- Switching Tables -

Layer-2 Forwarding Overview

Layer-2 devices build **hardware address tables,** which at a minimum contain the following:

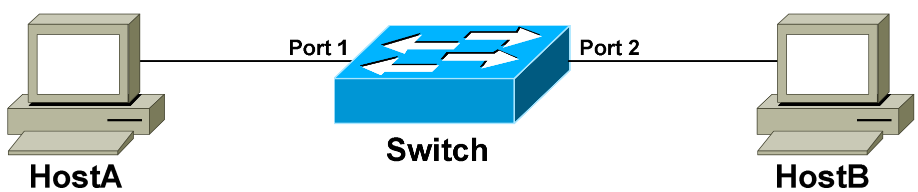
- Hardware addresses for hosts (such as Ethernet MAC addresses)
- The port each hardware address is associated with

Using this information, Layer-2 devices will make intelligent **forwarding** decisions based on the frame (or data-link) header. A frame can then be forwarded out *only* the appropriate destination port, instead of *all* ports.

Layer-2 forwarding was originally referred to as **bridging.** Bridging is a largely deprecated term (mostly for marketing purposes), and Layer-2 forwarding is now commonly referred to as **switching.**

Switching Queues

Layer-2 switches utilize **queues** to store incoming and outgoing frames. Consider the following diagram:



1. The switch receives a frame on Port 1, from HostA destined for HostB.
2. The frame is placed in Port 1's **ingress queue.**
3. The switch performs a lookup on the destination hardware address - HostB in this example.
4. The switch determines that the appropriate destination port for HostB is Port 2.
5. The frame is placed in Port 2's **egress queue.**

If the switch had no knowledge of HostB's hardware address, the frame would be placed in the egress queue **of all ports** except for the originating port**,** and thus flooded to the entire network.

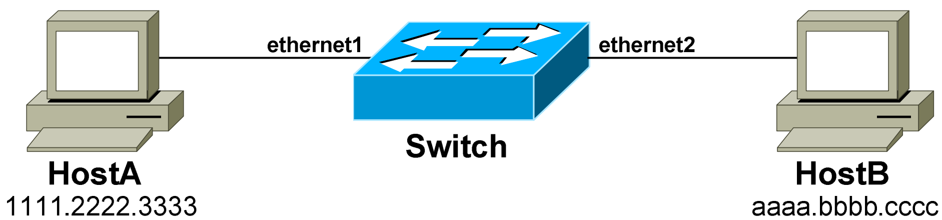
A port can contain multiple ingress or egress queues. This allows critical traffic to be prioritized over less important traffic.

MAC Address Table

In the previous example, the switch made a **forwarding decision** based on the destination host's hardware address. The switch essentially forwarded the frame from one port's *ingress* queue to another port's *egress* queue.

To perform this forwarding decision, a switch consults its hardware address table. For Ethernet switches, this is referred to as the **MAC address table**, or the Layer-2 forwarding table.

When a switch is first powered on, the MAC address table will be empty. The switch will build the table through a dynamic learning process, by observing the **source MAC address** of frames:



1. Initially, the switch will have no knowledge of the MAC addresses of HostA and HostB.
2. When HostA sends a frame to HostB, the switch will add *HostA's* MAC address to its table, associating it with port e*thernet1*.
3. The switch will not learn HostB's MAC address until HostB sends a frame back to HostA, or to any other host connected to the switch.
4. HostB's MAC address will then be associated with port *ethernet2*.

Remember: a switch will only add MAC address table entries based on the **source** MAC address in a frame.

The MAC address table is stored in fast volatile memory**,** allowing lookups to be performed very quickly. However, this also results in dynamically-learned MAC addresses being lost if the switch is rebooted or powered off.

Stale (or *idle*) entries in the table will be **aged out.** By default on Cisco switches, idle entries will be purged after 300 seconds.

Most switches support **statically** configuring MAC addresses into the table, which will survive a reboot or power failure, and never be purged. Statically configuring entries in the table is only required in limited circumstances.

CAM and TCAM Tables

On Cisco switches, the MAC address table is stored in **Content Addressable Memory (CAM).**

CAM differs from the more prevalent Random Access Memory (RAM):

- RAM queries a specific *memory address***,** and then returns the *data* or *content* stored at that address location.
- CAM operates essentially in the reverse, and does not require that a memory address be provided. Instead, CAM queries for the desired *content,* and then returns all matching results, including any *associated* content.

CAM is *significantly* faster than RAM, as it searches the entire memory content in one cycle, instead of a single address at a time. However, CAM is more expensive than RAM.

When performing a MAC address table lookup, the MAC address itself is the *content* being queried. For any matching results, CAM will return the destination port (the *associated content).*

Cisco uses the terms *MAC address table* and *CAM table* interchangeably. This guide will use the term **CAM table** moving forward.

Idle entries in the CAM are purged after **300 seconds**, by default. This timer is reset every time a frame is received with the associated MAC address on the correct port.

If a host moves to a different port on a switch, the CAM table entry for the previous port will be **purged immediately**. This is desirable behavior - a MAC address is unique, and should never exist on more than one switch port unless a switching loop or other issue exists.

Ternary Content Addressable Memory (TCAM) tables provide high-speed lookups for two additional functions:

- Filtering traffic using access-lists
- Prioritizing traffic using QoS

TCAM tables are covered in greater detail later in this guide.

Multilayer switches utilize the **Forwarding Information Base (FIB)** table for L3 forwarding decisions. Multilayer switching is covered extensively in a different guide.

Managing the CAM Table

Each entry in the CAM table contains the following information:

- The MAC address
- The switch port the MAC address was learned on
- The VLAN of the switch port
- A time stamp, for the aging timer

To view the entire CAM table:

Switch# *show mac address-table*

```
 vlan    mac address     type      port
-------+---------------+---------+---------------------
   9    000c.291e.96f0   dynamic  GigabitEthernet1/1
   9    000c.293c.7cac   dynamic  GigabitEthernet1/1
   9    000c.2950.e3e9   dynamic  GigabitEthernet1/1
   9    000c.29ba.fe28   dynamic  GigabitEthernet1/2
   9    842b.2ba6.3a7d   dynamic  GigabitEthernet1/3
   9    d067.e50b.1975   dynamic  GigabitEthernet1/5
   9    d067.e51e.e35a   dynamic  GigabitEthernet2/1
   9    f04d.a2f6.d37b   dynamic  GigabitEthernet2/2
```

A single switch port can learn many addresses. In the above output, *GigabitEthernet1/1* has multiple MAC addresses associated with it. This usually indicates this is an uplink to another switch.

To view the CAM table entries for a specific port or MAC address:

Switch# *show mac address-table interface GigabitEthernet 1/5*

```
 vlan    mac address     type      port
-------+---------------+---------+---------------------
   9    d067.e50b.1975   dynamic  GigabitEthernet1/5
```

Switch# *show mac address-table address f04d.a2f6.d37b*

```
 vlan    mac address     type      port
-------+---------------+---------+---------------------
   9    f04d.a2f6.d37b   dynamic  GigabitEthernet2/2
```

The output of a command can be filtered using the **pipe** command. For example, to search for any entry that contains *3a7d* in the MAC address:

Switch# *show mac address-table | include 3a7d*

```
 vlan    mac address     type      port
-------+---------------+---------+---------------------
   9    842b.2ba6.3a7d   dynamic  GigabitEthernet1/3
```

Managing the CAM Table (continued)

To specifically display only *dynamic* or *static* CAM entries:

Switch# *show mac address-table dynamic*
Switch# *show mac address-table static*

To view the total number of entries in the CAM table:

Switch# *show mac address-table count*

```
MAC Entries for all vlans:
Dynamic Unicast Address Count:                 234
Static Unicast Address (User-defined) Count:   0
Static Unicast Address (System-defined) Count: 6
Total Unicast MAC Addresses In Use:            240
Total Unicast MAC Addresses Available:         55000
Multicast MAC Address Count:                   9
Total Multicast MAC Addresses Available:       32768
```

The CAM aging timer can be changed from its **default of 300**, though this is needed only in rare circumstances:

Switch# *config t*
Switch(config)# *mac address-table aging-time 360*

To add a static entry into the CAM table:

Switch(config)# *mac address-table static 0011.2233.4455 vlan 9 interface GigabitEthernet 2/7*

To clear all dynamic entries in the CAM table:

Switch# *clear mac address-table dynamic all*

To clear a single entry in the CAM, either by MAC address or interface:

Switch# *clear mac address-table dynamic address d067.e51e.e35a*
Switch# *clear mac address-table dynamic interface GigabitEthernet 2/1*

Note: In Cisco IOS versions prior to 12.1, the syntax for all CAM table commands contained an additional hyphen between *mac* and *address:*

Switch# *show mac-address-table*

This additional hyphen is **no longer required** on modern versions of the IOS. Some IOS versions may support both syntaxes.

Ternary Content Addressable Memory (TCAM)

Recall that switches utilize **TCAM tables** for two purposes:

- Filtering traffic using access-lists
- Prioritizing traffic using QoS

Some Layer-3 devices store the routing table in TCAM as well. Most Layer-3 switches support multiple TCAM tables, to separately manage the access-lists for inbound and outbound traffic, and for QoS.

The TCAM consists of two components:

- **Feature Manager (FM) –** automatically integrates access-lists into the TCAM.
- **Switching Database Manager (SDM) –** supports partitioning the TCAM for separate functions (supported on only some Cisco models).

Each entry in the TCAM table contains three components, defined by access-list entries:

- **Values –** defines the addresses or ports that must be matched
- **Masks –** defines how much of each address to match
- **Result –** defines the action to take when a match occurs

Consider the following access-list:

Switch(config)# *access-list WEB permit tcp 10.1.1.0 0.0.0.255 host 10.2.1.1 eq 443*
Switch(config)# *access-list WEB deny tcp 10.1.0.0 0.0.0.255 host 10.2.1.1 eq 80*

- The **values** are the source (*10.1.1.0*) and destination (*10.2.1.1)* addresses, and the TCP ports (*443* and *80*, respectively).
- The **masks** are *0.0.0.255* for the source, and *0.0.0.0* for the destination. This indicates that the first three octets must match for the source, and the destination much match exactly.
- The **results** are *permit* for the first entry, and *deny* for the second. Other results are possible - such as when using QoS which is more concerned with *prioritizing* traffic than *filtering* it.

The Feature Manager (FM) will automatically integrate the access-list named *WEB* into the TCAM. Configuring the TCAM consists *solely* of creating the necessary access-lists. However, the access-list will not take effect until it's applied to an interface or VLAN.