com

TLNX001.1

2011/07/31



This presentation by [Andy Fletcher](#) is licensed under a Creative Commons [Attribution-NonCommercial-ShareAlike 3.0 Unported License](#).

Introduction

VLANS (Virtual LANs) are a way of running multiple networks over Ethernet from one physical interface. These can be linked to specific ports on a switch thereby expanding the ports on a router.

Traffic in different VLANS is completely separated and allows for control of traffic between different services on a network. The only thing to be careful of is that traffic on one VLAN can saturate the common bearer (trunk) resulting in degradation of performance for the other VLANS. This can be avoided by applying Linux Traffic Control (TC) and rate limiting the traffic on the different VLANS.

Traffic can be routed between VLANS in the same way as any other network interface.

Linux supports VLANS in the kernel by using the 8021q module. Once this is installed VLANS can be created at will on Ethernet interfaces. This presentation considers only the “Debian Way”

Ethernet Frame

VLANs are standard Ethernet frames which have an additional header of 4 bytes inserted after the two MAC headers. The first 16 bits contain the value 0x8100 and indicate that the frame is a VLAN. The following 16 bits are split into three fields as below:

Priority Code Point (PCP). This three bit field indicates the priority of the packet.

Canonical Format Indicator (CFI). This single bit field indicates how the MAC is to be interpreted. Always set to 0 in normal Ethernet networks.

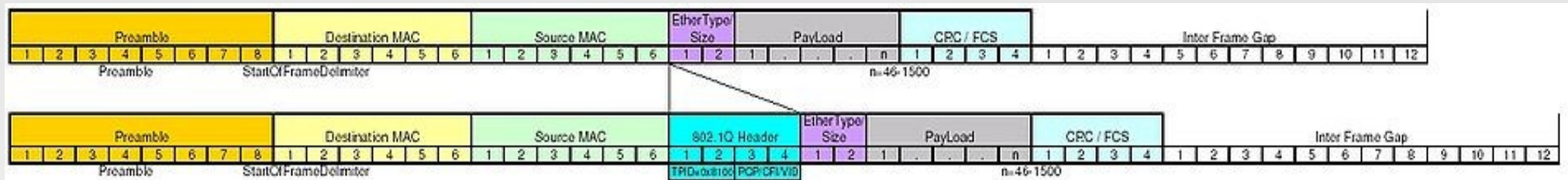
VLAN Identifier (VID). This 12 bit field contains the VLAN number the packet is associated with. Numbers 0, 1, 1001-1005 and 4095 are normally reserved and should be avoided.

Linux

VLANs

Ethernet Frame

The diagram below shows a normal Ethernet frame and one containing a packet belonging to a VLAN.



Note:
Image taken from Wikipedia

Configuring VLANs – Cisco

The code segment below shows a Cisco switch configuration with the first port acting as a VLAN trunk, the second port having VLAN 10 and the third operation on VLAN 20.

```
interface GigabitEthernet1/0/1
description Trunk to router
switchport trunk encapsulation dot1q
switchport mode trunk
```

```
interface GigabitEthernet1/0/2
description VLAN 10
switchport access vlan 10
switchport mode access
spanning-tree portfast
```

```
interface GigabitEthernet1/0/3
description VLAN 20
switchport access vlan 20
switchport mode access
spanning-tree portfast
```

Configuring VLANs – Cisco

The following additional commands should be considered when configuring Cisco switch ports.

Trunk ports

```
switchport trunk allowed vlan 10,20  
no cdp enable  
duplex full  
speed 1000  
no mdix auto
```

Non Trunk ports

```
spanning-tree portfast  
no cdp enable  
duplex full  
speed 1000  
no mdix auto
```

Preparing Linux for VLANs

Install the Debian VLAN utilities..

```
apt-get install vlan
```

Add the VLAN module to /etc/modules ..

```
# /etc/modules: kernel modules to load at boot time.
```

```
#
```

```
# This file contains the names of kernel modules that should be loaded
```

```
# at boot time, one per line. Lines beginning with "#" are ignored.
```

```
# Parameters can be specified after the module name.
```

```
8021q
```

```
loop
```

Manually insert the VLAN module ..

```
modprobe 8021q
```

Configuring VLANs

In Debian VLANs can be defined using the configuration file `/etc/network/interfaces`. If they are done there then the normal networking commands can be used to start and stop the VLANs. The example below shows a single VLAN being created on `eth0`. The name of the VLAN interface can be almost anything you want like `vlan10`, `eth0.10` etc.

`#RAW` interface (native VLAN) has no address and is not used for traffic in this case. If a native `# VLAN` is used then configure this as a normal interface. Make sure this interface is up or the `# VLANs` will not work.

```
auto eth0
iface eth0 inet manual
    up ifconfig eth0 0.0.0.0 up
```

```
# VLAN 10
auto vlan10
iface vlan10 inet static
    address 10.122.10.20
    netmask 255.255.255.0
    network 10.122.10.0
    broadcast 10.122.10.255
    vlan-raw-device eth0
```

Configuring VLANs

You can cause routes to be added to the networking by adding 'up' lines at the end of the interface configuration. Scripts can also be run using the same method. The example below shows a VLAN being configured on a bond interface with a static route and a script (traffic shaping) being run when the interface is up.

```
auto vlan12
iface vlan12 inet static
    address 10.122.12.1
    netmask 255.255.255.0
    network 10.122.12.0
    broadcast 10.122.12.255
    vlan-raw-device bond0
    up ip route add 10.122.41.0/24 via 10.122.12.2
    up /etc/network/htb/htb-vlan20.sh
```

Enabling and disabling interfaces

You can use the normal ifup and ifdown commands to manually enable and disable interfaces.

Enabling an interface

```
root@x31em:/usr/local/bin# ifup vlan10
Set name-type for VLAN subsystem. Should be visible in /proc/net/vlan/config
Added VLAN with VID == 10 to IF -:eth0:-
root@x31em:/usr/local/bin#
```

Disabling an interface

```
root@x31em:/usr/local/bin# ifdown vlan10
Removed VLAN -:vlan10:-
root@x31em:/usr/local/bin#
```

Directly adding VLANs

Instead of using the interfaces file you can use the commands vconfig and ifconfig. Note the default naming for the interface is the parent interface with a dot and the VLAN number appended.

```
vconfig add eth0 10
ifconfig eth0.10 up
ifconfig eth0.10 10.122.10.1 netmask 255.255.255.0
```

Directly adding VLANs using only ip link commands

You can also add VLANs using the ip link commands as in the two examples below.

```
ip link add link eth0 name eth0.3 type vlan id 3
ip addr add 10.122.3.1/24 dev eth0.3
ip link set eth0.3 up
```

```
ip link add link eth0 name vlan20 type vlan id 20
ip addr add 10.122.20.1/24 dev vlan20
ip link set vlan20 up
```

Routing between VLANs

You can route between VLANs in the same way as any other interface.

If you are configuring routing between interfaces remember to enable IPv4 forwarding in the kernel or nothing will work.

To see if forwarding is enabled use the following command.

```
cat /proc/sys/net/ipv4/ip_forward
```

If you get 0 then forwarding is disabled, 1 means it is enabled

To enable forwarding on a running system just echo the number 1 to the proc filesystem value as below

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

To enable forwarding at boot time edit `/etc/sysctl.conf` and ensure the following line is set in the file

```
net.ipv4.ip_forward=1
```

Linux

VLANs

Questions?