12 Virtualisation

12.1 Goal of this section

The goal of this section is for the reader to understand the concepts of virtualisation as they relate to networking, and specifically how virtual local access networks (VLANs) work within Ethernet Layer 2 and virtual private networks (VPNs) work within IP. We'll also briefly touch on Virtual Machines and Containers, but those do really not form part of a computer *networks* course.

12.2 Problem

The course so far covers how to build a network and connect things that you want to communicate – and everything on the network can then communicate with everything else. If you want to allow different sets of endpoints to communicate with each other, but not between the sets, then you build different networks.

For two examples:

- Consider a hospital, where you probably want a network for staff to communicate, a network for patients to connect out to the internet, a network for medical devices to communicate logistic information, etc.
- Consider a university with campuses across a city or different cities.

You *could* build physically separate networks (and if you're an organisation with sufficient paranoia or requirements, you might well do that!) but it would be much cheaper to build one network and partition it up somehow.

In particular, that partitioning needs to work such that that all of the endpoints on all of the networks just think they're on a normal network and simply cannot see the other network's traffic or endpoints.

12.3 Layer 2 – Virtual LANs (VLANs) layered on one physical LAN.

12.3.1 Abstract

We have one physical ethernet network with one physical set of cables and switches. The physical ports on the switches where hosts are connected are each associated (by administrator configuration) with a particular group ID (called a VLAN ID)

All frames arriving at the switch over that port are tagged with the VLAN ID, and flow around in the network with that VLAN ID.

Only frames with the VLAN ID associated with the port are sent out that port (others are ignored by that port) and the VLAN ID tagging is removed before they're sent out the port.

Hosts are therefore assigned into the group by the physical port that they're plugged into, all traffic from them is tagged by the switch, and the switch filters out all other traffic. The host never sees the tagging and is unaware that it is not plugged into a "normal" LAN.

12.3.2 Ethernet VLAN tagging (802.1Q)

To squeeze this additional VLAN tagging information into the network, we use a modified Ethernet header, that is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Destination MAC...                      |
+-------------------------------+-------------------------------+
|         ...Dest MAC           |        Source MAC...          |
+-------------------------------+-------------------------------+
|                         ...Source MAC                         |
+-------------------------------+-------------------------------+
|            0x8100             |  Ctrl |       Label           |
+-------------------------------+-------------------------------+
|     Ether Type / Size         |          Payload....          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

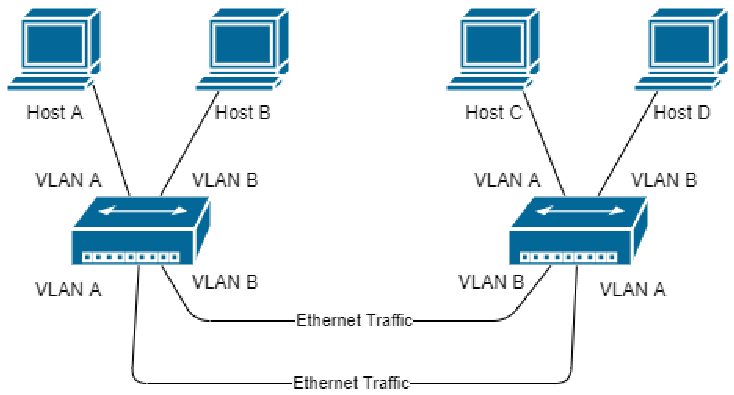
Unpacking this header…

- **Destination MAC address** (6 bytes) As before.
- **Source MAC address** (6 bytes) As before.
- **Protocol ID / Length** (2 bytes). This is the length field in the Ethernet header. The maximum length of an ethernet frame is 1500, so this field can double as aa protocol ID field. Any value above 1500 indicates another protocol (values assigned by IANA). The protocol ID for 802.1Q is **0x8100**, indicating this is an 802.1Q header, and the following fields should be interpreted as per 802.1Q.
    - In this case, this means there is an additional 2 bytes inserted next:
- **Priority** (3 bits) & **Drop Eligibility Indicator** (1 bit). These are used to provide prioritisation – we won't cover these in this course.
- **VLAN ID** (12 bits). This is a 12-bit number that is the VLAN ID – the tag that indicates which virtual LAN
- **Length** (2 bytes) From this point the Ethernet header continues as before with the Length field…

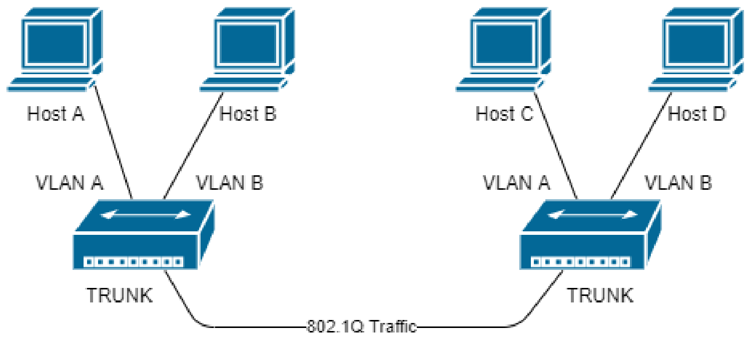
12.3.3 Trunking

On a single VLAN-aware switch, this is straightforward. We configure each port with a VLAN ID. Incoming frames are tweaked to insert the new header, frames are broadcast only out of interfaces that match the VLAN ID, and those outgoing frames are tweaked back to the normal Ethernet header. It's a bit heavy-weight (you don't need the new header to travel on any wires!) but it just works. Now consider the case with two switches:

You can make this work, as long as you have one cable between each switch for each VLAN ID that you're using, as per the picture below. Note that this still does not make use of the 802.1Q header – all traffic on all cables is regular Ethernet traffic.



An Administrator can also configure switch ports to be trunking ports. Trunking ports do not add or remove VLAN tags, and distribute the 802.1Q traffic directly, as shown below.



12.3.4 Example Flows

12_8021Q.pcap shows VLAN tagged traffic flowing over a VLAN trunk link. How many different VLAN IDs are there?

12.3.5 Switching and Routing

Switching

VLAN aware switches are running several separate VLAN networks over one physical LAN. This means that they need to run all of the layer 2 protocols that they were running once *per VLAN.* For example, instead of STP, they now run MSTP (Multiple Spanning Tree Protocol), which takes advantage of the separation of VLANs to optimise the spanning trees on each separate VLAN to provide more efficient connectivity than just creating a single STP and using that for all VLANs.

The exact details of how this works are outside the scope of this course.

Routing

As with a physical LAN, when looking at layer 3, each VLAN has an associated subnet and routers to allow IP routing. The router must have an interface onto the VLAN, and so for most LANs that are split into VLANs ,the default router is VLAN aware and runs a trunking interface, on which it has multiple IP addresses (one for each VLAN).

Note that if your VLANs are using private IP address spaces and you do not have routing between them, then it is possible to use the same IP subnet on different VLANs on the same LAN. You can even use the same IP addresses on each VLAN, as they're kept separate by the VLAN tagging. **Do not do this**. One mis-plugged cable and you will be in for a world of network debugging pain.

In real life, it is best practice to keep separate subnets for separate (V)LANs and have each connected to your common router with an interface on each (V)LAN.

12.4 Layer 3 - Virtual private networks (VPNs) layered on one IP network.

VLANs allow multiple LANs to coexist. Scaling up, we need an equivalent solution for our IP networks. As for VLANs, we need IP endpoints to be unaware that they're within a VPN. In fact, if you consider the example case of a company or university with a number of sites, we need each IP networked site to be internally unaware that it is in a VPN, and just be running normal IP routing.

Critically, this means that any service that is providing virtual private networking needs to not only be able to pass around different sets of data for different VPNs (keeping it separate) – it also needs to be able to pass around different sets of routing information for different VPNs (and keep that separate). This makes VPNs more complex than VLANs, where it is just data that we have to separate.

12.4.1 Abstract

IP endpoints and routers within individual private networks (sites) are unaware that they're in a VPN and perform IP routing as normal. At the edge of each site is a VPN aware router that performs normal IP routing within the site and VPN aware routing across the public internet to other sites (in a similar manner to L2 VLAN aware switches having trunk links).

VPN aware routers that are at the edge of the sites are called edge routers. There are also VPN aware routers in the central internet that handle the VPN routes but don't have to directly handle non-VPN traffic – called core routers.

Note that the VPN aware routers (both edge and core) need to be able to safely distribute both the IP routing information *and* the data traffic across the core network, so that the IP routers in the sites can run their own IP routing without being aware that they are communicating across the wider internet.

As with L2 VLANs, this is achieved by tagging both routes and traffic to associate them with the VPN. Edge routers must maintain a routing table per VPN in addition to handling untagged core traffic.

12.4.2 BGP MPLS VPNs (RFC 4364)

(That's Border Gateway Protocol Multi-Protocol Label Switching Virtual Private Networks.)

We can build the route tagging and data tagging described above using technologies that we've already covered in the course – Border Gateway Protocol for distributing tagged routes, and MPLS tunnels to allow tagged data traffic.

Routing

Within each VPN site, routing happens as normal, although the BGP routers at the edge of the VPN sites and in the core have extra work to do to get the routing information across the core.

Two BGP concepts covered in the section 8 on internet routing help us here: **NLRIs** and **Attributes**.

- BGP provides global routing, providing next hops to Network Layer Reachability Information (NLRIs). When we encountered BGP previously, these NRLIs were simple IP addresses, but BGP allows other NLRIs too. For this use case, BGP uses VPN-IPv4 NLRIs, which are a standard IPv4 NLRI with an additional 8-byte route-distinguisher. This allows different VPNs to use the same IP addresses and for BGP in the core to be able to still distinguish between routes to those addresses without confusing them.

- BGP routes include attributes associated with NLRIs that are passed around. For this use case, BGP adds a Route Target attribute to VPN routes – which again is a simple 8-byte tag that marks the route as being for a particular VPN.

Routing works as follows:

- The BGP edge router receives routes from inside the VPN site and stores them in a per-VPN routing table. It adds a route target to each route indicating which VPN it has come from, adds the route distinguisher to the NLRI to turn it into a VPN route – and advertises this on to its peers.
  Note that an edge router could have multiple different VPN sites attached. It can distinguish which VPN a route has come from by which IGP router it has arrived from (in a similar manner to an L2 switch tagging packets depending on which port they came in over) and maintains a per-VPN routing table for each. In this case, the edge router has multiple per-VPN routing tables (which could even have routes to the same IP destinations in different VPNs). These routing tables are completely separate and work independently.
- The router *also* sets up an MPLS label to turn the next hop into a labelled next hop (needed for distinguishing traffic – see below). This builds up a set of LSPs across the BGP routers in the core network, between the VPN sites.
- Other edge routers that receive the VPN routes check the route target of incoming routes against their own local VPNs and if the route target matches, they remove the route target and advertise the route into that local site (again similar to a switch removing the VLAN tag if the port matches). When adding the route to their own per-VPN routing table, they use the labelled next hop for the LSP across the core.

Traffic

Within a site, the traffic is forwarded as normal following routing rules within the site. Traffic that arrives at the BGP edge router is routed using the per-VPN forwarding table for the VPN associated with the physical port that the traffic arrived over. Traffic that needs to be routed to another site is label-tagged and sent down the LSP across the core network. The edge router where the traffic departs the core removes the label and routes the traffic using its routing table for its VPN site. (In the case of multiple sites, that router has multiple LSPs with different labels for the different sites, so can distinguish traffic bound for each VPN and use the correct per-VPN forwarding table).

A note on labels

BGP VPN routes (which I've called tagged routes above) are often called labelled routes. MPLS labelled traffic is labelled traffic. These are different sets of labels in different label spaces. One set of labels distinguishes routes used in different VPNs and one set distinguishes traffic bound for different VPNs. There is no need for them to match (and in fact they typically do not match).

12.5 Pseudowires

Rather than offering VPN IP routing, a service provider *could* offer to provide what looks to each site like an Ethernet cable directly between the sites (called a Pseudowire), allowing the sites to run as a single larger LAN. By encapsulating traffic arriving on a particular physical port on the edge router with MPLS labels *outside* the ethernet headers (and removing those labels on the far side), the connection across the core appears to be "just" a physical wire over which ethernet traffic can flow.

There is another suite of protocols to handle this that we won't cover in this course.

12.6 Virtualising hosts and networks

This is a networking course, not a computer architecture or virtualisation course – so we will not cover virtualisation itself. Rather, the key point to take away from this section is that handling the networking is a critical part of being able to make your hosts virtual – and that the virtualisation architecture needs to handle this in addition to just handling the virtual-physical resource mapping on virtual hosts.

12.6.1 VMs

Virtual Machines run on a Hypervisor – the software that presents each VM with a view of the hardware and juggles the physical resources around to allow each VM to believe that it really does own the hardware it is using.
Routing and switching between virtual machines is done using virtual switches, which are also just software running on the same physical host.

Virtual switches are configured in the hypervisor like you would a VM, but the Hypervisor *knows* all the IPs and MACs of all the machines on the switch and knows the complete “physical” topology of the LAN at all times (it’s just software and configuration), and so typically delivers packets/frames directly between VMs, with no need for MAC learning and L2 routing protocols within the physical host.

Note, however, that to connect out of the physical host and into a real network, the virtual switch does need to perform MAC learning and run L2 protocols to be part of the wider real LAN.

12.6.2 Containers

A container is an executable package of code that includes everything needed to run an application, including system libraries and settings. The container manager needs to be able to network (switch or route) packets in and out of containers.

Linux provides the concept of a network namespace, and most container networking solutions are built on this. Network name spaces work conceptually in the same way as the per-VPN routing tables of a BGP MPLS VPN edge router. They operate completely independently, with traffic assigned to the relevant namespace based on the container it has come from.

Because of the potential for scale for containers, container networking is usually IPv6.