- Multilayer Switching -

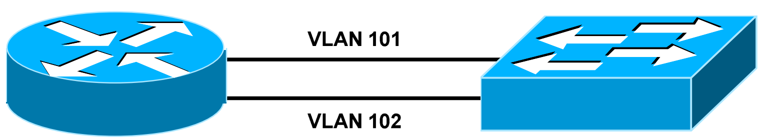
Routing Between VLANs

By default, a switch will forward both broadcasts and multicasts out *every* port but the originating port. However, a switch can be *logically* segmented into separate broadcast domains, using **Virtual LANs** (or **VLANs**).

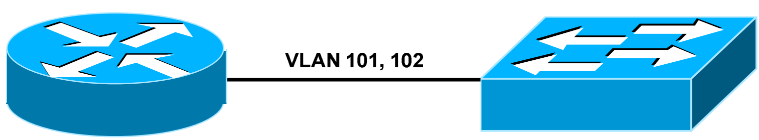
Each VLAN represents a unique broadcast domain:

- Traffic between devices within the *same* VLAN is switched.
- Traffic between devices in *different* VLANs requires a Layer-3 device to communicate.

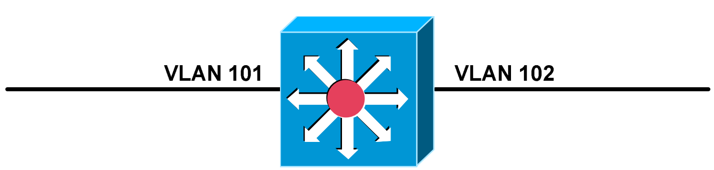
There are *three* methods of routing between VLANs. The first method involves using an **external router** with a separate physical interface **in each VLAN.** This is the *least scalable* solution, and impractical for environments with a large number of VLANs:



The second method involves using an **external router** with a single **trunk link** to the switch, over which all VLANs can be routed. The router must support either 802.1Q or ISL encapsulation. This method is known as **router-on-a-stick:**



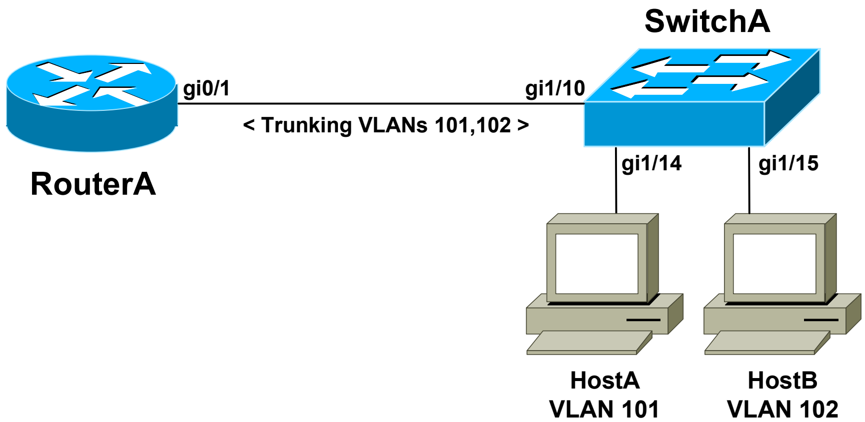
The final method involves using a **multilayer switch,** which supports both Layer-2 and Layer-3 forwarding:



Multilayer switching is a generic term, encompassing any switch that can forward traffic at layers higher than Layer-2.

<u>Configuring Router-on-a-Stick</u>

Consider the following router-on-a-stick example:



Four elements must be considered in this scenario:

- Interface gi1/10 on SwitchA must be configured as a **trunk port**.
- Interfaces gi1/14 and gi1/15 on SwitchB must be assigned to their specified VLANs.
- Interface gi0/1 on RouterA must be split into **subinterfaces,** one for each VLAN.
- Each subinterface must support the **encapsulation protocol** used by the trunk port on SwitchA.

Configuration on **SwitchA** would be as follows:

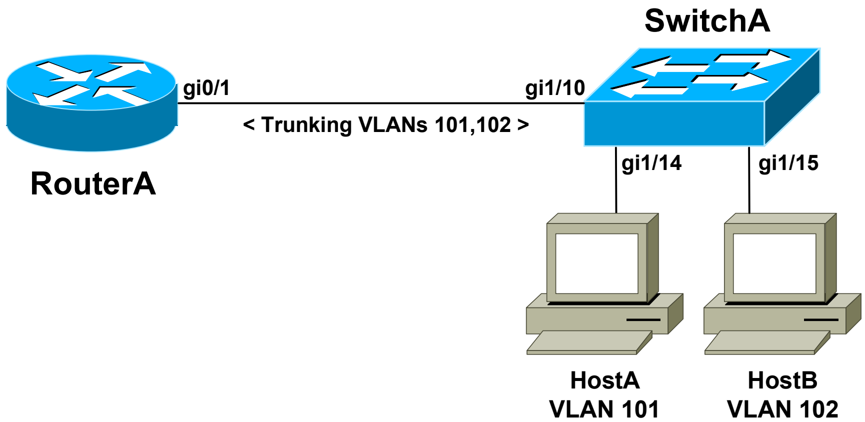
SwitchA(config)# *interface gi1/10*
SwitchA(config-if)# *switchport mode trunk*
SwitchA(config-if)# *switchport trunk encapsulation dot1q*
SwitchA(config-if)# *no shut*

SwitchA(config)# *interface gi1/14*
SwitchA(config-if)# *switchport mode access*
SwitchA(config-if)# *switchport access vlan 101*
SwitchA(config-if)# *no shut*

SwitchA(config)# *interface gi1/15*
SwitchA(config-if)# *switchport mode access*
SwitchA(config-if)# *switchport access vlan 102*
SwitchA(config-if)# *no shut*

Note that no Layer-3 information was configured on SwitchA.

Configuring Router-on-a-Stick (continued)



Configuration on RouterA would be as follows:

RouterA(config)# *interface gi0/1*
RouterA(config-if)# *no shut*

RouterA(config)# *interface gi0/1.101*
RouterA(config-subif)# *encapsulation dot1q 101*
RouterA(config-subif)# *ip address 10.101.101.1 255.255.255.0*

RouterA(config)# *interface gi0/1.102*
RouterA(config-subif)# *encapsulation dot1q 102*
RouterA(config-subif)# *ip address 10.102.102.1 255.255.255.0*

Each subinterface was configured with *dot1q* encapsulation, to match the configuration of SwitchA's trunk port.

Frames sent across the trunk port will be **tagged** with the VLAN ID. The number after each *encapsulation dot1q* command represents this VLAN ID. Otherwise, the router could not interpret the VLAN tag.

The IP address on each subinterface represents the **gateway** for each VLAN:

- HostA is in VLAN 101, and will use 10.101.101.1 as its gateway.
- HostB is in VLAN 102, and will use 10.102.102.1 as its gateway.
- HostA and HostB should now have full Layer-3 connectivity.

There are inherent disadvantages to router-on-a-stick:

- All routed traffic will share the same physical router interface, which represents a bottleneck.
- ISL and DOT1Q encapsulation puts an increased load on the router processor.

Multilayer Switch Port Types

Multilayer switches support both Layer-2 and Layer-3 forwarding.

Layer-2 forwarding, usually referred to as *switching*, involves decisions based on frame or *data-link* headers. Switches will build hardware address tables to intelligently forward frames.

Layer-3 forwarding, usually referred to as *routing*, involves decisions based on packet or *network* headers. Routers build routing tables to forward packets from one network to another.

A multilayer switch supports three **port types:**

- **Layer-2** or **switchports**
- **Layer-3** or **routed ports**
- **Switched Virtual Interfaces (SVIs)**

A **switchport** can either be an *access* or *trunk* port. By default on Cisco switches, all interfaces are switchports. To *manually* configure an interface as a switchport:

Switch(config)# *interface gi1/10*
Switch(config-if)# *switchport*

A **routed port** behaves exactly like a physical router interface, and is not associated with a VLAN. The *no switchport* command configures an interface as a routed port, allowing an IP address to be assigned:

Switch(config)# *interface gi1/20*
Switch(config-if)# *no switchport*
Switch(config-if)# *ip address 10.101.101.1 255.255.255.0*

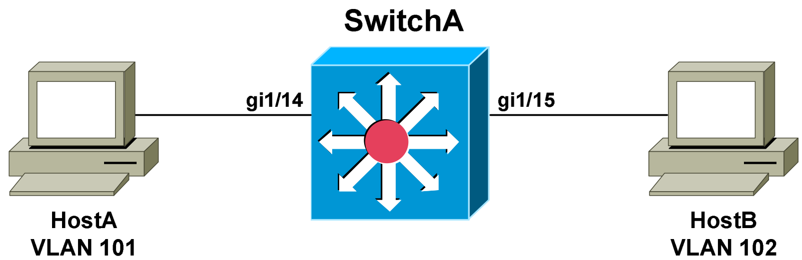
Multilayer switches support configuring a VLAN as a *logical* routed interface, known as a **Switched Virtual Interface (SVI).** The SVI is referenced by the VLAN number:

Switch(config)# *interface vlan 101*
Switch(config-if)# *ip address 10.101.101.1 255.255.255.0*
Switch(config-if)# *no shut*

SVIs are the most common method of configuring inter-VLAN routing. The logical VLAN interface will not become online unless:

- The VLAN is **created.**
- At least **one port is active** in the VLAN.

Configuring Inter-VLAN Routing Using SVIs



Configuring inter-VLAN routing using SVIs is very simple. First, the VLANs must be created:

SwitchA(config)# *vlan 101*
SwitchA(config-vlan)# *name VLAN101*

SwitchA(config)# *vlan 102*
SwitchA(config-vlan)# *name VLAN102*

Layer-3 forwarding must then be enabled globally on the multilayer switch:

SwitchA(config)# *ip routing*

Finally, each VLAN SVI must be assigned an IP address:

SwitchA(config)# *interface vlan 101*
SwitchA(config-if)# *ip address 10.101.101.1 255.255.255.0*
SwitchA(config-if)# *no shut*

SwitchA(config)# *interface vlan 102*
SwitchA(config-if)# *ip address 10.102.102.1 255.255.255.0*
SwitchA(config-if)# *no shut*

The IP address on each SVI represents the **gateway** for hosts on each VLAN. The two networks will be added to the routing table as **directly connected routes.**

Remember: an SVI requires at least one port to be active in the VLAN:

SwitchA(config)# *interface gi1/14*
SwitchA(config-if)# *switchport mode access*
SwitchA(config-if)# *switchport access vlan 101*
SwitchA(config-if)# *no shut*

SwitchA(config)# *interface gi1/15*
SwitchA(config-if)# *switchport mode access*
SwitchA(config-if)# *switchport access vlan 102*
SwitchA(config-if)# *no shut*

Multilayer Switching – Route Once, Switch Many

Originally, multilayer switches consisted of two independent components:

- Routing engine
- Switching engine

The first packet in an IP traffic flow must be sent to the routing engine to be *routed*. The switching engine could then **cache** this traffic flow. Subsequent packets destined for that flow could then be *switched* instead of routed. This greatly reduced forwarding latency.

This concept is often referred to as **route *once*, switch *many*.**

Just like a router, a multilayer switch must update the following header information:

- **Layer 2 destination address**
- **Layer 2 source address**
- **Layer 3 IP Time-to-Live (TTL)**

Additionally, the Layer-2 and Layer-3 checksums must be updated to reflect the changes in header information.

Cisco's original implementation of multilayer switching was known as **NetFlow** or **route-cache switching**. NetFlow incorporated separate routing and switching engines.

NetFlow was eventually replaced with **Cisco Express Forwarding (CEF),** which addressed some of the disadvantages of NetFlow:

- CEF is less CPU intensive.
- CEF does not dynamically cache routes, eliminating the risk of stale routes in the cache if the routing topology changes.

Cisco Express Forwarding (CEF)

CEF consists of two basic components:

- **Layer-3 Engine**
- **Layer-3 Forwarding Engine –** *Switches* data based on the FIB.

The **Layer-3 Engine** is responsible for building the routing table and then *routing* traffic. The routing table can be built either *statically* or *dynamically*, via a routing protocol.

Each entry in the routing table will contain the following:

- Destination prefix
- Destination mask
- Layer-3 next-hop address

CEF reorganizes the routing table into a more efficient table called the **Forward Information Base (FIB)**. The *most specific routes* are placed at the *top* of the FIB, to accelerate forwarding lookups. Any change to the routing table is immediately reflected in the FIB.

The **Layer-3 Forwarding Engine** uses the FIB to *switch* traffic in hardware, which incurs less latency than *routing* it through the Layer-3 Engine. If a packet cannot be switched using the Forwarding Engine, it will be **punted** back to the Layer-3 Engine to be routed.

The FIB contains the **Layer-3 next-hop** for every destination network. CEF will additionally build an **Adjacency Table**, containing the **Layer-2 address** for each next-hop in the FIB. This eliminates the latency from ARP requests when forwarding traffic to the next-hop address.

If there is no next-hop entry in the Adjacency Table, a packet must be sent to the Layer-3 Engine to **glean** the next-hop's Layer-2 address, via an ARP request.

CEF is **enabled by default** on most Cisco multilayer switch platforms. In fact, it is impossible to disable CEF on many platforms.

To manually enable CEF, when necessary:

Switch(config)# *ip cef*

Cisco Express Forwarding (CEF) (continued)

CEF can be disabled on a per-interface basis on some platforms. Depending on the model, the syntax will be different:

Switch(config)# *interface gi1/15*
Switch(config-if)# *no ip route-cache cef*

Switch(config-if)# *no ip cef*

To view entries in the FIB table:

Switch# *show ip cef*

```
Prefix              Next Hop     Interface

172.16.1.0/24       10.50.1.1    Vlan50
172.16.2.0/24       10.50.1.2    Vlan50
172.16.0.0/16       10.50.1.2    Vlan50
0.0.0.0/0           10.10.1.1    Vlan10
```

The most specific entries are installed at the top of the table. Note that each FIB entry contains the following information:

- The destination prefix
- The destination mask
- The next-hop address
- The local interface where the next-hop exists

To view the Adjacency Table:

Switch# *show adjacency*

```
Protocol     Interface    Address
IP           Vlan50       10.50.1.1(6)
                          4 packets, 1337 bytes
                          0001234567891112abcdef120800
                          ARP 01:42:69

Protocol     Interface    Address
IP           Vlan50       10.50.1.2(6)
                          69 packets, 42012 bytes
                          000C765412421112abcdef120800
                          ARP 01:42:69
```