VLANs

IEEE 802.1Q IEEE 802.1Q is the networking standard that supports Virtual LANs (VLANs) on an Ethernet network (Layer 3 networking).

The advantages of a VLAN are:

- increased switching efficiency
- network segmentation
- security
- trunks
- ability to sub-divide a LAN for security purposes

In order to use VLAN's you need:

1. a switch that supports IEEE 802.1Q
2. a NIC (Network Interface Card) that supports 802.1Q (Not all network drivers support VLAN)
3. 802.1Q support in the kernel (8021q module, CONFIG_VLAN_8021Q)

VLANs work by applying a **tag** to each frame (which increases the header size by 4 bytes). The tag contains an 'ID' and a 'prority'. The priority can be used for a quality of service (QoS) scheme known IEEE 802.1p.

Only portions of a network which are VLAN-aware (802.1Q compliant) can include VLAN tags - traffic on other segments (802.1D conformant) will not contain tags. When a frame enters the VLAN-aware portion a tag is added to denote the VLAN info (membership etc).

In general, you add a VLAN tag to a network port (on a managed switch, for example), and that switch port will drop all packets that don't have the specified VLAN ID, while allowing those with the appropriate VLAN ID to continue on.

Here is an image showcasing a potential use case for VLANs:

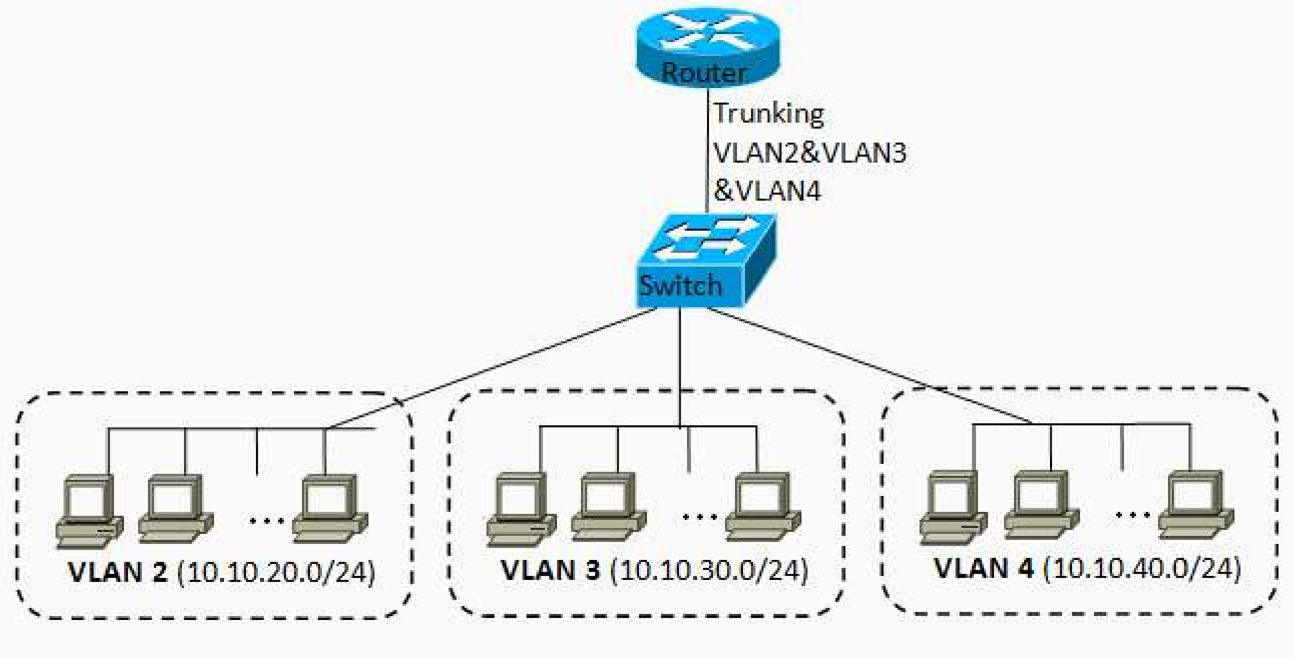

Figure 12.5. 802.1Q trunk between the router and the switch

Generally speaking, VLAN tagging is done on a switch basis, though you can also use Linux to listen to VLAN tagged packets on interface devices.

Linux VLAN support

The Linux network layer supports VLAN if built with CONFIG_VLAN_8021Q. To create a VLAN you need to slave it off a physical interface port. Note that the physical interface will still accept 'un-tagged' traffic, but only traffic matching the VLAN ID will be presented to the virtual interface.

The Linux `ip` utility from the ip-route2 package allows the manipulation of the network stack on the Layer 3 network layer.

Examples:

- use `ip link add` to create VLAN ID 5 slaved off physical interface eth0:

  ```
  ip link add link eth0 name eth0.5 type vlan id 5
  ifconfig eth0.5 192.168.1.100 up
  ```

  - adds a 'link' object to the `eth0` nic with a name of `eth0.5` (the name can be anything but it is convention to use the <physical>.<id>)
  - any packets leaving `eth0.5` will be tagged with VLAN ID #5?
  - only packets coming into `eth0` tagged with VLAN ID #5? will be presented to `eth0.5`
- use `ip link set` to re-configure the VLAN:

- use `ip link show` to show the sate of a VLAN:

```
$ ip -d link show eth0.5
5: eth0.1@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP mode DEFAULT group default
    link/ether f8:bc:12:8c:23:21 brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 5 <REORDER_HDR>
```

  - `eth0.1@eth0` is using the IEEE 802.1Q standard with VLAN id 42.
- use tcpdump to show information when you receive a packet on this nic with the VLAN ID of 5:

```
tcpdump -i eth0 -Uw - | tcpdump -en -r - vlan 5
```

- delete a VLAN interface:

```
ifconfig eth0 down
ip link delete eth0.5
```

See ip(8) or here for more info

Alternatively the Linux `vconfig` (which is a bit older and deprecated) can be used as well.

Examples:

- create VLAN ID 5 slaved off physical interface eth0:

```
vconfig add eth0 5
ifconfig eth0.5 192.168.1.100 up
```

- show info about a VLAN ID (older kernels only):

```
# cat /proc/net/vlan/config
VLAN Dev name    | VLAN ID
Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
eth0.5         | 5  | eth0
# cat /proc/net/vlan/eth0.5
eth0.5  VID: 5   REORDER_HDR: 1  dev->priv_flags: 1
         total frames received            4
          total bytes received          252
      Broadcast/Multicast Rcvd            0

      total frames transmitted            8
       total bytes transmitted          688
Device: eth0
INGRESS priority mappings: 0:0  1:0  2:0  3:0  4:0  5:0  6:0 7:0
 EGRESS priority mappings:
```

- remove VLAN ID 5

```
vconfig rem eth0.5
```

See vconfig(8) for more info