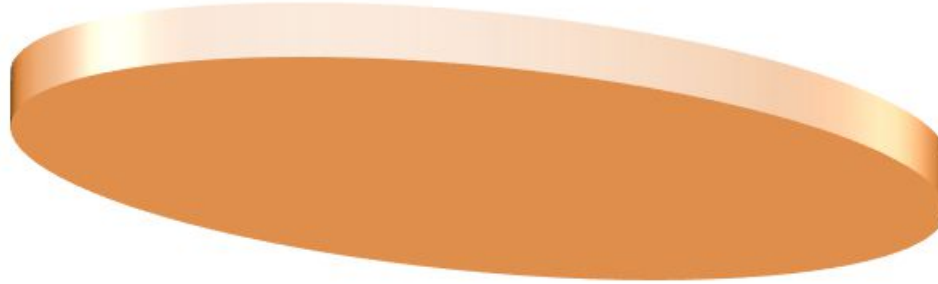


CCNP SWITCH Summary



No filler.

No hype.

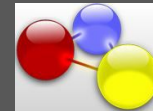
Exam and job-focused.

Cutting-Edge Enhancements.

“A portable, comprehensive guide with everything you need to get up to speed and pass the SWITCH Exam - the first time.”

Aaron's site: **CCNPGuide.**

Jean-Christophe's links:



Introduction, by Aaron

I started *www.ccnpguide.com* as a way for me to capture technical notes as I prepared for the three major CCNP Exams – SWITCH, ROUTE, & TSHOOT. As I began sharing my notes with the world, I immediately started to receive feedback on the SWITCH exam's focus areas and how difficult it was. What I realized was that the exam prep resources available (read: Cisco Press Books) were not even covering all of the exam topics, including some that you were required to configure in live simulation scenarios. First-time fail rates seemed normal and a big part of that was because the some of the simulation scenarios required you to know some extremely specific protocol configuration details that most network professionals just wouldn't know off the top of their heads.

I began to tailor my notes to include topics that were not being covered in "official" exam guides and trimmed down those that just were not necessary. The feedback was overwhelmingly positive from the online community! The problem is, of course, that the notes were not formatted well for off-line consumption.

This guide is an answer to the countless requests to create a portable, comprehensive, and exam-focused SWITCH prep guide. I've refined the online notes even more to focus exclusively on exactly what Cisco expects you to know on exam day.

If you have questions or want to reach out to me directly - shoot me an email at **aaron@ccnpguide.com**. I promise you'll get a response.

For simulation scenarios, hundreds of interactive flashcards and other CCNP guides (ROUTE & TBSHOOT), follow the CCNPGuide link on first page.

Best of luck.

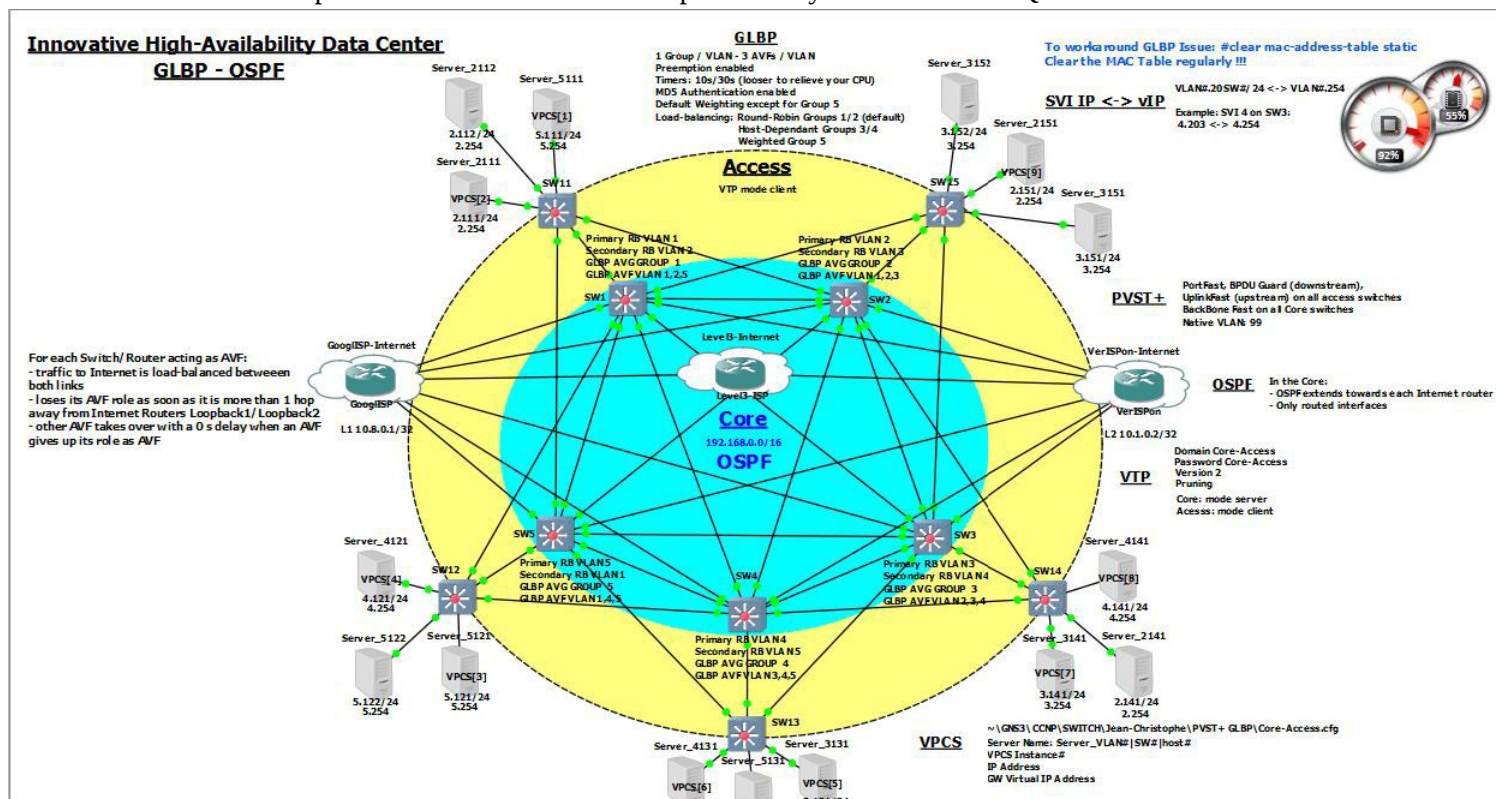
Introduction, by Jean-Christophe

I've found Aaron's first guide interesting and simple. I saw I could improve it through many enhancements and summarizations. Then, I felt that all this personal work could be helpful to so many other people preparing for this exam.

The layout of this document is somehow different from what many of you might have experienced in my previous work, namely '[CCNP ROUTE Summary](#)'. I recommend you to check the detailed information there about **ACLs, SLA and tracking objects**.

I made a collection of [GNS3 CCNP labs](#) (you'll need to bring in your own IOS):

- labs from the official CCNP SWITCH Lab Manual, which you'll be able to practice with GNS3 and a free online switch labs.
- self made labs which helped me practice with real IOS; they cover approximately 50% of the SWITCH topics due to GNS3 limitations, until the new GNS3 1.0 is released where ~ 90% of the topics should be covered. The expected early release date is 1Q 2014 for contributors.



You can find some official exam practice tools [here](#).

As always, everyone has the possibility to write comments online [here](#) to help improve this project. Good luck.



References

- CCDA 640-864 Official Cert Guide, by Anthony Bruno and Steve Jordan
- CCNP SWITCH 642-813 Quick Reference, by Denise Donohue
- CCNP SWITCH 642-813 Official Certification Guide, by David Hucaby
- CCNP ROUTE Summary, by Jean-christophe Manciot
- CCNP, by INE
- Understanding VLAN Trunk Protocol (VTP), by Cisco
- Configuring 802.1Q and Layer 2 Protocol Tunneling, by Cisco
- Configuring Private VLANs, by Cisco
- Configuring Isolated Private VLANs on Catalyst Switches, By Cisco
- PVLAN, by Cisco
- STP Enhancements, by Cisco
- Troubleshooting Spanning Tree PVID and Type-Inconsistencies, by Cisco
- RSTP, by Cisco
- Understanding STP and RSTP Convergence, by INE
- Configuring Port Security, by Cisco
- Layer 2 Attacks and Their Mitigation, by Louis Senecal, Cisco
- IP Address Spoofing, by Wikipedia
- CAM, by Wikipedia
- Cisco Express Forwarding, by Wikipedia
- Configuring a CEF Load-Balancing Scheme, by Cisco
- Troubleshoot Unicast IP Routing Involving CEF on Catalyst 6500/6000 Series Switches, by Cisco
- Campus Network for High Availability Design Guide, by Cisco
- Cisco Wireless Mesh Networking, by Cisco
- Cisco Wireless Mesh Access Points, Design and Deployment Guide, Release 7.0, by Cisco
- Wireless LAN Fundamentals, by CA E-Service Training
- Configuring Voice VLAN, by Cisco
- VoIP bandwidth: Calculate consumption, by TechTarget
- DSCP numbers, by Brian Osgood (arteq)
- Quality of Service on the Cisco Catalyst 4500E Supervisor Engines, by Cisco
- QoS Best Practices, by Packet Pushers

Chapter 1. PPDIO. Design & Planning

Chapter 2. VLANs & Trunks

Chapter 3. EtherChannels

Chapter 4. Spanning Tree Protocol

Chapter 5. Security

Chapter 6. Inter-VLAN Routing

Chapter 7. IP SLA

Chapter 8. High Availability

Chapter 9. Wireless

Chapter 10. VoIP & QOS

Chapter 11. Syslog, CDP, SNMP, RMON & NetFlow

Shortcuts.

Cisco
642
813

Chapter 1

PPDIOO, Planning & Design

Planning

It's tough for Cisco to test how to write up an implementation plan within the time frame allowed for the exam, so they test it indirectly. They may present a complicated business problem with many undefined technical "implementation" components and require you to solve the problem. In order to do so, you'll have to be able to come up with an implementation plan on the fly to know which technologies, protocols, interfaces, etc. need to be configured. Once you configure them, you will also need to come up with a "verification plan" in your head so you can verify that the business need was met (and you get your points for the question).

An example may be a complex problem requiring you to configure new VLANs on a recently added switch (VLAN plan), add LACP trunks (VLAN plan), change the routing on the existing multilayer switches to add the new VLAN networks (layer 3 planning) and load balance all the new connections using GLBP (High Availability plan) based on business VLAN requirements (VLAN plan).

It's easy to see how quickly a simulation scenario like that can cover many of the blueprint planning topics in a single exam question. Expect to see situational problems like that example.

Implementation Plan Components

The implementation should consist of several phases (ex. install hardware, push configurations, cut-over to production, etc.). It is important to remember the following steps for each phase:

- Description of the step
- Reference to design documents
- Detailed implementation guidelines
- Detailed roll-back guidelines in case of failure
- Estimated time needed for implementation

Design, Implementation and Verification Plans Recommendations

There are some general guidelines Cisco recommends around Layer 2 design:

- **Access Layer:** local VLAN approach if possible within the campus environment. That allows the access layer to focus on port density and VLAN termination. It can also be used for inter-vlan routing and boundary definitions.
- **Distribution layer:** it can be used for inter-vlan routing.
- **Core layer:** it is used exclusively for optimized transport of traffic.

General Network Planning Guidelines

Design Plan

- When verifying a new network design, test it first on a pilot network before implementing it network-wide on the production network
- When planning for HA, to minimize the risk of potential outages, it is critical to use the appropriate technology as well as redundancy within that technology to prevent single points of failure

Implementation Plan

A documented rollback plan should be part of any implementation plan.

Security Planning Guidelines

Design and Implementation Plans

- Make sure you have a list of the applications running in the environment
- If it is a security design, Cisco recommends having a network audit
- Critical pieces to include when designing and implementing a security solution:
 - An incident response plan
 - The organization's security policy
 - A list of customer requirements

Verification Plan

Verification of an implemented security solution requires results from audit testing of the implemented solution.

VLAN Planning Guidelines

Implementation Plan

- Some examples of organizational objectives when developing a VLAN implementation plan could include: improving customer support, increasing competitiveness, and reducing costs
- It is critical to have a summary plan that lays out the implementation overview.
- Incremental implementation of components is the recommended approach.

Verification Plan

A VLAN-based verification plan should include:

- Verification that the SVIs have already been created and that they show up on all required switches.
- Verification that trunked links are configured to allow the newly created VLANs

PPDIOO

- **Prepare** – organizational requirements, strategy, financial justification
- **Plan** – network requirements, gap analysis with existing network infrastructure, project plan
- **Design** – design specification created (used for implement phase)
- **Implement and Verify** – network is built, additional components added and verified
- **Operate** – maintaining network health, day-to-day operations
- **Optimize** – proactive management, potential to optimize network redesign

High-level benefits of a life cycled approach:

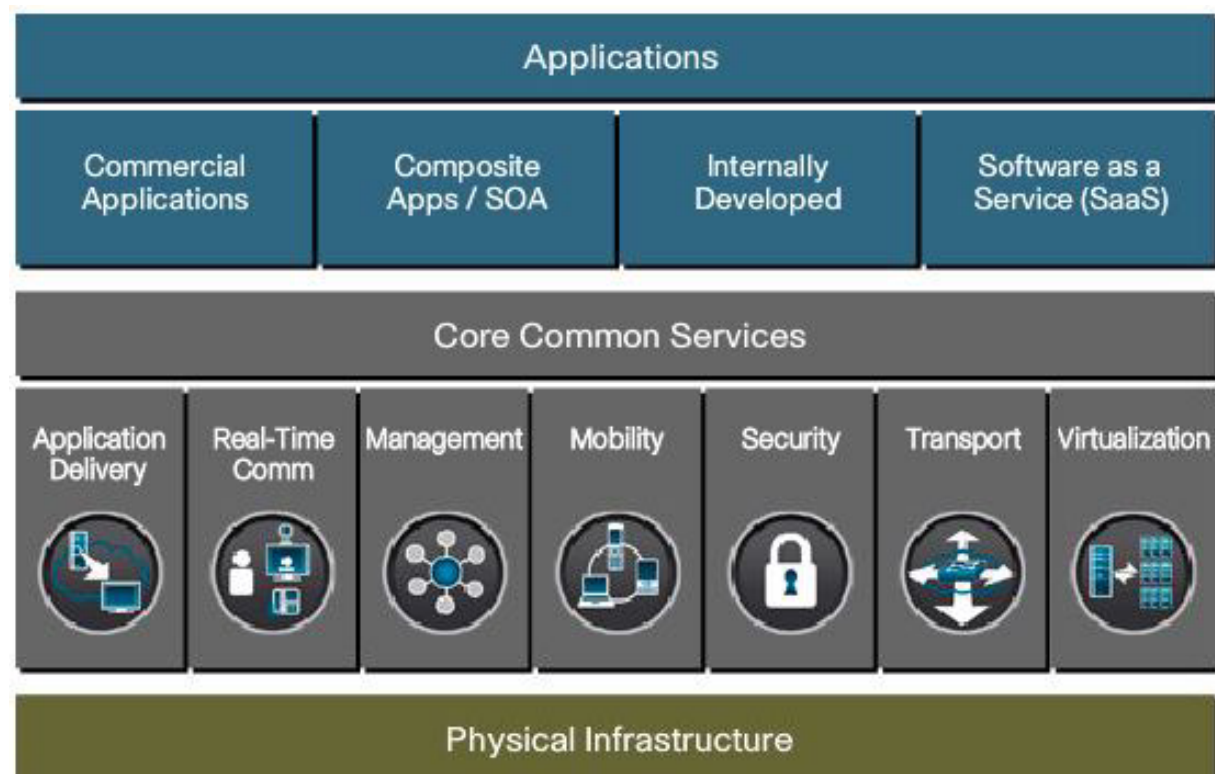
- Lower TCO of network
- Increased availability
- Improved business agility
- Faster access to applications and services

SONA

SONA (Service-Oriented Network Architecture) is a Cisco model that provides guidance, best practices, and blueprints for connecting network services and applications to enable business solutions.

SONA outlines three layers for the enterprise network:

1. **The Network Infrastructure Layer** – where all the network devices are connected (network, servers, storage, etc)
2. **The Core Common Services Layer** – Allocated resources to applications delivered through the network infrastructure layer:
 -> Services such as security, mobility, virtualization...
3. **The Application Layer** – Includes business applications.



Cisco
642
813

Chapter 2

VLANS &

Trunks

VLANs

A VLAN = a single broadcast domain = logical network segment (subnet)

VLANs are used to segment large broadcast domains into smaller, more manageable sections. By default, all switch ports are assigned to VLAN 1, type Ethernet, and MTU of 1500 bytes.

Note: End user devices associated with a VLAN are unaware that the VLAN even exists.

VLAN Configuration

Static VLANs

You should be aware that there are two types of VLAN configuration, static and dynamic. The most common method is static because it is simple and easy to configure. It must be configured on every interface for every device.

To create a vlan:

```
Switch# conf t
Switch(config)# vlan 43
Switch(config-vlan)# name Marketing
Switch(config-vlan)# exit
```

Assign it to an interface:

```
Switch(config)# int fa 1/23
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 43
Switch(config-if)# no shut
```

To delete a vlan:

```
Switch(config)# no vlan 43
```

To manage data transfer in the VLAN:

Switch(config-vlan)# shutdown	=> state: Active/lshut - no data transfer on that VLAN on that switch
Switch(config-vlan)# state suspend	=> state: Suspended - no data transfer on that VLAN on all switches of the same VTP domain
Switch(config-vlan)# state active	=> state: Active - data transfer restored on that VLAN on all switches of the same VTP domain

Dynamic VLANs: VMPS with VQP

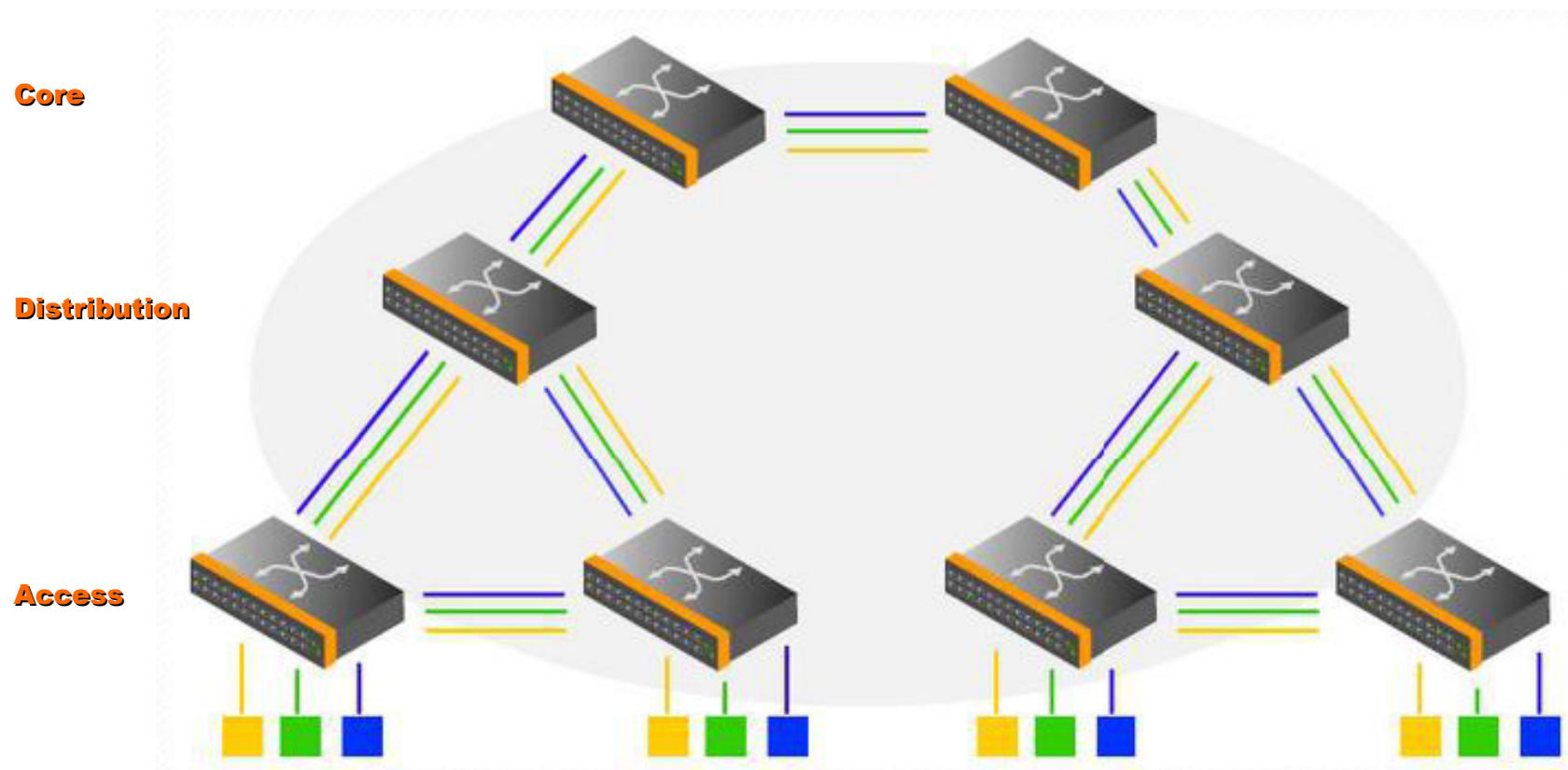
A VLAN Membership Policy Server (VMPS) can be used with VLAN Query Protocol (VQP) to dynamically assign ports to a VLAN based on the source MAC address of the host that is attached to the interface. If the same host moves to another switch port on the network, the new interface is automatically assigned to the proper VLAN.

VLAN Models

End-to-end

In this campus-wide VLAN deployments, every VLAN is made available to every access switch across the network. In this option, broadcasts must cross the core and suck up valuable resources. Usually it uses VTP Client/Server modes.

This model is sometimes implemented for 3 primary reasons. First, users can connect to any switch port independently of their physical location and be placed on the correct VLAN. Second, resource and security parameters can be defined for all members of a particular VLAN and can be updated from a central location. Third, the possibility to move Virtual Machines (VMs) around in a Data Center usually requires the VM to remain in the same subnet (VLAN) during the 'motion'.

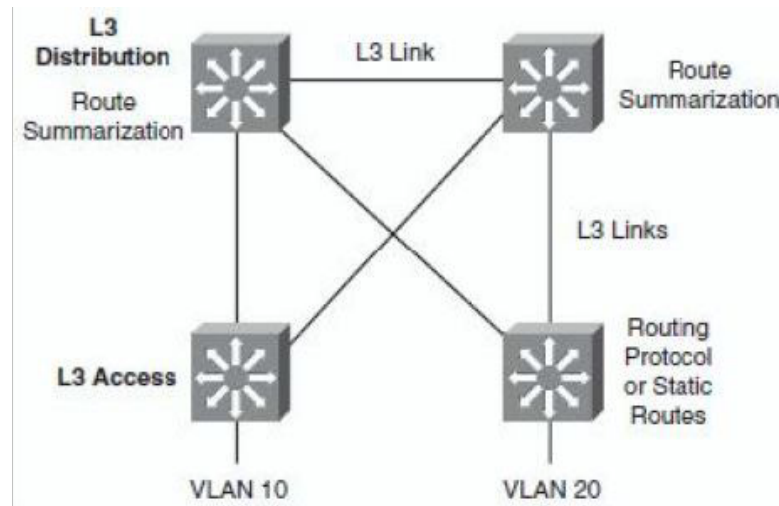


Local

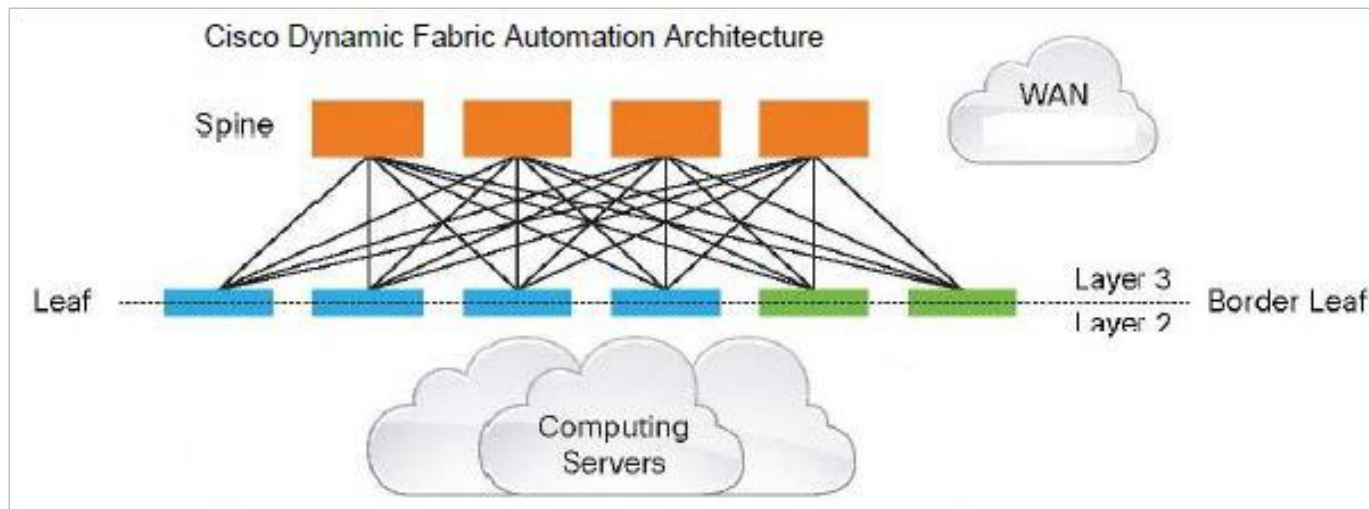
2/26/15

It uses layer three at the access/distribution layer to keep inter-VLAN traffic within that switch block and is better suited for environments where most traffic is not locally destined. Usually uses VTP transparent mode because you don't want the VLANs propagated around the network (hence, "local"). In this model, a VLAN should not extend past the access/distribution switch.

Recommended design: L3 access switches. This gives the fastest convergence and is easier to implement. All links between switches are L3. There is no need for FHRP (HSRP, VRRP, GLBP) nor RSTP, although RSTP with enhancements could still be enabled on edge ports in case of a misconfiguration. Access switches can load balance traffic across both uplinks. The access switches either run a routing protocol or use static routes. The distribution switches summarize routes for the access VLANs.



In recent Cisco **Dynamic Fabric Automation (DFA)** architecture designed for Data Centers, the distribution layer has collapsed:



The 80/20 and 20/80 Rules

Back in the 1990s when most network resources were local (ex. printers, servers), a design rule developed known as the “80/20 rule”. The rule stated that you should design network boundaries such that 80% of traffic stays within the local subnet (doesn’t cross a backbone or leave the VLAN) and only around 20% of traffic should be destined for remote sites (ex. Internet). Well the enterprise and computing has changed dramatically since then with web-based services exploding – and now the new recommendation is the opposite, or the 20/80 rule. That means that 20% of traffic is local and 80% traverses the distribution/core layers to access remote sites.

It’s an important concept because local VLAN model generally follows the 20/80 rule, where most traffic is destined remotely.

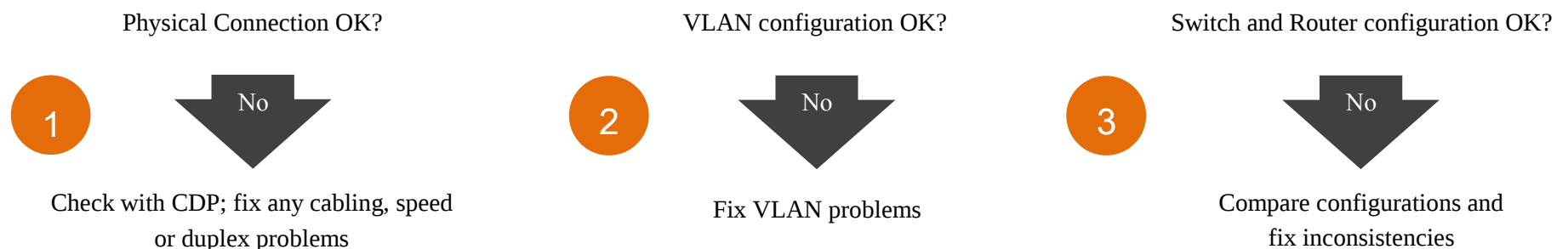
Planning to Implement a VLAN-Based Network Solution

- Number of needed IP subnets (data, voice,...)
- VLAN numbers, names and uses
- VLAN <-> IP mapping
- Scope of each VLAN: end-to-end or local VLAN
- Location of each VLAN
- VLAN assignments: static VLAN on port basis or dynamic VLAN using VMPS with host MAC address
- Trunking
- VTP
- Verification plan

Best practices for VLAN design

- For the local VLANs model:
 - limit 1-3 VLANs per access switch and limit those VLANs to only a couple access switches and the distribution switches,
 - VTP is not necessary (use transparent mode).
- Try to separate data, voice, video, management, default, and blackhole VLANs (with their own VLAN ID).
- Avoid using VLAN 1 as:
 - the “blackhole” for all unused ports,
 - the data VLAN.
- Turn off DTP (Dynamic Trunk Protocol) on trunk ports and configure them manually – also use IEEE 802.1Q rather than ISL.
- Manually configure access ports that are not intended to be trunks by using the **switchport host** command: it enables PortFast, disables trunking and EtherChannel.
- Avoid Telnet on management VLANs, use SSH instead.
- User access ports are typically at least Fast Ethernet or Gigabit Ethernet. Links between the access and distribution layers are typically Gigabit Ethernet or faster, layer 2, and have an **oversubscription** ratio (ratio of downlinks to uplinks) of no more than 20:1. Links between the distribution and core should be Gigabit Etherchannel, 10G Ethernet or 40G Ethernet, layer 3, with an oversubscription ratio of no more than 4:1. In Data Centers, maximum oversubscription ratios revolve around 3:1.

VLAN Troubleshooting Steps



VLAN Verification

To determine the trunking status of an interface:

```
# show int fa 1/24 trunk
```

*A simpler alternative would be
show interfaces trunk*

To determine the physical status of a link:

```
# show int fa 1/24 status
```

To check if an interface is assigned to a specific VLAN:

```
# show vlan id 100
```

This command is especially helpful as it displays all ports belonging to the VLAN as well as the MTU of each assigned port and type.

For more detailed switchport information:

```
# show int fa 1/24 switchport
```

*Note: If you are troubleshooting a trunk link with this command and notice that the operational mode description says “down”, the interface itself is shutdown and needs to be started with the **no shut** command.*

To see a list of VLANs and their assigned interfaces:

```
# show vlan brief
```

To see a complete detailed interface list for all VLANs:

```
# show vlan
```

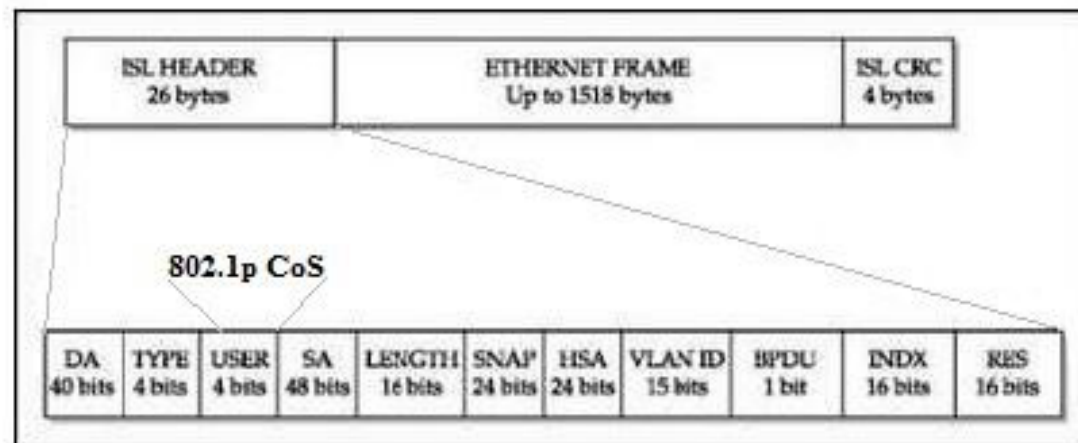
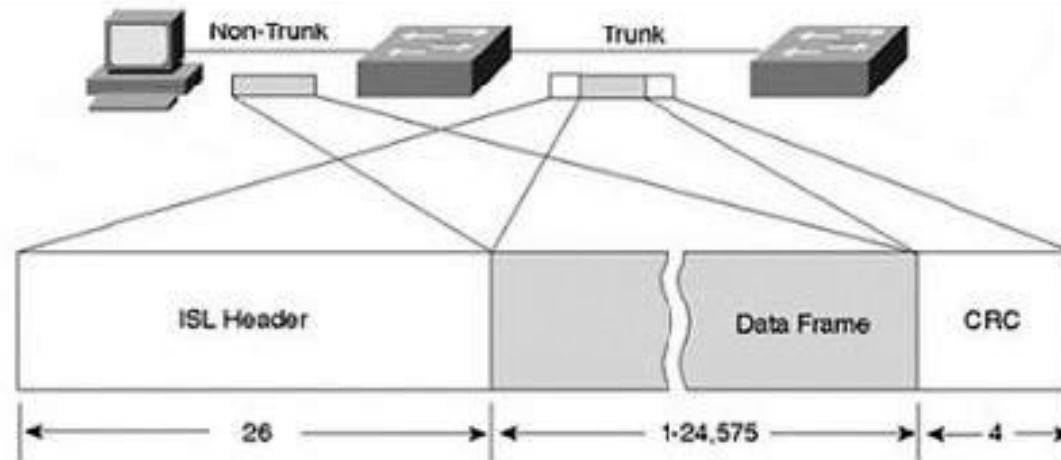
Note: The show vlan command does not include trunk ports in its VLAN port output as they carry a variety of VLANs.

VLAN Trunking

There are two frame tagging methods for trunk links:

ISL

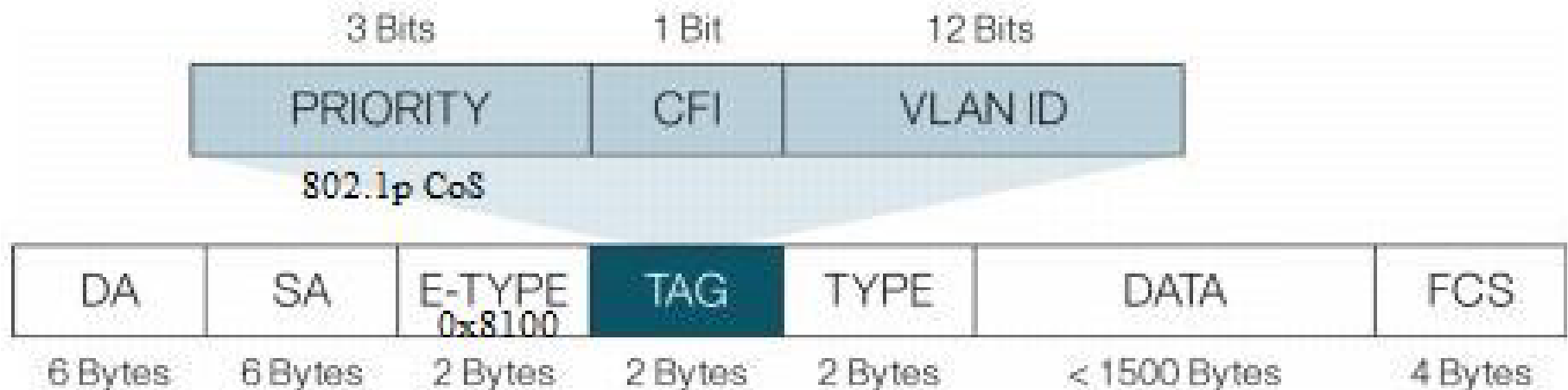
The ISL header is 26 bytes and it appends an additional CRC which is 4 bytes, for a total of 30 additional bytes to every ISL encapsulated frame. The header has a 1-byte User field that carries an IEEE **802.1p class of service (CoS)** value in the three least-significant bits. Because it is proprietary, ISL trunk encapsulation will only work with Cisco devices – and *not all Cisco switches even support it*.



802.1Q

802.1Q is an open standard, inserts its own 4 byte tag into the Ethernet frame for VLAN tagging and recalculates the CRC value, allows for native VLANs (untagged frames to go through).

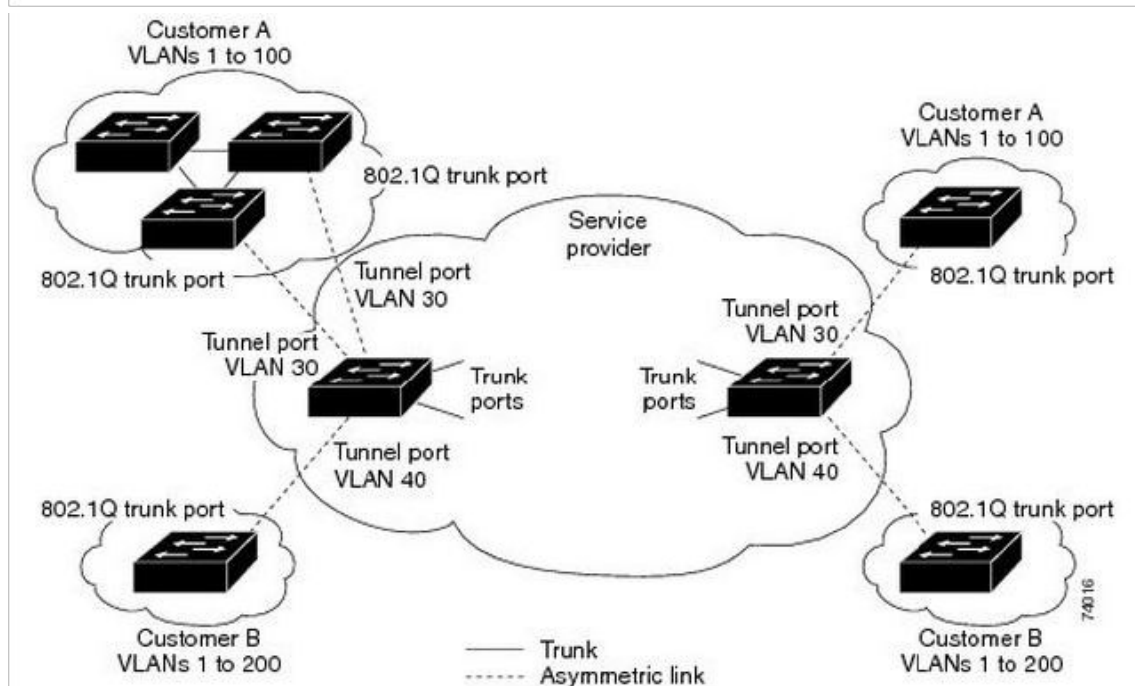
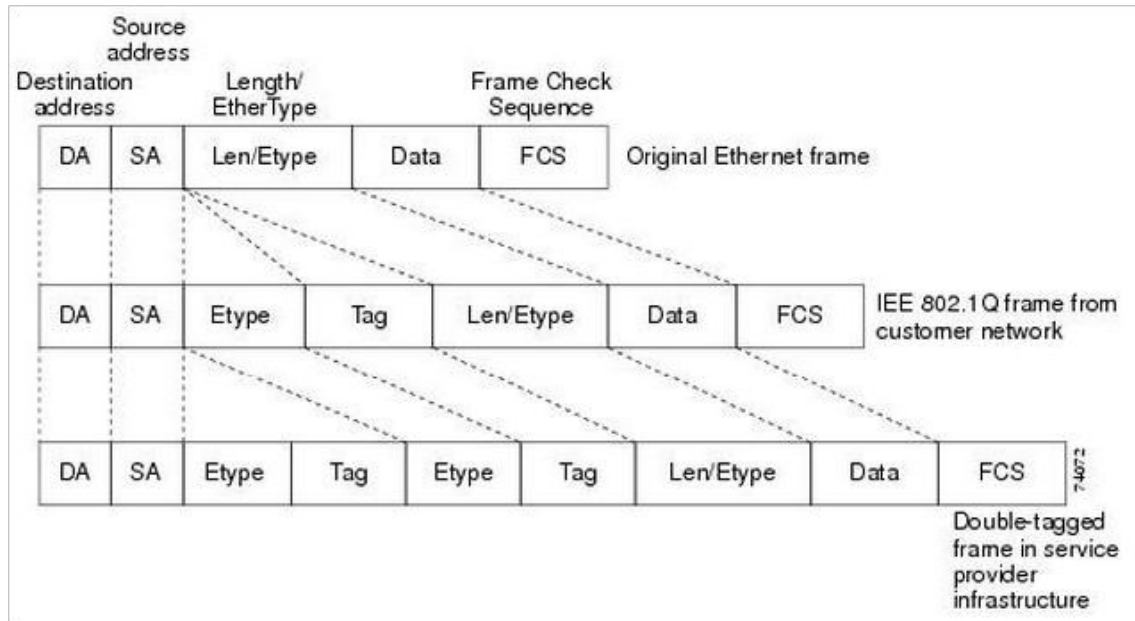
802.1Q has become the dominate layer 2 trunking protocol in use today as fewer organizations use Cisco's proprietary ISL. 802.1Q is designed exclusively for point-to-point links. The 4 byte field that is inserted by 802.1Q does not interfere with the original frame header, so the MAC source/destination information is unchanged.



Using 802.1Q or ISL can create problems with other tagging methods. The maximum size for any Ethernet frame, as specified by IEEE 802.3, is 1518 bytes. That means that if a frame entering a trunk port is already near the maximum size, the header and CRC added by ISL or the inserted tag added by 802.1Q will push the frame size over the IEEE limit. To resolve this conflict, the IEEE 802.3 committee created a subgroup – 802.3ac that extended the maximum Ethernet frame size to 1522 bytes.

If you see the “Giants” counter on an interface different than 0, you probably have an *encapsulation mismatch* between both ends of the trunk..

802.1Q is often used by service providers for tunneling secure VPNs. 802.1Q tunneling feature allows ISPs to segregate different customer's traffic throughout their infrastructure; this is often called **Q-in-Q**.



VLAN Trunking Configuration

To Configure a VLAN Trunk Interface

```
Switch(config)# int fa 1/5
Switch(config-if)# switchport
Switch(config-if)# switchport trunk encapsulation {isl | dot1q | negotiate}
// Default is negotiate: Specify that if Dynamic Inter-Switch Link (DISL) and Dynamic Trunking Protocol (DTP)
// negotiation do not resolve the encapsulation format, ISL is the selected format.
Switch(config-if)# switchport mode {trunk | dynamic {desirable | auto}}
// Default is Dynamic auto.
// If set to dynamic, it defaults to ISL if not specified.
Switch(config-if)# switchport nonegotiate
// Disables DTP.
Switch(config-if)# switchport trunk native vlan n
// For 802.1Q trunks only
Switch(config-if)# switchport trunk allowed vlan {list | add list | remove list | except list | all}
// Allowed VLANs for transmission on the trunk; no control for VLANs received on the trunk.
// Removing some VLANs also prevents STP BPDUs from being properly exchanged on that trunk for these VLANs
```

Trunk links by default allow all active VLANs (those that the switch knows about). Also, all dot1Q trunks use VLAN 1 as the default native VLAN.

Because the switch will forward broadcasts out all ports on that VLAN, frames will be forwarded over the trunk too – which could waste trunk bandwidth. It is recommended to specifically allow only VLANs that cross the trunk using the “**switchport trunk allowed vlan**” command, or VTP pruning on all switches.

If a non-trunking port receives an ISL encapsulated frame, it will not be able to remove the ISL header and will by default drop the ISL frames.

If a non-trunking port receives an 802.1Q encapsulated frame, it simply reads the destination MAC address and forwards the frame as it would for any other layer 2 frame.

Trunking Modes

Notice that even dynamic desirable (the most active dynamic trunking mode) will still not form a trunk if the other end is configured as an access port.

Trunk

Manual permanent trunking mode. It uses DTP though.

Dynamic desirable

The port actively tries to bring up the link as a trunk, sending negotiations to the other end with DTP.

Dynamic auto (Default)

The port can be converted to a trunk link, but only if the far end requests it (Dynamic desirable or Trunk). It uses DTP.

Nonegotiate

Does not send DTP frames. Both sides will need *trunk* mode in order to trunk.

	Dynamic Auto	Dynamic Desirable	Trunk	Access
Dynamic Auto	Access	Trunk	Trunk	Access
Dynamic Desirable	Trunk	Trunk	Trunk	Access
Trunk	Trunk	Trunk	Trunk	Limited Connectivity
Access	Access	Access	Limited Connectivity	Access

Limited Connectivity:

- Trunk port is **Trunk**
- Access port is **Blocking (Inconsistent type)**

DTP

DTP (Dynamic Trunking Protocol) is a proprietary protocol for negotiating a common trunking mode between two switches.

DTP negotiates in order:

- ISL trunk
- 802.1Q trunk
- Access port

Trunk Troubleshooting

When troubleshooting a trunk link, all of the following must be set the same on both ends:

- trunking mode (trunk or dynamic auto or dynamic desirable)
- encapsulation
- native VLANs (For 802.1Q only and will only break native VLAN traffic if mismatched)
- allowed VLANs

=> Native VLANs

It is important that the native VLAN is configured correctly on both sides of an 802.1Q trunk. Native VLAN is a “default” VLAN that allows frames to be passed through the trunk untagged. If the switches were connected through a hub and there were other devices connected to it, they could use the native VLAN. This is a rare situation, but worth understanding.

If you are required to troubleshoot VLAN traffic that is not being passed across a trunk, make sure that the VLAN is in the interface allowed list for each side of the trunk. While all VLANs are allowed by default across trunk links, many organizations explicitly define allowed VLANs over trunks for security and to prevent unnecessary broadcast traffic on the link.

VTP

- VTP (VLAN Trunking Protocol) uses layer 2 trunk frames to communicate VLAN information among switches. It manages the addition, deletion, and renaming of VLANs across the network from a single source.
- Organized into domains (only one per switch). Each switch within that domain must have the same VTP domain name and password configured otherwise database information will not be synchronized. Because each switch can only be configured with a single VTP domain, it will only listen and act on VTP advertisements it hears that match its own VTP domain name. Advertisements are used to communicate changes to other switches.
- Dynamic Trunking Protocol (DTP) sends the VTP domain name in a DTP packet. Therefore, if you have two ends of a link that belong to different VTP domains, the trunk does not come up if you use DTP. In this special case, you must configure the trunk mode as trunk and nonegotiate, on both sides, in order to allow the trunk to come up without DTP negotiation agreement.
- New switch with null VTP domain will accept configured domain from other switch with DTP, and trunk on negotiated link will be formed.

VTP Modes

VTP has three modes:

Server mode (default) – These switches have full control for creation and changes to VLANs. All changes are advertised out to all other switches, except in v3. Each domain has at least one VTP server.

Client mode – Cannot create or change VLANs, but they do send periodic advertisements and can change their configurations to match those they hear. Domain name and password are configurable.

Transparent mode – Does not participate in VTP. In VTP version 1, a switch in transparent mode inspects VTP messages for the domain name, password and version and forwards a message only if all match. VTP version 2 forwards VTP messages in transparent mode without checking the version - only matching VTP domain name and password are required.

VTP v3 has two special server modes:

Primary Server mode – These switches are the only ones allowed to update other switches, regarding their VLAN (or MST, see below) configuration. The designation of the primary server overwrites the configuration of all devices in the domain, including any conflicting devices.

Secondary Server mode (default) – These switches are **NOT** allowed to update other switches. This feature reduces the risk of unintended automatic configuration changes, for example, by adding a configured switch with a higher configuration revision number to an existing network.

VTP Configuration Revision Number

VTP switches use an index called the VTP configuration revision number which is sent with VTP advertisements. The configuration revision number helps to identify changes to the network by increasing the revision number by one every time a change occurs on a server. Every switch stores the revision number of the last advertisement it heard.

If a switch receives an advertisement with a higher revision number than the one stored locally, it sends an advertisement request and after having received the subset advertisement, its configuration is changed to reflect the new advertisement and forwards it to its neighboring switches. If the revision number is the same as in the switch local database, it simply ignores the advertisement. Finally, if the number in the advertisement is lower than the number stored in its database, the switch will respond back with its more current VLAN information.

It is important to set the revision number to 0 before inserting a new switch into a production environment with VTP v1/2. Transparent mode revision number is always 0.

There are two ways to do it:

- Change it to transparent mode, then back to server.
- Change the VTP domain name, then change it back to the original.

If a switch is set to server (the default) or client and is inserted into the environment with a higher rev. number than the last advertisement, a VTP synchronization problem occurs, potentially disabling all VLAN-assigned ports. Note that even a client with a higher revision number can take down the entire network if it propagates its VLAN database to its peers - so be very careful when adding new switches!

Another solution is to use VTP version 3 where a switch must be manually configured as a “**vtp primary server**” to be allowed to send VTP advertisements.

Also, VTP information is stored in flash in the vlan.dat file. That way it survives reboots.

To check the VTP Revision Number:

```
Switch# show vtp status
```

To backup VTP information on another location:

```
Switch # copy { flash | cat4000_flash | const_nvram }:vlan.dat { tftp | ftp }://url/vlan.dat
// For Catalyst 29xx,35xx,3750 | 4000,4500 | 6000,6500
```

VTP Message Types

They are always exchanged in untagged VLAN 1 between switches, even if the 802.1Q trunk native VLAN is different.

There are three different types of VTP messages:

Summary Advertisements

Sent from all switches every 300 seconds (5 minutes) and after any VLAN-related changes (Added, removed, renamed). They contain the version, domain name, MD5-hashed password and configuration revision number.

Subset Advertisements

VTP servers send subset advertisements after a VLAN change occurs that follow the summary advertisements. They provide specific VLAN details.

Advertisement Requests

Clients can request any VTP information they don't have. The server will respond with a summary advertisement and subsequent subset advertisements.

VTP Version => Propagated to all switches in the VTP domain

VTP has three versions (1, 2 & 3) that are not interoperable. All that is required to change from v1 to v2/v3 across the network is to change one (primary) server switch to v2/v3 and it will send out an advertisement to all other switches to make the change as well. v1 is the **default**.

To Configure a VTP server for v2/v3:

```
Switch(config)# vtp version {2 | 3}
```

VTP v2 has the following enhancements over v1:

- Token Ring VLAN support
- Unrecognized TLV (Type-Length-Value) support
- Version-independent message forwarding
- Performs consistency checks

When VTP v3 switches are connected to other switches with lower VTP versions:

- VTP v1 transitions to v2,
- VTP v2 clients and servers will be supplied with VTP information,
- VTP v2 switch will never update a VTP v3 switch.

VTP v3 enhancements over v2:

- Control over which device (primary server) is allowed to update other devices (secondary servers/clients) view of the VLAN topology.
- Supports the whole IEEE 802.1Q VLAN range up to 4095.
- Supports Private VLAN (PVLAN) structures.
- Support for databases other than VLAN (for example, MST).
- Supports the option to turn VTP on or off on a per-trunk (per-port) basis.
- Supports enhanced password (hidden or secret)

VTP Pruning => Propagated to all switches in the VTP domain

VTP Pruning makes more efficient use of trunk bandwidth by reducing unnecessary flooded traffic over trunk links. Broadcasts and unicast frames are only transmitted over a trunk link if the switch on the receiving end of the trunk has ports in that VLAN.

By default, VTP pruning is disabled; to enable it:

```
Switch(config)# vtp pruning
```

When pruning is enabled on a server, it propagates the pruning to all switches in the management domain. (This is generally the quickest way to enable it within your switched network). Also, VLAN 1 is considered pruning ineligible by Cisco. VLANs 2-1000 are eligible for pruning by default.

VTP Configuration

2/26/15

To Configure a VTP v1/v2 Management Domain:

```
Switch(config)# vtp domain domain-name
Switch(config)# vtp password password
Switch(config)# vtp mode {server | client | transparent}
```

VTP v3

```
Switch(config)# vtp domain domain-name
Switch(config)# vtp password password [hidden | secret]
// hidden: the password is stored in a unreadable format in vlan.dat.
// secret: the password must be entered in 32-digit hexadecimal format.
Switch(config)# vtp mode {server | client | transparent} [vlan | mst]
Switch# vtp primary-server [vlan | mst] [force]
Switch(config-if)# [no] vtp
```

- Notes:
- If a VTP password is locally configured, the same password must be set on all VTP-participating switches in the domain.
 - In Client mode, only domain name and password can be configured.
 - VTP information will not be exchanged without first configuring the VTP domain name.

After VTP is configured, the switch will begin passing the configuration revision number, domain name, encrypted password, version number, pruning mode, known VLANs and their parameters through its trunk links.

VTP Example Configuration:

Step 1. Enter global configuration mode:

```
Switch# configure terminal
```

Step 2. Configure the VTP mode as server:

```
Switch(config)# vtp mode server
```

Step 3. Configure the domain name:

```
Switch(config)# vtp domain domain_name
```

Step 4. (Optional.) Enable VTP version 2:

```
Switch(config)# vtp version 2
```

Step 5. (Recommended) Specify a VTP password:

```
Switch(config)# vtp password password_string
```

Step 6. (Optional.) Enable VTP pruning in the management domain:

```
Switch(config)# vtp pruning
```

Step 7. (Optional) Enable VTP version 3:

```
Switch# vtp version 3
```

Step 8. (Optional) Configure the updating server as primary:

```
Switch# vtp primary-server vlan
```

Step 9. (Optional) Turns off VTP at some interface level:

```
Switch(config-if)# no vtp
```

Step 10. (Recommended) Specify a hidden VTP password:

```
Switch(config)# vtp password password hidden
```

VTP Verifying

To display information about the VTP configuration:

```
Switch# show vtp status
```

The “**show vtp status**” command is extremely valuable when troubleshooting a VTP issue. It shows the configuration revision number on the switch, the VTP domain name, VTP version number, and VTP mode (ex. server).

To display information about the password:

```
Switch# show vtp password
```

To display statistics about the VTP operation:

VTP v3

```
Switch# show vtp counters
```

```
Switch# show vtp counters  
Switch# show vtp devices  
Switch# show vtp interface
```

VTP Troubleshooting

Troubleshooting VTP if a switch does not seem to be receiving updates from a VTP server switch:

- Make sure there is physical connectivity between switches
- The link towards the VTP server may not be in trunking mode. Remember that VTP advertisements are only sent over trunked links. Perform a “**sh int xx {trunk | switchport}**” to verify.
- Make sure the VTP domain name and password **match** those of the server (they are case sensitive).
- Make sure the switch is **not** set to transparent mode.
- Make sure the v3 Server switch is set to **primary**.

Private VLANs → Controlling Traffic Within One Subnet

In a regular VLAN, all ports can communicate. **PVLANS provide Layer 2 isolation between the ports within the same PVLAN without assigning ports to separate VLANs or assigning individual hosts to different Layer 3 IP networks.** PVLANS provide a logical separation of the network that keeps traffic isolated. **Both ISPs and web hosting providers are frequent users of private VLANs.**

Primary VLAN

A normal or primary VLAN can be subdivided into several secondary VLANs.

PVLAN ports are associated with a primary and at least one secondary VLAN.

The primary VLAN carries unidirectional traffic downstream from the promiscuous ports to the isolated and community host ports and to other promiscuous ports.

Secondary VLAN Types

They are a 'subdivision' of a primary VLAN and all secondary VLANs are *in same subnet as their primary VLAN*.

There are two secondary VLAN types:

Isolated VLAN

An isolated VLAN is a secondary VLAN that carries unidirectional traffic upstream from the host ports towards the promiscuous ports.

All ports have complete Layer 2 separation (including broadcast separation) from other ports within the same PVLAN (and with other PVLANS), except for promiscuous ports. All traffic to these ports is also blocked, except traffic with same PVLAN promiscuous ports.

Even with static ARP entries, ports within an isolated VLAN cannot communicate with each other

Example: a port configured for a highly-secure server.

Community VLAN

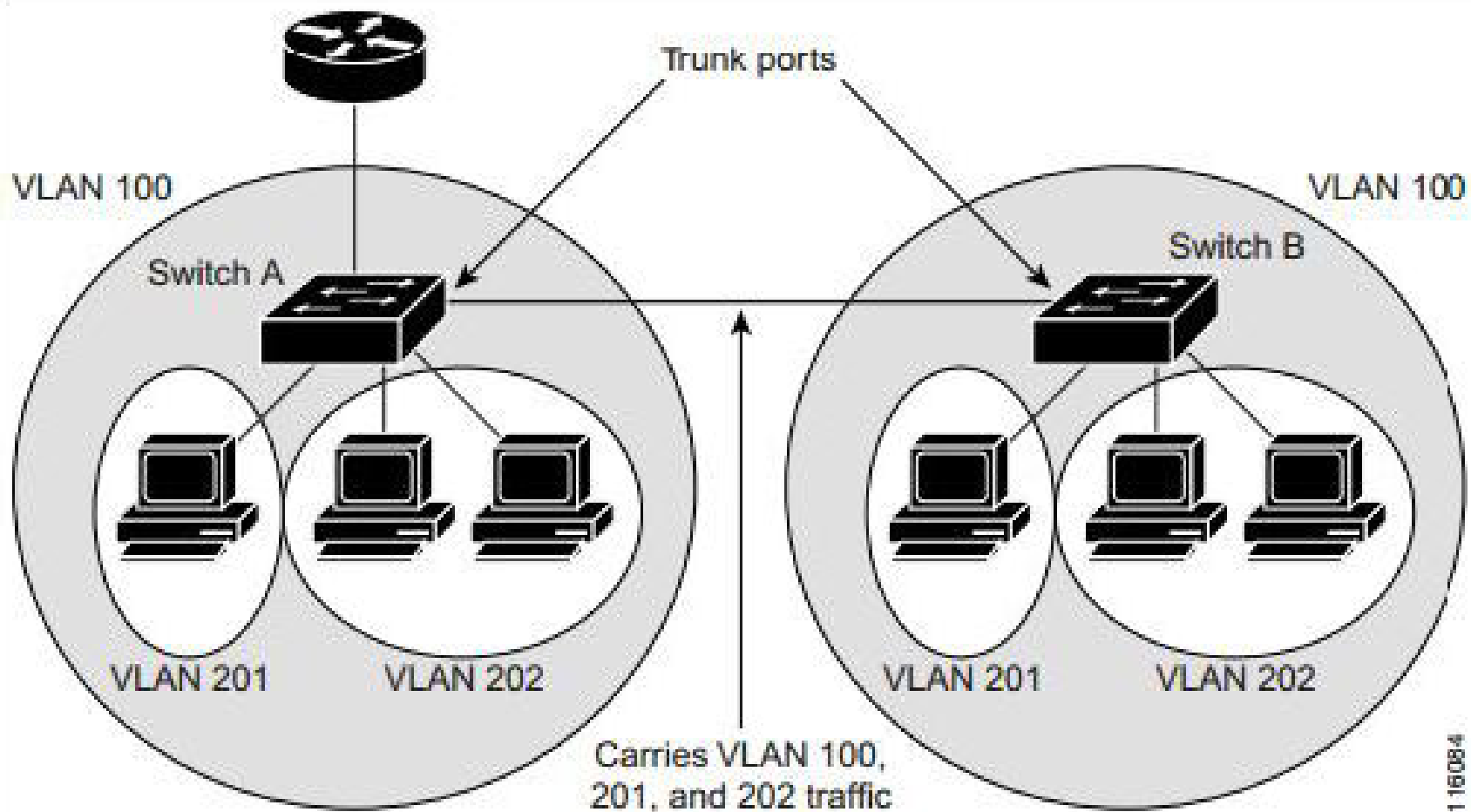
A community VLAN is a secondary VLAN that carries upstream traffic from the host ports to the promiscuous ports and to other host ports in the same community VLAN.

All ports communicate among themselves in the community VLAN as well as with promiscuous ports within the same PVLAN, but cannot communicate with ports in other community VLANs, other isolated VLANs or other PVLANS.

Example: the marketing staff.

Notes:

- All secondary VLANs must be associated with one primary VLAN.
- VTP v1/v2 does **not** pass private VLAN information: you must manually configure private VLANs on all switches in the Layer 2 network.
- VTP v3 does pass private VLAN information, so the configuration can be made only on the VTP primary server.
- On a trunk, frames originating from a promiscuous port are tagged with the primary VLAN ID.
- On a trunk, frames originating from a host port are tagged with its secondary (isolated or community) VLAN ID



VLAN 100 = Primary VLAN
VLAN 201 = Secondary isolated VLAN
VLAN 202 = Secondary community VLAN

116084

Port Modes

Each physical switch interface that uses a private VLAN must be configured with a PVLAN association. The interface can be one of two modes:

Promiscuous Port

This port communicates with all other PVLAN ports and is the port used to communicate with network devices, including routers, backup servers, and administrative workstations. This port listens to the secondary VLANs and sends traffic using the primary VLAN.

Note: Trunk ports are already promiscuous; no need to configure them.

Host Port: Community - Isolated

A switch port that connects to a regular host that resides in a community or isolated VLAN. There are community and isolated host ports. The port only communicates with the promiscuous port or ports in the same community VLAN.

Note that a host port belongs to different PVLANS at the same time: downstream primary and upstream secondary.

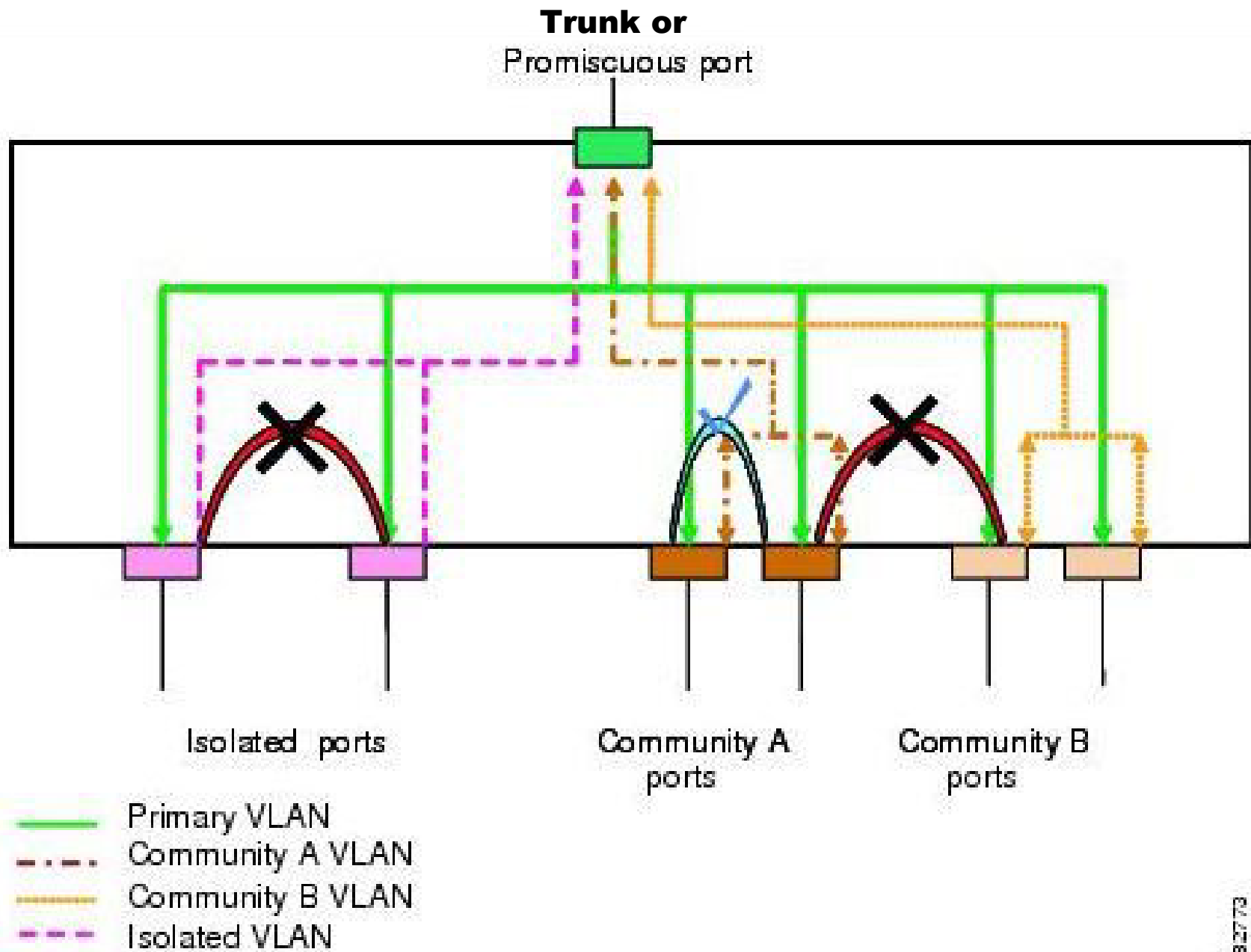
Broadcast and Multicast Traffic

In a regular VLAN, broadcasts are forwarded to all ports in that VLAN. Private VLAN broadcast forwarding depends on the port sending the broadcast:

- A host port in an isolated PVLAN sends a broadcast only to the promiscuous ports or trunk ports.
- A host port in a community PVLAN sends a broadcast to all promiscuous ports, trunk ports, and ports in the same community VLAN.
- A promiscuous port sends a broadcast to all ports in the PVLAN: other promiscuous ports, trunk ports, isolated ports, and community ports.

Limitations

- PVLANS cannot include VLANs 1 or 1002–1005.
- You must set VTP v1/v2 mode to transparent.
- You can only specify one isolated VLAN per primary VLAN.
- You can only designate a VLAN as a PVLAN if that VLAN has no current access port assignments. Remove any ports in that VLAN before you make the VLAN a PVLAN.
- Configuring PVLAN ports as *EtherChannels* is only possible since **IOS Release 15.0(2)SG**.



182773

PVLAN Configuration

1. Set the VTP v1/v2 mode to Transparent

```
Switch(config)# vtp mode transparent
```

2. Define the secondary VLAN(s)

```
Switch(config)# vlan 20
Switch(config-vlan)# private-vlan {isolated | community}
```

3. Define the primary VLAN

```
Switch(config)# vlan 10
Switch(config-vlan)# private-vlan primary
Switch(config-vlan)# private-vlan association {vlan-list | add vlan-list | remove vlan-list}
// vlan-list: n,p,q | n - q
```

4. Define the physical interface

```
Switch(config-if)# switchport mode private-vlan {host | promiscuous}
// For host port:
Switch(config-if)# switchport private-vlan host-association primary-vlan-id secondary-vlan-id
// For promiscuous port:
Switch(config-if)# switchport private-vlan mapping primary-vlan-id secondary-vlan-list [{add secondary-vlan-list} |
{remove secondary-vlan-list}
```

Notes: For interfaces set to host mode, you must "associate" the port with one primary VLAN and one secondary VLAN.

For interfaces set to promiscuous mode you must "map" the port to one primary VLAN and several secondary VLANs.

PVLAN Configuration Example

Let's run through an example that configures both isolated and community secondary private VLANs as well as host and promiscuous interfaces:

```
Switch# conf t
Switch(config)# vtp mode transparent
Switch(config)# vlan 40
Switch(config-vlan)# private-vlan community
Switch(config)# vlan 50
Switch(config-vlan)# private-vlan community
Switch(config)# vlan 60
Switch(config-vlan)# private-vlan isolated
Switch(config)# vlan 100
Switch(config-vlan)# private-vlan primary
Switch(config-vlan)# private-vlan association 40,50,60
Switch(config-vlan)# exit

Switch(config)# int fastethernet 0/4
Switch(config-if)# switchport mode private-vlan host
Switch(config-if)# switchport private-vlan host-association 100 40
Switch(config)# int fastethernet 0/5
Switch(config-if)# switchport mode private-vlan host
Switch(config-if)# switchport private-vlan host-association 100 50
Switch(config)# int fastethernet 0/6
Switch(config-if)# switchport mode private-vlan host
Switch(config-if)# switchport private-vlan host-association 100 60
Switch(config)# int fastethernet 0/1
Switch(config-if)# switchport mode private-vlan promiscuous
Switch(config-if)# switchport private-vlan mapping 100 40,50,60
```

PVLANs on SVIs

On switched virtual interfaces (SVIs) or layer 3 VLANs with IP addresses, an additional map must be inserted. For this example, let's use VLAN 300 as the primary VLAN and VLANs 80 and 90 as secondary VLANs.

```
Switch(config)# vlan 80
Switch(config-vlan)# private-vlan isolated
Switch(config)# vlan 90
Switch(config-vlan)# private-vlan community
Switch(config)# vlan 300
Switch(config-vlan)# private-vlan primary
Switch(config-vlan)# private-vlan association 80,90
Switch(config-vlan)# exit
Switch(config)# interface vlan 300
Switch(config-if)# ip address 192.168.1.199 255.255.255.0
```

At this point, VLAN 300 can communicate at layer 3, but the secondary VLANs (80 & 90) are stuck at layer 2. To allow the secondary VLANs to route layer 3 traffic as well, you need to insert this mapping **on the primary VLAN SVI interface** because you cannot configure Layer 3 VLAN interfaces for secondary VLANs:

```
Switch(config-if)# private-vlan mapping 80,90
// Traffic that is received on the secondary VLANs is routed by the SVI of the primary VLAN
```

Protected Ports

On some lower-end switches, protected ports can provide a simple version of *isolated* private VLANs. Traffic from a protected port can access only an unprotected port. Traffic between protected ports is blocked.

```
(config-if)# switchport protected
// Isolates unicast, multicast, and broadcast traffic at Layer 2 from other protected ports on the same switch
```

Cisco
642
813

Chapter 3

EtherChannels

EtherChannel is a term used to describe bundling or aggregating 2-8 parallel links. EtherChannel provides a level of link redundancy. If one link in the bundle fails, traffic sent through that link is automatically moved to an adjacent link.

Normally multiple links between switches create the potential for bridging loops, but because an EtherChannel bundle is treated as a single logical link by both switches, it avoids the problem, allowing greater bandwidth utilization. If there are two redundant EtherChannel bundles, one entire EtherChannel will be blocked by STP to prevent a loop.

Any changes made to the port-channel interface after the EtherChannel has been created will be automatically made to all other ports in that bundle. Also, bundles cannot form if any of the assigned ports are **SPAN** ports (Switched Port ANalyzer => Port Mirroring).

EtherChannel links can be either *access*, *trunk links* or *routed ports*, but if they are trunked (usually the case), **they require the following to be the same on all connected interfaces:**

- Speed
- Duplex
- Encapsulation type
- Trunking mode
- Allowed VLANs
- Native VLAN
- Channel protocol and compatible modes

EtherChannel Link Negotiation Protocols

2/26/15

PAgP (Port Aggregation Protocol)

- Cisco proprietary
- Will automatically modify interface parameters on all ports in the bundle if the EtherChannel interface is changed.
- STP sends packets over only one physical link in a PAgP bundle. Because STP algorithm uses the lowest port priority (priority + port ID), if defaults are set, STP will always use the **lowest** port number for BPDUs.
- Uses port priority to determine which ports will be active for transmission: the **higher** the port priority, the more likely the port will be used for transmission.

LACP (Link Aggregation Control Protocol)

- An open standard: IEEE 802.3ab
- Switch with the lowest **system priority** (2 byte value followed by MAC) determines which ports are active in the EtherChannel at any given time.
- Uses **port priority** to determine which ports to place in standby mode if hardware limitations do not allow all ports to participate in the EtherChannel: **lowest** priority ports are active (max 8), others are backup ports (max 8).
- Most leave the system and port priority to defaults

EtherChannel Link Negotiation Protocols Summary

PAgP	LACP	Negotiation Characteristics Sent?	
On	On	No	All ports channeling
Auto	Passive	No	Waits to channel until asked
Desirable	Active	Yes	Actively asks to form a channel

PAgP Correct combination	LACP Correct combination
On <-> On	On <-> On
Auto <-> Desirable	Passive <-> Active
Desirable <-> Auto Desirable	Active <-> Passive Active

EtherChannel Configuration

PAgP

PAgP EtherChannel Interface Example:

```
Switch(config)# interface fa 1/1/2
Switch(config-if)# channel-protocol pagp
Switch(config-if)# channel-group number
                        mode {on | {{auto [non-silent]] | {desirable [non-silent]}}}
// Default mode: none
Switch(config-if)# pagp port-priority number
// number <0 <-> 255>, defaults to 128
// The higher the value, the higher the port priority
// the more likely the port will be used for PAgP
// transmission.
```

LACP

LACP EtherChannel Interface Example:

```
Switch(config)# lacp system-priority number
// number <1 <-> 65535>, defaults to 32768
// The lower the value, the higher the system priority
Switch(config)# interface fa 1/1/3
Switch(config-if)# channel-protocol lacp
Switch(config-if)# channel-group number mode {on | passive | active}
// Default mode: depends on the platform
Switch(config-if)# lacp port-priority number
// number <1 <-> 65535>, defaults to 32768
// The lower the value, the higher the port priority
// the more likely the port will be used for
// transmission.
```

By default, PAgP operates in silent submode – allowing ports to be added to the EtherChannel, even if it does not hear anything from the far end. This allows a switch to form an EtherChannel with a non-PAgP devices such as a network analyzer or server. It is best practice to always use non-silent mode when connecting two switches together.

Configuring a channel group with "mode on" disables negotiation and its delay. Hence, you cannot choose either channel protocol of PAgP or LACP. The port-channel virtual interface inherits the characteristics of all member ports.

An LACP system priority can be assigned to define the decision-making switch. If no priority is assigned, the switch with the lowest MAC address will be assigned.

Layer 2 and Layer 3 Port-Channels

It's important to note that EtherChannels can operate at layer 2 and 3. The configuration is a bit different between the two, so it is important to recognize what type you need before you begin your configurations. Layer 2 EtherChannel links are simply bundled switch links that acts as one logical link. This is most commonly used for trunked links between switches.

Layer 3 EtherChannel bundles allow you to create a virtual port-channel link that can be configured with an IP address. An example where this would be useful would be if you are connecting an EtherChannel bundle to a router. The router will require that its bundle has an IP address, so the virtual port-channel interface that you create can be assigned an IP address. Another example would be between multilayer switches at the distribution and core layers. Cisco recommends running layer 3 connectivity between the two and EtherChannels would assist with providing increased bandwidth and redundancy.

Layer 2 Port-Channel

```
Switch(config)# interface range f1/1 - 8
Switch(config-if)# switchport
Switch(config-if)# channel-group number mode on
```

```
Switch(config-if)# interface port-channel number
Switch(config-if)# switchport trunk encapsulation dot1q
Switch(config-if)# switchport mode trunk
Switch(config-if)# switchport trunk allowed vlan 2,56,70
Switch(config-if)# switchport trunk native vlan 99
```

Layer 3 Port-Channel

```
Switch(config)# interface range f1/1 - 8
Switch(config-if)# no switchport
Switch(config-if)# channel-group number mode on
```

```
Switch(config-if)# interface port-channel number
Switch(config-if)# no switchport
Switch(config-if)# ip address w.x.y.z s.r.t.u
```

Note that in the configuration example above how the switchport mode (trunk) and VLANs are all configured on the port-channel directly and not on the physical interfaces that make up the bundle. While it will pass traffic either way, it is much simpler to manage VLAN consistency and configuration on the bundled link. All the changes on the port-channel are trickled down to all the members.

Etherchannel Load Balancing

The bundles use an algorithm to determine each link load, so they will never be able to operate at 100% capacity of the sum of the links. That means the load will not be balanced equally amongst the individual links. A hash algorithm is used to determine which individual interface each frame is forwarded through.

The algorithm can use source MAC, destination MAC, a combination of the two, source IP, destination IP, or TCP/UDP port numbers. If only one address or port number is used for the hash, the switch uses one or more low-order bits of the hash results as an index into the bundled links. If two addresses and/or ports are hashed, the hash performs an XOR (eXclusive OR: "one or the other but not both.") on the low-order bits of the addresses or ports as the index.

To configure the EtherChannel load balancing type globally on the switch:

```
(config)# port-channel load-balance      method
```

Methods (default depends on the platform):

- src-ip source IP
- dst-ip destination IP
- src-dst-ip source and destination IP (XOR)
- src-mac source MAC
- dst-mac destination MAC
- src-dst-mac source and destination MAC (XOR)
- src-port source port
- dst-port destination port
- src-dst-port source and destination port (XOR)

Troubleshooting an EtherChannel

Remember that there should be consistent configurations on both ends of the bundle.

- If using mode “on”, make sure both ends are set to it.
- If one end is set to desirable (PAgP) or active (LACP), the other side must be set to either desirable or auto (PAgP), or to active or passive (LACP).
- Auto (PAgP) and passive (LACP) modes require the far end to request for participation.
- PAgP auto and desirable modes default to silent submode – which will establish an EtherChannel without hearing from the far end. If set to non-silent submode, packets must be received from the far end before a channel will form.

To verify the EtherChannel Status / load balancing:

```
Switch# show etherchannel {summary | load-balance}
```

To display PAgP channel-group information:

```
Switch# show pagp [channel-group-number] [neighbor]
```

To display LACP channel-group information:

```
Switch# show lacp [channel-group-number] [neighbor]
```

To verify an individual port configuration:

```
Switch# sh run interface xx/xx
```

To check for EtherChannel details and errors on an interface:

```
Switch# sh interface xx/xx etherchannel
```

To display EtherChannel debug messages:

```
Switch# debug etherchannel [all | detail | error | event]
```

Cisco
642
813

Chapter 4

Spanning Tree Protocol

Spanning Tree Protocol (STP) is designed to prevent problems related to bridging loops. STP solves the problem by blocking redundant paths and allowing only a single active path.

Spanning tree works by selecting a root switch then selecting a loop-free path from the root switch to every other switch. To do that spanning tree must choose a single root bridge, one root port for each nonroot switch, and a single designated port for each network segment.

Several different versions of Spanning Tree have been introduced over the years. Here are a few:

Common Spanning Tree (CST)

IEEE 802.1D, one instance of spanning tree runs for the entire switched network resulting in low CPU requirements, but suboptimal traffic paths when multiple VLANs are used. It is also slow to converge. Bridge ID does not include System ID Extension field (VLAN ID on 12 bits).

Per VLAN Spanning Tree Plus (PVST+)

A Cisco enhancement to CST: one instance of STP per VLAN, more resources required, slow convergence still, includes PortFast, BackboneFast, Uplinkfast, BPDU filter, BPDU guard, Root Guard, and Loop Guard enhancements. It interoperates with CST.

Rapid STP (RSTP)

IEEE 802.1W, one instance of STP, but very fast convergence time. Still suboptimal traffic flows because only a single instance for the entire switched network.

RSTP natively includes most of the Cisco proprietary enhancements to the 802.1D spanning tree, such as BackboneFast, UplinkFast, and PortFast. RSTP can achieve much faster convergence in a properly configured network, sometimes in the order of a few hundred milliseconds. Classic 802.1D timers, such as forward delay and max_age, are only used as a backup and should not be necessary if point-to-point links and edge ports are properly identified and set by the administrator. Also, the timers should not be necessary if there is no interaction with legacy bridges.

PVRST+ , aka RPVST+

A Cisco enhancement to RSTP that behaves similarly to PVST+. It supports a separate instance of RSTP for each VLAN and all instances support BPDU Filter, BPDU guard, Root Guard, and Loop Guard enhancements. This option has the largest CPU and memory requirements.

Multiple Spanning Tree (MST)

An IEEE 802.1S standard that allows you to map multiple VLANs with similar traffic flow requirements into the same rapid spanning-tree instance. MST supports RSTP for fast convergence. All instances support BPDU filter, BPDU guard, Root Guard, and Loop Guard enhancements.

Note: MST and PVRST+ have become the dominate spanning-tree protocols of choice and in Cisco switches, PVST+ is the default flavor of STP that is enabled when a VLAN is created on most switches.

Spanning Tree Protocol

STP Path Selection

Spanning tree builds the tree structure attempting to use the fastest links it has available for the active paths.

A BPDU is superior to another if it has the following:

1. Lowest bridge ID (BID) 1 => **Root Bridge (RB)**
2. Lowest path cost to the root 2/3/4 => **Root Port (RP)**
3. Lowest *upstream* bridge ID 2/3/4 => **Designated Port (DP)**
(without "upstream")
4. Lowest *upstream* port ID (PID)

Port ID

STP Definitions

Bridge ID – bridge priority + MAC Address

Bridge Priority – 0-65,535

Default Priority – 32,768

Path Cost – The cumulative cost of all links between the switch and the root bridge

Port ID – port priority + port number

Port Priority – 0-240 (default is 128, increments of 16)

	Port Priority	Port Index
Size in bits	6	10
Default value	32	Fixed unique value

STP Convergence

STA (Spanning-Tree Algorithm) implements distributed variation of the **Bellman-Ford** iterative algorithm, which could be described as a “gradient” process, meaning it repeatedly looks for the optimal solution, selecting an “optimal ” candidate every time.

1. Root bridge election

Each VLAN elects one root bridge. All ports on the root bridge act as designated ports, which send and receive traffic as well as BPDUs. The bridge with the lowest BID becomes root.

2. Root ports are determined on all non-root bridges

Each non-root bridge is assigned a single root port that sends and receives traffic. The root port is chosen based on the port with the lowest-cost path between the non-root bridge and the root bridge. If two paths are equal cost, the lowest **upstream** BID is chosen, and in case it's the same switch for all redundant paths, the **upstream** port with the lowest port ID will win.

3. Designated port selection

Each segment has a single designated port. Designated ports are chosen **between switches** from non-root ports that have the lowest path cost to the root bridge. In the event of a tie, the bridge ID acts as a tiebreaker (lowest wins). In the event of a tie, the port ID acts as a tiebreaker (lowest wins).

All ports on a root bridge are designated ports.

STP Port Roles

Root Port (RP)

Always an upstream facing interface and always facing a DP

- On non-root bridges only
- Receives and forwards traffic towards the root bridge as needed
- **Mostly receives BPDUs** and sends very few of them
- Only one per switch (and per VLAN in PVST+)
- Can populate the MAC table

Designated Port (DP)

Always a downstream facing interface

- On root and non-root bridges
- All ports on root bridge are designated ports
- Receives and forwards traffic towards the root bridge as needed
- **Mostly sends BPDUs** and receives very few of them
- Only one per segment (and per VLAN in PVST+)
- Can populate the MAC table

Non Designated Port

Always an upstream facing interface

- Does not forward nor receives traffic (blocking), *except for BPDUs*
- **Mostly receives BPDUs** and sends very few of them
- Does not populate the MAC table

Disabled port

- A port that is shut down

STP Port States

Disabled

- Does not participate in STP
- Does not forward nor receive frames, including BPDUs

Blocking

Always an upstream facing interface

- Absolutely needs to receive BPDUs to remain blocking and to determine root switch
- Not RP nor DP
- In non designated role and does not forward nor receives traffic
- Default 20 seconds in this state before trying to converge (max-age)

Listening

- Receives and sends BPDUs
- 15 seconds (forward delay)

Learning

- Populates the CAM table
- 15 seconds (forward delay)

Forwarding

- RP or DP
- Part of the active topology
- Forwards and receives frames
- Sends and receives BPDUs

STP Link Cost

Spanning-tree uses a link cost calculation to determine the RPs and DPs.

It is calculated by adding the costs of all links between the root bridge and the local switch.

Ethernet Speeds	802.1D Costs	802.1T Costs
10 Mbits/s	100	2.000.000
100 Mbits/s	19	200.000
1 Gbits/s	4	20.000
10 Gbits/s	2	2.000

STP BPDU Types

Configuration

Used for STP Computation

Sourced by RB to advertise Bridge and link attributes every 2 s as Hellos:

- Root BID
- Root path cost
- Sender BID
- Sender Port ID
- Timers
- Topology Change Flag not set
- Topology Change Ack not set

Topology Change Notification (TCN)

Used to advertise network topology modifications

- Configuration BPDU with Topology Change Flag set
- Used to quickly age out the CAM table of all switches in case of a port state/topology change

Topology Change Acknowledgement (TCA)

Used to acknowledge the receipt of a TCN

- Configuration BPDU with Topology Change Ack set

BPDUs in PVST+

STP Timers

The default Hello timer is 2 s, the default Max Age timer is 20 s (10 Hellos) and the default Forward Delay is 15 s.

They are all advertised and set by the current root bridge for the whole layer 2 (VTP) domain. They may change as the root bridge fails over to another switch.

STP Reconvergence

The BPDU (Bridge Protocol Data Unit) TCN is sent by a switch through its RP as soon as it detects a port state/topology change.

Upstream switch sends a TCA back.

Upstream switch sends a TCN out its RP.

Next upstream switch sends a TCA back.

Next upstream switch sends a TCN out its RP.

Process goes on until the RB receives the TCN.

The RB replies with a TCN out all its ports for a certain period of time (**max_age + forward_delay**), relayed by all downstream switches.

The result is that the CAM aging time is reduced to "**Forward delay**" in all switches: from 5 mn to 15 seconds during 35 seconds (by default).

This whole process does not happen if the state of a port changes while it is configured with the **portfast** feature.

BPDUs in PVST+

From a Peter Paluch's post:

If the native VLAN is VLAN1 then:

- VLAN1 standard STP BPDU is sent **untagged**
- VLAN1 PVST+ BPDU is sent **untagged**
- Other VLANs PVST+ BPDUs are sent **tagged with their appropriate VLAN**

If the native VLAN is different from VLAN1 then:

- VLAN1 standard STP BPDU is sent **untagged**
- VLAN1 PVST+ BPDU is sent **tagged with VLAN1**
- Other VLANs PVST+ BPDUs are sent **tagged accordingly (the one for the native VLAN will be untagged)**

PVST+ Configuration

Enable PVST+ globally (default):

```
Switch(config)# spanning-tree mode pvst
```

Designate and configure a primary root bridge:

```
Switch(config)# spanning-tree vlan 2 root primary
// Default: the switch priority is set to 24576
or
Switch(config)# spanning-tree vlan 2 priority 0
```

Designate and configure a secondary root bridge:

```
Switch(config)# spanning-tree vlan 2 root secondary
// Default: the switch priority is set to 28672
or
Switch(config)# spanning-tree vlan 2 priority 4096
```

Disable BPDUs and STP instance for some VLANs:

```
// From the whole switch:
Switch(config)# no spanning-tree vlan vlan-range
// From this trunk:
Switch(config-if)# switchport trunk allowed vlan remove vlan-list
```

Verify the configuration

```
Switch# show spanning-tree [vlan id] [interface id] [brief | detail | summary] [root | bridge]

// Cost of Root port: Total path cost to the RB
// Port path cost: Cost of the local link
// Designated path cost: Total path cost to the RB as advertised by the DB on that link
// Designated Port ID Port ID as advertised by the DB on that link
```

Change Port Cost:

```
Switch(config-if)# spanning-tree [vlan id] cost value
```

Change Port Priority:

```
(config-if)# spanning-tree [vlan id] port-priority value
// value: Number from 0 to 240, in increments of 16. The
// lower the number, the higher the priority.
```

Change Timers on RB:

```
(config)# spanning-tree [vlan id] hello-time seconds
(config)# spanning-tree [vlan id] forward-time seconds
(config)# spanning-tree [vlan id] max-age seconds
// The range for the timers are:
// - hello-time: 1 - 10 seconds
// - forward-time: 4 - 30 seconds
// - max-age: 6 - 40 seconds
```

STP Load Balancing

The default STP mode on current Cisco switches is PVST+. That has three major implications by default:

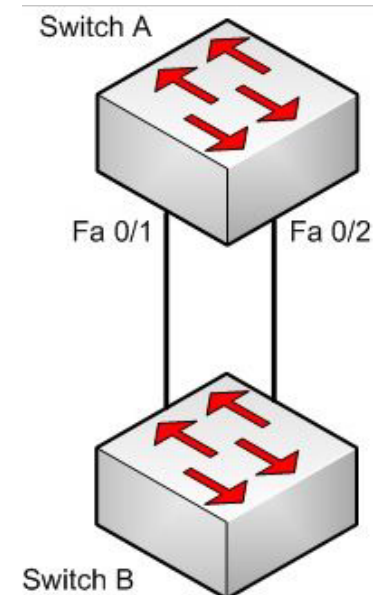
- it means that all VLANS will elect the same root bridge which will be the one with the lowest MAC address, generally the **oldest switch!**
- a port state / topology change will impact all VLANs in exactly the same way
- all redundant links will also be blocked in exactly the same manner

One way to use STP to load balance across redundant uplinks between two switches is to **change the local port cost or upstream port priority for the active VLANs** to intentionally force some VLANs to prefer one link and others to prefer other links. By lowering the local port cost or upstream port priority for a VLAN on a redundant link interface, traffic for that VLAN would begin to use that link and place the other uplink into the blocking state.

Example:

```
Switch A# conf t
Switch A(config)# interface fa 0/1
Switch A(config-if)# spanning-tree vlan 1-10 port-priority 16
Switch A(config-if)# switchport mode trunk
Switch A(config-if)# interface fa 0/2
Switch A(config-if)# spanning-tree vlan 11-20 port-priority 16
Switch A(config-if)# switchport mode trunk
```

In this example, where Switch A is the **upstream switch**, VLANs 1-10 would traverse the left link (priority of 16 is less than default of 128) and use the right link as a backup only, while VLANs 11-20 would prefer the right uplink and use the left link as a backup only. This way both uplinks are being used, but only one for each VLAN. Make sure you understand how this works because this is a very common implementation design.



STP Enhancements

PortFast

Spanning Tree Portfast causes layer 2 switch interfaces to enter forwarding state immediately, bypassing the listening and learning states. It should be used on ports connected directly to end hosts like servers or workstations.

Also, the switch does not generate any TCN when something changes on a portfast access port, leaving the CAM tables unchanged in the entire layer 2 domain.

Note: If Portfast isn't enabled, DHCP timeouts can occur while STP converges, causing more problems.

To configure PortFast on an interface:

```
Switch# conf t
Switch(config)# int fa 3/1
Switch(config-if)# [no] spanning-tree portfast [disable | trunk]
// disable: disables portfast on this interface (when portfast is enabled globally)
// trunk: enables portfast on this trunk interface connected to a router or server
```

PortFast can be configured globally on an access switch for all interfaces to save configuration time. Also, it only applies to access interfaces, not trunks.

To configure PortFast globally on all non-trunking interfaces:

```
Switch# spanning-tree portfast default
```

To verify PortFast:

```
Switch# sh spanning-tree int fa 3/1 portfast
Switch# sh spanning-tree summary
```

To configure switch port Host Mode:

To configure **PortFast** and **access mode** (disables trunking negotiation on an interface) and to **disable channeling**:

```
Switch (config-if)# switchport host
```

Connecting Switch Ports to PortFast Switch Ports (STP or RSTP)

Static Trunk → PortFast Access:

- The access port is **blocked into an inconsistent type**
- **No risk of loop**
- BPDU Filter feature applied at the port level could be used to avoid this inconsistent state
- BPDU Guard feature is useless in this situation

Dynamic Trunk → PortFast Access:

- The access port **loses its PortFast feature, becomes a normal STP or RSTP port, is eligible to all port roles (DP, RP, ...) and generates a topology change notification (TCN)** - both ports end up as Access.
- **No risk of loop**
- BPDU Filter and BPDU Guard could be used, depending on the local security policy

Static / Dynamic Trunk → PortFast Static / Dynamic Trunk:

- **Some loops may appear** in a redundant environment between *switches* only. This configuration (spanning-tree portfast trunk) should be avoided when facing another switch.
- **No risk of loop** with connections between switches and legitimate **Servers or Routers**.
- BPDU Filter and BPDU Guard could be used when the switches are connected to Servers / Routers, depending on the local security policy

UplinkFast

UplinkFast is applied at the access layer and provides a mechanism to very quickly (saving 30 seconds) have a secondary uplink to the distribution layer if the current STP uplink fails. This is similar to RSTP alternate port role and only addresses a direct link failure.

In addition, that switch floods its CAM table out the alternate port.

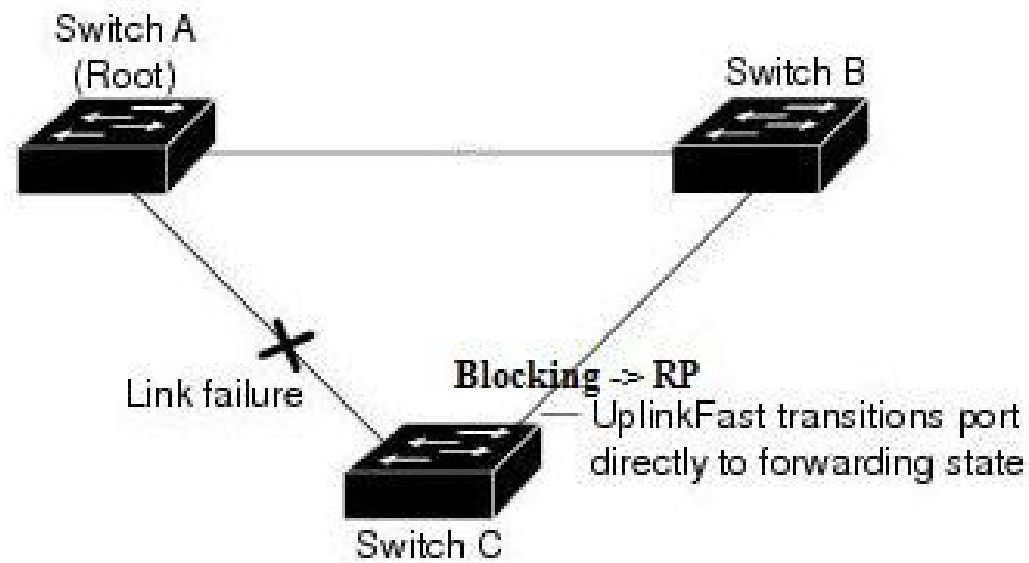
At last, it also changes all interfaces costs to a very high value, in order to "persuade" downstream switches to use another switch for their path to the root since it appears to have at least one unstable link.

To enable Uplinkfast on a switch:

```
Switch(config)# spanning-tree uplinkfast
```

To verify UplinkFast globally:

```
Switch# sh spanning-tree uplinkfast
Switch# sh spanning-tree summary
```



BackboneFast

BackboneFast is often applied at the distribution and core layers and was developed to fix STP convergence slowness on indirect backbone links. UplinkFast is designed to detect direct failures, whereas BackboneFast is designed to detect indirect failures. An indirect failure is not immediately detected when it occurs, and under normal STP operation, the Max Age timer is used to detect it. That means about 50 seconds of STP convergence.

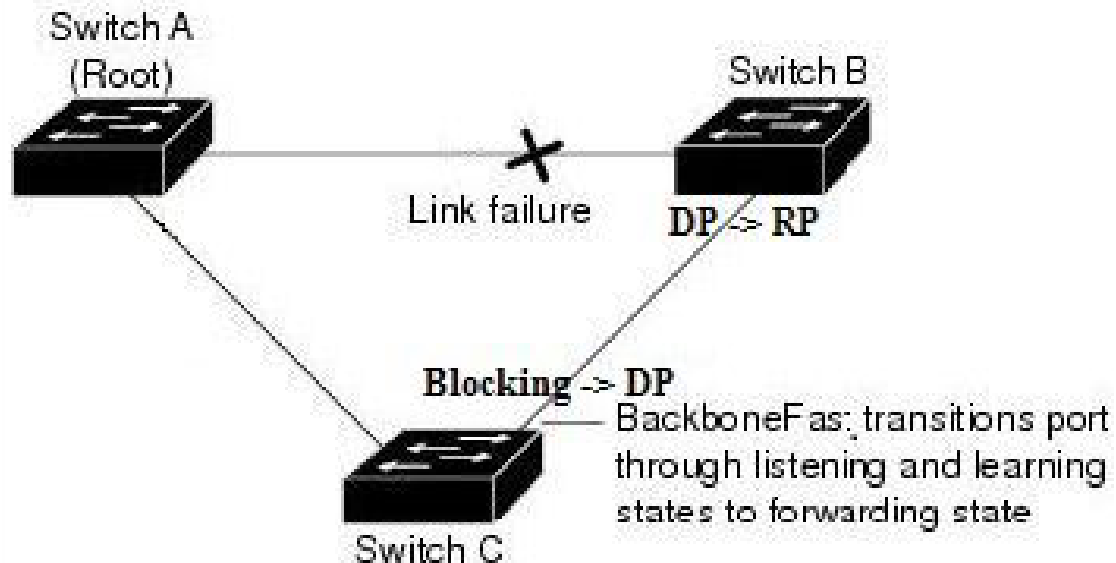
BackboneFast effectively eliminates the Max Age timeout period associated with an indirect failure, lowering convergence from the default 50 seconds to 30 seconds.

To enable Backbonefast on a switch:

```
Switch(config)# spanning-tree backbonefast
```

To verify BackboneFast globally:

```
Switch# sh spanning-tree backbonefast
Switch# sh spanning-tree summary
```



BPDU Filtering

When enabled globally on the switch: PortFast Toggle

- Configures all PortFast ports for pseudo BPDU filtering
- If BPDUs are seen, the port loses its PortFast status, BPDU filtering is disabled, and STP resumes default operation on the port
- When the port comes up, it sends 10 BPDUs, if it hears any BPDUs during that time PortFast and BPDU filtering are disabled

*Notes: If you enable BPDU Guard and BPDU filtering on the same interface, BPDU Guard has no effect because BPDU filtering has precedence over BPDU Guard.
It is possible to configure BPDU Filtering on a NON-PortFast port.*

When applied to an individual port: True BPDU filtering

- It ignores all BPDUs it receives
- It does not transmit BPDUs

To enable BPDU filtering globally on the switch:

```
Switch(config)# spanning-tree portfast bpdupfilter default
```

To enable BPDU filtering at the interface level:

```
Switch(config-if)# spanning-tree bpdupfilter {enable | disable}
```

To verify:

```
Switch# show spanning-tree summary  
Switch# show spanning-tree interface fa 0/3 detail
```

BPDU Guard

Prevents problems related to switches accidentally being connected to PortFast-enabled ports. Bridging loops could normally instantly occur on static or dynamic Trunk ports (not on Access Ports). It places the port in **err-disabled state** if it receives a BPDU - disabling the interface.

Note: it is possible to configure BPDU Guard on a NON-PortFast port.

To enable BPDU Guard globally on the switch on all portfast enabled ports:

```
Switch(config)# spanning-tree portfast bpduguard default
```

To enable BPDU Guard at the interface level:

```
Switch(config-if)# spanning-tree bpduguard {enable | disable}
```

To verify:

```
Switch# show spanning-tree summary  
Switch# show spanning-tree interface fa 0/3 detail
```

Root Guard

Root guard was developed to control where root bridges can be located within the network. Switches learn about and elect root bridges based on BPDUs they receive, so if a new switch is added to the environment with a lower bridge ID than the current root bridge, the new switch will become root – and in turn disrupt your carefully planned traffic patterns.

To prevent this from occurring, root guard can be applied to the interface where a root bridge should never be seen.

When root guard is applied to an interface, it forces the port to essentially always remain a DP, never allowing it to transition to a root port.

If a root guard-enabled port receives a superior BPDU, it immediately moves the port to a **root-inconsistent STP state** (essentially the same as the listening state) and does not forward any traffic out that port.

When the root guard protected port stops receiving superior BPDUs, it automatically unblocks the port and proceeds through its normal listening, learning, and eventually forwarding states. No intervention is required.

To enable root guard on an interface:

```
Switch(config)# int fa 4/4  
Switch(config-if)# spanning-tree guard root
```

To verify:

```
Switch# show spanning-tree inconsistentports  
Switch# show spanning-tree interface fa 0/3 inconsistency  
Switch# show spanning-tree interface fa 0/3 detail
```

Loop Guard

Most bridging loops that occur when STP is active happen when a port in blocking state stops receiving BPDUs on the interface and therefore transitions the port to forwarding state – creating an all-ports-forwarding loop. Loop Guard blocks ports on a **per-VLAN basis**, so on trunks it will only block that VLAN – not the whole trunk.

Loop guard should be applied to all **non-DP** (RP, Alternate Port).

Loop guard is most effective when it is configured on the entire switched network. Loop guard is automatically applied to all of the active VLANs or instances to which that port belongs; beware of the fact that the same trunk port might be DP for one VLAN and non-DP for another one.

When the switch is operating in PVST+ or RPVST+ mode, loop guard prevents Alternate and RPs from becoming DPs. If BPDUs are not received on a non-DP, and loop guard is enabled, that port is moved into the blocking **loop-inconsistent STP state**, instead of the listening / learning / forwarding state.

Follow these guidelines when using loop guard:

- do not enable loop guard on PortFast-enabled ports and the other way around.
- do not enable loop guard if root guard is enabled.
- do not enable loop guard on ports that are connected to a shared link

To enable loop guard globally on the switch:

```
Switch(config)# spanning-tree loopguard default
```

To enable loop guard on an interface:

```
Switch(config)# int fa 4/4  
Switch(config-if)# spanning-tree guard loop
```

To verify:

```
Switch# show spanning-tree summary  
Switch# show spanning-tree inconsistentports  
Switch# show spanning-tree interface fa 0/3 inconsistency  
Switch# show spanning-tree interface fa 0/3 detail
```

UDLD

UDLD (UniDirectional Link Detection) is another loop-prevention mechanism for STP. It tries to discover improper cabling or unidirectional links on Layer 2 before they grow into bridging loops. This situation is much more common in fiber optic networks where there is a physical Rx/Tx pair and a situation can arise where one is not functioning correctly.

STP relies on constant and consistent reception of BPDU messages. If a switch stops receiving BPDUs on a blocking port, STP ages out the information for the port and transitions it into forwarding state. This might lead to a loop.

UDLD sends UDLD protocol packets to its neighbor switch – every 15 seconds is the default. The neighbor is then expected to echo the packets before a timer expires. If the switch does not hear a reply it waits for, it shuts down the port.

There are two UDLD modes:

Normal – UDLD simply places the port into an undetermined state if it stops hearing responses from its directly-connected neighbor

Aggressive (Preferred) – Tries to re-establish the connection up to 8 times, then puts the port in **err-disabled state** (essentially shutting down the port)

To enable UDLD on an interface:

```
Switch(config)# int fa 4/4
Switch(config-if)# udld port [aggressive]
```

To verify UDLD:

```
Switch# show udld [interface-id]
```

To enable UDLD globally on all fiber ports:

```
Switch(config)# udld {enable | aggressive}
```

To reset all interfaces disabled by UDLD:

```
Switch# udld reset
// permit traffic to begin passing through them again
```

- Notes:
- While both loop guard and aggressive UDLD have many overlapping functions, enabling both provides the best protection.
 - In an **EtherChannel**, in case of a unidirectional port, **UDLD disables only that port for all VLANs, whereas Loop Guard blocks the whole port channel into loop-inconsistent state for only the failing VLAN(s)**. On EtherChannel, UDLD alone seems a much better solution.

Storm Control

By default, the traffic is filtered on port(s) where a storm is detected. But the action can be configured to disable the port or send an SNMP trap:

```
Switch(config-if)# storm-control {{broadcast | multicast | unicast} level {level | bps bps | pps pps}} | {action {shutdown | trap}}
```

STP Enhancements Summary

Enhancements	Configuration	Verification	Error State
PortFast	1)#spanning-tree portfast [disable trunk] 2)#spanning-tree portfast default	1)#show spanning-tree interface portfast 2)#show spanning-tree summary	N/A
UplinkFast	1)N/A 2)#spanning-tree uplinkfast	1)N/A 2)#show spanning-tree uplinkfast #show spanning-tree summary	N/A
BackboneFast	1)N/A 2)#spanning-tree backbonefast	1)N/A 2)#show spanning-tree backbonefast #show spanning-tree summary	N/A
BPDU Filter	1)#spanning-tree bpdufilter {enable disable} 2)#spanning-tree portfast bpdufilter default	1)#show spanning-tree interface detail 2)#show spanning-tree summary	N/A
BPDU Guard	1)#spanning-tree bpduguard {enable disable} 2)#spanning-tree portfast bpduguard default	1)#show spanning-tree interface detail 2)#show spanning-tree summary	(Interface) err-disabled

* : 1) Interface
 2) Globally

Root Guard	1)#spanning-tree guard root 2)N/A	1) #show spanning-tree interface detail #show spanning-tree inconsistentports #show spanning-tree interface inconsistency 2)N/A	(STP) root-inconsistent
Loop Guard	1)#spanning-tree guard loop 2)#spanning-tree loopguard default	1)#show spanning-tree interface detail #show spanning-tree inconsistentports 2)#show spanning-tree summary	(STP) loop-inconsistent
UDLD°	1)#udld port [aggressive] 2)#udld {aggressive enable}	1)#show udld interface-id 2)#show udld	(Int-Aggressive Mode) err-disabled
Storm Control	1)#storm-control {(broadcast multicast unicast) level { ~~and~~ bps ~~bps~~ pps ~~pps~~ (action {shutdown trap}) 2)N/A	1)#show storm-control 2)N/A	(Int-Shutdown Action) err-disabled

Switch# show interfaces status err-disabled

// Displays interfaces in err-disabled state and the reason why, for example:

// BPDU Guard violation, UDLD detection, Storm control rate limit violation, Port Security violation, DHCP snooping

// rate limit violation, Dynamic ARP inspection rate limit violation, ...

*: 1) Interface
2) Globally

° not stricto sensu an STP enhancement

STP Best Practices

Something to consider with spanning tree is the lack of multipathing options. STP eliminates loops by creating a tree structure where a single link is created to each switch. This means that even with all the redundant links you put in place, STP will always only allow one – reducing much of your available bandwidth. Because of this and other limitations, it is recommended to use layer 3 at both the distribution and core layers. Using layer 3 between the distribution and core allows you to use multipathing (*up to 32 paths*) using Equal-Cost MultiPathing (ECMP) without the dependency of STP (and Unequal-Cost MultiPathing with EIGRP).

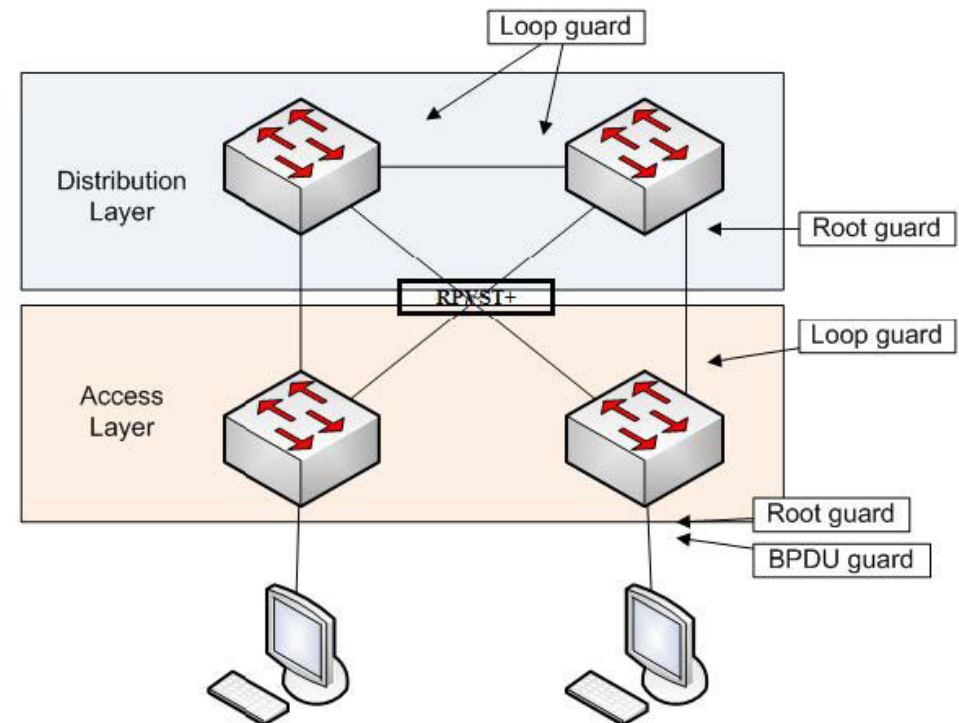
Also, the new Nexus 7K allows layer 2 multipathing with two links using virtual port channels: the virtual PortChannel (vPC) feature allows one end of a PortChannel to be split across a pair of Cisco Nexus 7000 Series Switches.

- Because a 50 second network convergence delay is usually not acceptable in modern networks, RPVST+ or MST is preferred.
- PVST+ with portfast or RPVST+ should absolutely be used on the network edge to prevent user/wiring errors from propagating throughout the network.
- A root bridge should be manually assigned in every STP topology
- If using PVST+ or RPVST+, assign a root bridge for each VLAN using the command:

```
#spanning-tree vlan-ID root [primary | secondary]
```

- If using HSRP, make sure the STP root bridge and HSRP active router are assigned to the same device if possible.
- Use the STP Enhancements (sometimes referred to as the STP toolkit) to optimize the topology.
- **Loop guard** - Implement on all (non portfast and non root guard) layer 2 ports between access, distribution and core layers.
- **Root guard** - Implement on downstream switch ports.
- **BPDU guard / Filtering**- Implement on access ports connected to end devices.
- **UDLD** - Cisco recommends implementing UDLD on all point-to-point links between switches. I recommend to implement it alone on EtherChannels.

STP Enhancement Recommendations



Troubleshooting STP

2/26/15

One or a few loops form because of the lack of reception of BPDUs in a redundant area or because of a switch redundantly connected to another one through PortFast ports

<=

- 3 Trunk Misconfigurations
- 2 Trunk Physical Issues
- 1 VACL Misconfiguration
- 1 CPU Load Issue

And some ports could end up in an inconsistent state

<=

5 Types of Inconsistencies

3 Trunk Misconfigurations

Duplex Mismatch on Switch Trunk Ports

In this scenario, one side of a link (downstream) is set to half duplex and the other is set to full. If the full duplex side sends a lot of traffic to the half duplex interface, that last interface will experience collisions when it attempts to transmit STP BPDUs. The full duplex interface may therefore never receive them during max-age:

- If that upstream interface is a RP, STP will transfer it to a DP role and change its alternate port to RP -> creating a loop.
- If that upstream interface is an alternate port, STP will transfer it to a DP role -> creating a loop.

BPDU Filtering on Switch Trunk Ports

When a non-DP port or its upstream port is configured with BPDU Filtering:

- if the non-DP is RP, its role is transferred to DP and its alternate port is changed to RP -> creating a loop.
- if the non-DP is Alternate, that interface moves directly into DP forwarding state -> creating a loop.

PortFast on Switch Trunk Ports

PortFast interfaces move directly into forwarding state, so if some **switch** static or dynamic Trunk ports get connected to some **switch** static or dynamic Trunk ports configured with PortFast (**spanning-tree portfast trunk**) in a redundant area, a loop might form.

Notes:

- Connecting switch static Trunk ports to switch PortFast Access ports is not a 'loop maker' since the latter ports will be blocked into an inconsistent state.
- Connecting switch dynamic Trunk ports to switch PortFast Access ports is not a 'loop maker' either since the latter ports will lose their PortFast ability and become normal (R)STP ports. This could become a security situation though ; BPDU Filter/Guard could be helpful here.

2 Trunk Physical Issues

2/26/15

Unidirectional link failure

This occurs when a hardware failure causes a normally two-way link to become a one-way link. The potential loop problem is the same as with the duplex mismatch issue, with one side moving from blocking to forwarding because they stop receiving superior BPDUs on the interface, so the switch believes its port is the DP.

Aggressive UDLD can prevent loops from forming when this occurs by putting the offending port into **err-disabled** state. Cisco recommends using aggressive UDLD on all point-to-point links in a switched environment.

Frame Corruptions

This is a very uncommon cause of STP loops, but it exists when errors on an interface do not allow BPDU frames from being received. Again, a port moves from blocking to forwarding because they stop receiving superior BPDUs on the interface. This could be caused by a bad cable, or incorrect cable length.

1 VACL Misconfiguration

VACL Filtering BPDUs

A VACL filters all traffic by default at the end of the VLAN list, including (R)STP BPDUs.

To avoid this situation, you need to allow all traffic at the end of the VACL, or specifically allow each type of traffic within the VLAN list, cf. [VACL](#).

1 CPU Load Issue

Over-Utilization of CPU

If for any reason the CPU of a switch is over-utilized, there exists the possibility that it will be unable to send out BPDUs. STP is generally not very resource intensive, but be careful when running PVST+ / RPVST+.

5 Different Types of Inconsistencies

Inconsistent State Ports

- **Type inconsistency:** A (R)PVST+ BPDU is received on an access port.
- **Port VLAN ID (PVID) inconsistency:** A (R)PVST+ BPDU is received on a different VLAN than it was originated (different native VLAN ids on a trunk)
- **EtherChannel inconsistency:** One side of the EtherChannel is a port-channel and the other side is not.
- **Root inconsistency:** This is detected by the Root Guard feature
- **Loop inconsistency:** This is detected by the Loop Guard feature.

General STP Troubleshooting Methodology

2/26/15

Develop a Plan

In order to make a plan, you must know the following parts of the network:

- The switched topology
- The location of the root bridge(s)
- The location of blocking ports

Isolate the Cause(s) and Solve the Issue(s)

Using a Packet Sniffer

The best way to determine a loop is to capture packets on a saturated link and look for duplicate packets, but it may take too much time.

Using Common Symptoms

Some common symptoms include consistent flapping of MAC values because MAC addresses should not flap, HSRP may complain of duplicate IP addresses or the CPU may become overloaded. (near 100%).

Using the CLI

1) Blocking port(s) which have been turned into DP/RP:

Display all the ports that are currently blocked by the spanning tree algorithm and check them against the topology:
the missing blocked port(s) is/are responsible for the current loop(s).

```
Switch# show spanning-tree blockedports
```

• Duplex Mismatch on Switch Trunk Ports

```
Switch# show interface fa 1/0/12
```

• Unidirectional Link Failure

```
Switch# show udld [f1/12]
```

• VACL Filtering BPDUs

```
Switch# show vlan access-map [vACL-name]
```

2) Portfast on Switch Trunk Ports

```
Switch# show spanning-tree interface fa 1/5 portfast
```

3) Inconsistent Ports:

```
Switch# show spanning-tree inconsistentports
```

• BPDU Filtering on Switch Trunk Ports

```
Switch# show spanning-tree interface fa 1/6 detail
```

• Frame Corruptions

```
Switch# show interface fa 1/9
```

• Over-utilization of CPU

```
Switch# show process cpu sorted
```

Restore Connectivity Urgently

Most of the time administrators do not have the luxury of time to identify the root cause of a loop, instead they must stop it as quickly as possible. Here are some options:

- Shutdown the ports displayed on console logged messages about MAC addresses flapping.
- Disable every port that is providing redundancy, starting with the most affected areas of the network. Try to disable ports you know should be in blocking state if possible.
- If it is difficult to pin down, increase the level of STP logging on the switches. The loops form when a port moves into forwarding state, so it can later be identified.

Try this:

```
Switch# debug spanning-tree events
```

To log the events:

```
Switch(config)# logging buffered security-level | size-in-bytes
```

Document Findings

It is important to document both your findings and any changes to the network after the dust clears. Current and detailed documentation also reduces troubleshooting time in the future.

Flex Links → Alternative to STP

Flex Links are a pair of a Layer 2 interfaces (ports or port channels), where one interface is configured to act as a backup to the other. Flex Links are typically configured in service-provider or enterprise networks where customers do not want to run STP. Flex Links provide link-level redundancy that is an alternative to Spanning Tree Protocol (STP). STP is automatically disabled on Flex Links interfaces.

Flex Links are supported only on Layer 2 ports and port channels, not on VLANs or on Layer 3 ports.

To configure the Flex Links feature, you configure one Layer 2 interface as the standby link for the link that you want to be primary. With Flex Links configured for a pair of interfaces, only one of the interfaces is in the linkup state and is forwarding traffic. If the primary link shuts down, the standby link starts forwarding traffic. When the inactive link comes back up, it goes into standby mode.

Rapid Spanning Tree

Rapid Spanning Tree Protocol (IEEE 802.1W) was introduced to dramatically speed up STP convergence when network changes occur. RSTP can revert to 802.1D (common spanning-tree) to inter-operate with legacy bridges on a per-port basis. A rapid version of PVST+, RPVST+ is a per-VLAN implementation of rapid spanning-tree.

RSTP Port States

Discarding

- Merges the former disabled, blocking, and listening states
- Prevents the forwarding of frames
- Seen in both stable/active and synchronization/changes

Learning

- Receives frames to populate the MAC table
- Seen in synchronization/changes

Forwarding

- Forwarding ports determine the active topology
- An agreement process between switches occurs before frames can be forwarded
- Only seen in stable/active topologies

Note: In every RSTP port state, BPDU frames are accepted and processed.

Operational Status	STP Port State	RSTP Port State	Port Included in Active Topology
Disabled	Disabled	Discarding	No
Enabled	Blocking	Discarding	No
Enabled	Listening	Discarding	No
Enabled	Learning	Learning	Yes
Enabled	Forwarding	Forwarding	Yes

RSTP Port Roles

Root Port

Always an upstream facing interface and always facing a DP

- On non-root bridges only
- Best port towards the root bridge
- Only one per switch, per VLAN in PVRST+ or per instance in MST
- Is always in forwarding state in an **active**/stable topology
- Mostly **receives** BPDUs and sends very few of them

Designated Port

Always a downstream facing interface

- On root and non-root bridges
- All ports on root bridge are designated ports
- Receives and forwards frames as needed
- Only one per segment, per VLAN in PVRST+ or per instance in MST
- Is always in forwarding state in an **active**/stable topology
- Mostly **sends** BPDUs and receives very few of them

Disabled Port

- No role in spanning tree

Alternate Port

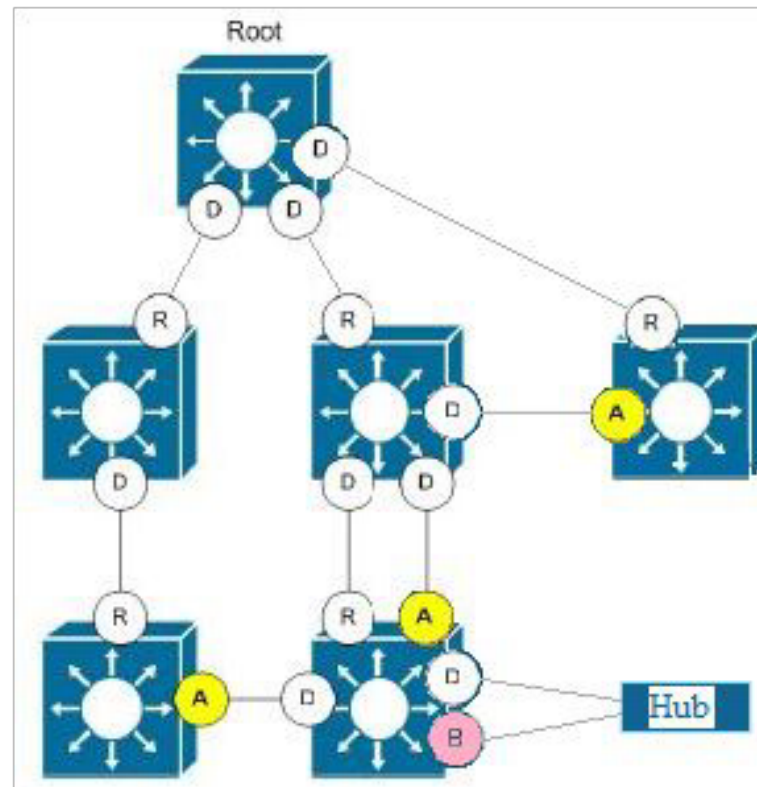
Always an upstream facing interface and always facing a DP

- Offers an alternate path towards the root bridge, but is in **discarding** state in an **active** topology
- Present on non-root switches and becomes RP if the other RP path fails
- If the RP fails and there's no Alternate Port, that switch declares itself as a RB and syncs this information downstream
- Mostly **receives** BPDUs and sends very few of them

Backup Port

Always a downstream facing interface

- An additional switch port on a redundant (and **shared**) link
- It has a higher port ID than its redundant peer port, so it assumes the **discarding** state
- Mostly **receives** BPDUs and sends very few of them



RSTP Link Type

In common spanning tree, it took 50 seconds maximum before a port could be placed in forwarding state after a network change. RSTP biggest advantage is its ability to rapidly transition alternate ports to a forwarding state. To do this, the protocol relies on two variables, **link type** and **edge port**.

Link type can be **point-to-point** or **shared** (between switches).

- Determined by duplex mode of port
 - Full Duplex – assumed to be point-to-point
 - Half Duplex – assumed to be shared
- Point-to-point links are considered candidates for rapid transition to forwarding state
- Link type can be manually configured if desired

The link types cannot be determined until the port role is first established.

Root Ports

- Don't use the link type parameter
- Make rapid transition to forwarding state as soon as it receives a BPDU from the root bridge and puts non-designated ports in discarding state (called sync)

Alternate and Backup Ports

- Do not use link type in most cases
- Simply go through RSTP operation process

Designated Ports

- Most common use of link type parameter
- Only allows rapid transition to forwarding if point-to-point

RSTP Edge Ports

Edge ports are assumed to be connected to an end host and never another switch. Edge ports immediately transition to rapid forwarding state when enabled.

- the RSTP equivalent of **PortFast**
- allowed to transition directly into forwarding state only if it is point-to-point
- designated through manual configuration
- Does not generate a topology change notification when link transitions to enabled or disabled status
- **If an edge port receives a BPDU from a dynamic trunk port, it loses its edge port status, becomes a normal RSTP port and generates a topology change notification (TCN).**

RSTP Topology Changes

In 802.1D spanning tree, when a switch detects a topology change, it first notifies the root bridge. The root bridge then sets the TC (topology change) flag on the BPDUs it sends out, which gets relayed throughout the switched network.

When a switch receives the notification, it reduces its MAC table aging time to the forward delay, down from 5 mn. That allows the outdated topology information to be flushed from the switches. This model works well, but there are 2 problems:

- 1) it takes a minimum of twice the forwarding delay (30 s) for switches to transition back to forwarding state.
- 2) the MAC address tables flushing should result in new unicast floodings by all the switches. This can cause large layer 2 networks some major issues related to reconvergence after one minor change in the network (one host entering/leaving) or one major topology change (trunk port going up/down) .

RSTP solves this. In RSTP, **only non-edge ports that are transitioning to forwarding state cause a TCN** to be sent out all non-edge ports and **all switches dump their CAM table associated with these ports**, except the one that generated the TCN.

This means that a loss of connectivity is not considered as a topology change any more, contrary to 802.1D

- that is, a port that moves to blocking no longer generates a TCN.

If a switch fails to receive 3 BPDUs in a row from a neighbor, it is certain that the connection to that neighbor is lost and protocol information can be immediately aged out, as opposed to STP where the problem might have been anywhere on the path to the root.

RSTP Timers

The default Hello timer is 2 s, the default Max Age timer is 6 s (3 Hellos) and there's no more any Forward Delay.

They are all advertised and set by the current root bridge for the whole layer 2 domain. They may change as the root bridge fails over to another switch.

Synchronization (Proposal & Agreement Process)

Synchronization is a term used to describe the RSTP network convergence process. It is also called '**Proposal and agreement process**'.

It is the Same as for STP, but on a hop-by-hop basis.

It happens during network bootup or when there is a change in the root bridge.

When better root bridge information is received, or root port changes, the local bridge **blocks all non-edge DPs**

which become **discarding**. Effectively, those are the ports connected to the bridges that are supposed to be **downstream** to the new root.

The local bridge sets **Proposal flag** in the outgoing BPDU sent out of the potential DPs.

oDownstream bridges, receiving the Proposal messages, compare the RB information there with the locally known information. If the information is superior, every downstream bridge elects a single RP, and then blocks all downstream ports. After this, every downstream bridge sends a single response BPDU upstream, with the **Agreement flag** set.

oAfter receiving an Agreement message, the upstream bridge unblocks its downstream ports and continues forwarding. The blocked ports are now moved to the downstream bridges, which continue the synchronization process, until all bridges in the topology are affected.

oEventually, the Sync process stops when either the leaf bridges have no further downstream ports to propagate Proposals, or the process reaches back to the upper part of the tree, where bridges reject the Proposals, as they have better paths to the root bridge.

oWhen a bridge rejects a Proposal it has better root bridge information. In this case, it sends back alternate proposal message with its own RB information. The bridge that has been rejected, adapts to the new information, and either blocks the port where the alternate proposal has been received or elects it as a new RP. In the first case, the synchronization process is not continued any further. In the second case, the synchronization “ripple” bounces back and goes in the opposite direction.

- The use of point-to-point links is critical to RSTP sync process.
- Downstream port blocking is essential in preventing transient loops during RB information change.
- Sync process ensures all switches agree on the same RB.

This 'Proposal and agreement process' is non-deterministic, and depends on the topology:

- very meshy topologies converge relatively slowly,
- very large topologies converge relatively slowly,
- RB failures may cause slow convergence time and temporary loops.

To ensure fast convergence:

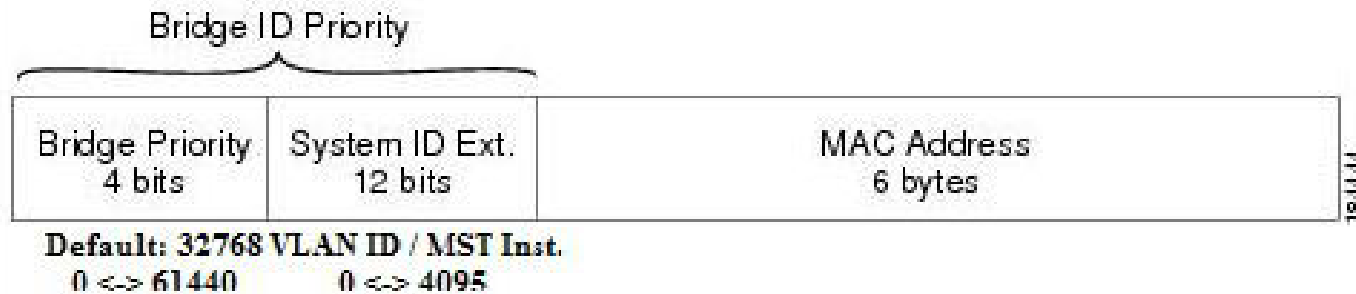
- + keep topology size small (3 ↔ 5 switches) □ avoid L2 only flat domain
- + avoid excessive redundancy,
- + rely on physical layer failure detection instead of the Hello BPDUs.

Bridge IDs

In 802.1D, each switch was required to have a unique bridge ID, consisting of a priority value + MAC address. PVST+ and PVRST+ also require the BID, but they must also include VLAN information within the BID because a unique instance must run for each VLAN on each switch.

To accomplish this, a portion of the priority field is used to carry the VID.

- Old bridge priority: Priority value (default 32,768 using increments of 1) + MAC address
- New bridge priority: Priority value (default 32,768 using increments of 4,096) + Extended system ID + MAC Address



Remember that if the priority value is not manually configured, the root bridge for each VLAN will be determined by the lowest MAC address., which usually corresponds to the oldest and least powerful switch...

Here's an example:

Default priority field for VLAN 11:	$32768 + 11 = 32779$
Higher priority for VLAN 11:	$28672 + 11 = 28683$

RSTP Compatibility with 802.1D

802.1W and PVRST+ are backwards compatible with common spanning tree, but lose the fast convergence benefit for that particular segment. If a switch receives BPDUs that do not reflect its current operating mode, for two times the hello time, it switches STP modes.

RPVST+ Configuration

Enable RPVST+ globally on the switch:

```
Switch(config)# spanning-tree mode rapid-pvst
```

Multiple Spanning Tree

MST, or 802.1S, expands upon the IEEE 802.1W RST algorithm in an attempt to reduce the number of STP instances, thus reducing the required CPU cycles on a switch. MST enables you to group VLANs and associate them with MST instances. Each instance topology can be independent of the rest, allowing VLANs to be grouped and load balanced for fault tolerance measures. MST is also backwards compatible with all older STP variations.

Switches participating in MST that have the same MST configuration information are referred to as a region. Switches with different MST configurations or that are running legacy 802.1D are considered separate MST regions.

Note: Switches in the same MST region must have the exact same MST configuration to work properly (including revision number).

MST is usually not implemented in campus environments because if you follow the local VLAN model (recommended by Cisco), there should not be that many VLANs on any given switch because they should only extend to the switch block boundary. That makes RPVST+ a better choice because of its simpler configuration. Because MST is still often deployed, Cisco definitely still considers it an important topic on the SWITCH exam.

MST Regions

Each switch that runs MST in one region has a single MST configuration consisting of the following 3 items:

- Region name (alphanumeric)
- Configuration revision number
- A 4096-element table that associates each VLAN to a given instance

Note: The default MST instance for all VLANs is MST00.

BPDUs in MST

From a Peter Paluch's post:
MSTP sends all its BPDUs **untagged**, regardless of the native VLAN on the trunk

MST Configuration

2/26/15

MST must be manually configured on each participating switch, except on VTP v3 where configuring the primary server is enough:

1

Only on VTP V3

Configure VTP v3 Switch as an MST Primary Server:

```
Switch(config)# vtp mode server mst
Switch# vtp primary mst
```

Configure Other VTP v3 Switches as MST Secondary Server or Client:

```
Switch(config)# vtp mode {server | client} mst
Switch(config)# spanning-tree mode mst
// Does not depend on VTP versions
```

2

Enter MST Submode and display current running configuration:

```
Switch(config)# spanning-tree mst configuration
Switch(config-mst)# sh current
```

Define a region name:

```
Switch(config-mst)# name XYZ
```

Set the MST revision number:

```
Switch(config-mst)# revision 1
```

Map the VLANs to an MST instance:

```
Switch(config-mst)# instance 1 vlan 3, 5, 7
Switch(config-mst)# instance 2 vlan 2, 4, 6
```

3

Display configuration to be applied:

```
Switch(config-mst)# show pending
```

Apply the configuration:

```
Switch(config-mst)# exit
```

Or Cancel the configuration:

```
Switch(config-mst)# abort
```

Assign an MST root bridge

```
Switch(config)#spanning-tree mst 2 root primary | secondary
```

Enable MST globally (To be done at the end of the configuration):

```
Switch(config)# spanning-tree mode mst
```

MST Verification Commands

```
Switch# show spanning-tree mst [configuration]
Switch# show spanning-tree mst [[instance-number] [detail]]
// instance-number: Shows MST info for a single instance
```

Cisco
642
813

Chapter 5

Security

Network perimeter security has long been the focus for security products and defenses such as firewalls for layer 3 attacks. The SWITCH exam covers several different security topics in depth, but all from a layer 2 perspective. These kinds of attacks are usually launched from within a network either from legitimate or rogue devices (ex. consumer wireless access points, access switches, and hubs).

A rogue switch added to the access layer could disrupt the Spanning Tree root bridge topology and even worse, could create a loop and bring an entire segment down.

MAC Address Attacks

MAC Flooding → Turns the switch into a hub for a particular VLAN → Mitigated by Port Security / DHCP Snooping + IP Source Guard

The primary MAC address attack attempts to overwhelm the CAM table.

In a MAC flooding attack, an attacker floods a target switch with invalid **source** MAC addresses which quickly fill the CAM table. Once the table is full, all frames whose MAC is not in the table are flooded out all ports causing everyone (including the attacker) to begin to see traffic on their port they would normally not. After the attack stops, the CAM table entries eventually age out so things will return to normal, but in the meantime the attacker may have collected valuable information.

Two preventative techniques for MAC flooding attacks are port security or implementing DHCP Snooping with IP Source Guard – with port security being the most common solution.

MAC Spoofing → Mitigated by PVLANS / DHCP Snooping + IP Source Guard / Dynamic ARP Inspection (DAI)

Another layer 2 MAC attack is MAC spoofing which allows an attacking device to receive frames intended for a different network host: gateway, DNS Server, DHCP Server, ...

One preventative technique for MAC spoofing attacks is implementing DHCP Snooping with IP Source Guard or DAI.

MAC Address Attacks Mitigation

Port Security

Port security can put limits on both what MAC addresses are allowed to be connected to a switch port and how many at any given time. Specific MAC addresses can be statically allowed, or dynamically “learned” using the **sticky** command.

If you simply enable port security on an interface, it will only allow a single MAC address to connect by default. You should specify the maximum number of MAC addresses that should connect to the port using the “**switchport port-security maximum**” command. If you then choose to statically assign MAC addresses to that interface, only those will be allowed plus however remaining until you reach the maximum allowed MAC addresses.

For example, let’s say you configure port security on an interface, configure it for a max of 2 MAC addresses, then statically configure a single MAC address with the “**switchport port-security mac-address**” command. If you try to add another device to the port, it will be accepted because you allowed two MACs with the “**switchport port-security maximum number**” command

Notes:

- Port security can be applied on access ports, voice ports, VLANs, PVLAN ports, static trunk ports and **EtherChannels** but **NOT on routed ports nor dynamic ports (Catalyst 4500, 12.2(31)SG)**.
- Port security commands are applied on static trunks with “**vlan vlan-range**” keyword or on VLANs with “**vlan vlan-range**” command. It restricts the allowed MAC addresses or the maximum number of MAC addresses to individual or range of VLANs on a trunk port.
- Port security does **NOT** support static secure or sticky secure MAC addresses in voice VLANs.
- Port security does or does **NOT** support aging of sticky secure MAC addresses, depending on the platform.

Configuring Port Security

To enable port security on the access or trunk interface or on VLAN

```
Switch(config-if/vlan)# switchport port-security
```

Specify the maximum number of MACs allowed (default is one)

```
Switch(config-if/vlan)# switchport port-security maximum number [vlan vlan-id]
```

Specify the violation action when requirements defined are not met or exceeded:

```
Switch(config-if/vlan)# switchport port-security violation {shutdown [vlan] | restrict | protect}
// Shutdown (default) puts the interface or VLAN in err-disabled state, sends an SNMP trap, logs it and counters++,
// Restrict will drop violator’s frames, send an SNMP trap, log it, and counters++
// Protect will drop violator’s frames quietly.
```

```
Switch(config-if/vlan)# switchport port-security mac-address MAC-address [vlan vlan-id]
```

Dynamically assign MAC addresses up to the allowed maximum (optional)

```
Switch(config-if/vlan)# switchport port-security mac-address sticky [MAC-address] [vlan vlan-id]
```

You can configure an interface to convert the dynamic MAC address to sticky secure MAC addresses and add them to the running configuration by enabling sticky port security. The sticky secure MAC addresses do not automatically become part of the startup configuration. If, however, you save the running configuration to the startup configuration, then reboot the switch, the MACs will be saved upon reboot.

Set the aging time for each assigned MAC (except for sticky MAC)

```
Switch(config-if/vlan)# switchport port-security aging {static | {time minutes } | type {absolute | inactivity}}  
// static: Enable aging for statically configured secure addresses on this port.  
// minutes: 0-1440 minutes; if the time is 0, aging is disabled for this port.  
// absolute: All the secure addresses on this port age out exactly after the time specified and are removed from  
// the secure address list  
// inactivity: The secure addresses on this port age out only if there is no data traffic from the secure source  
// address for the specified time period.
```

Note: Use this feature to:

- (1) remove and add PCs on a secure port without manually deleting the existing secure MAC addresses*
- (2) while still limiting the number of secure addresses on a port.*

To verify the port security settings:

```
Switch# show port-security [vlan | {interface interface-id} | address]
```

The port security **states** can be one of the following:

- **Secure-up:** the port protocol status is UP and the port is secured by port security.
- **Secure-down:** either the port protocol status is DOWN or the port is not secured by port security
- **Secure-shutdown:** the port is secured by port security with shutdown mode and a security violation has occurred.

Port security and VoIP

Port security can be applied to interfaces that use voice VLANs as well. Because voice VLANs typically also include data traffic from a connected PC and an internal switch in the phone, Cisco recommends setting the maximum number of allowed MAC addresses to 3 prior to IOS 12.2(31)SG and to 2 after that release when using port security in conjunction with voice VLANs.

VLAN Attacks → Mitigated by PACLs / VACLs

The major security concern related to VLANs is a concept commonly known as “**VLAN hopping**”. VLAN hopping attacks involve an attacker sending and/or receiving traffic from a VLAN to which they are not assigned. There are two ways this can be done, switch spoofing and double-tagging – both done by manipulating the way switches create and pass data through trunk links.

Switch Spoofing

Switch spoofing uses a computer to mimic a trunk tunnel with a directly connected switch using Dynamic Trunking Protocol (DTP). DTP is enabled by default on Cisco switches and trunk ports also pass all traffic across trunks by default. If an attacker is able to trick the switch into establishing a trunk port, they are able to access (and inject) all traffic going through the switch.

802.1Q Double-Tagging

A VLAN Hopping attack through 802.1Q double-tagging is possible because 802.1Q does not tag frames sent using the native VLAN. In this attack, the attacker sends a payload with two VLAN tags, the first assigned to the segment’s native VLAN and the second assigned to the target destination’s VLAN. The first switch to receive the attacker’s packet strips off the native VLAN tag and forwards it out all ports supporting the native VLAN (including adjacent trunk ports) because that is how 802.1Q handles native VLAN traffic in some switches (**depending on the brand and/or software release**). Once the next hop switch receives the packet, it sees only the second tag and forwards it on to the target destination.

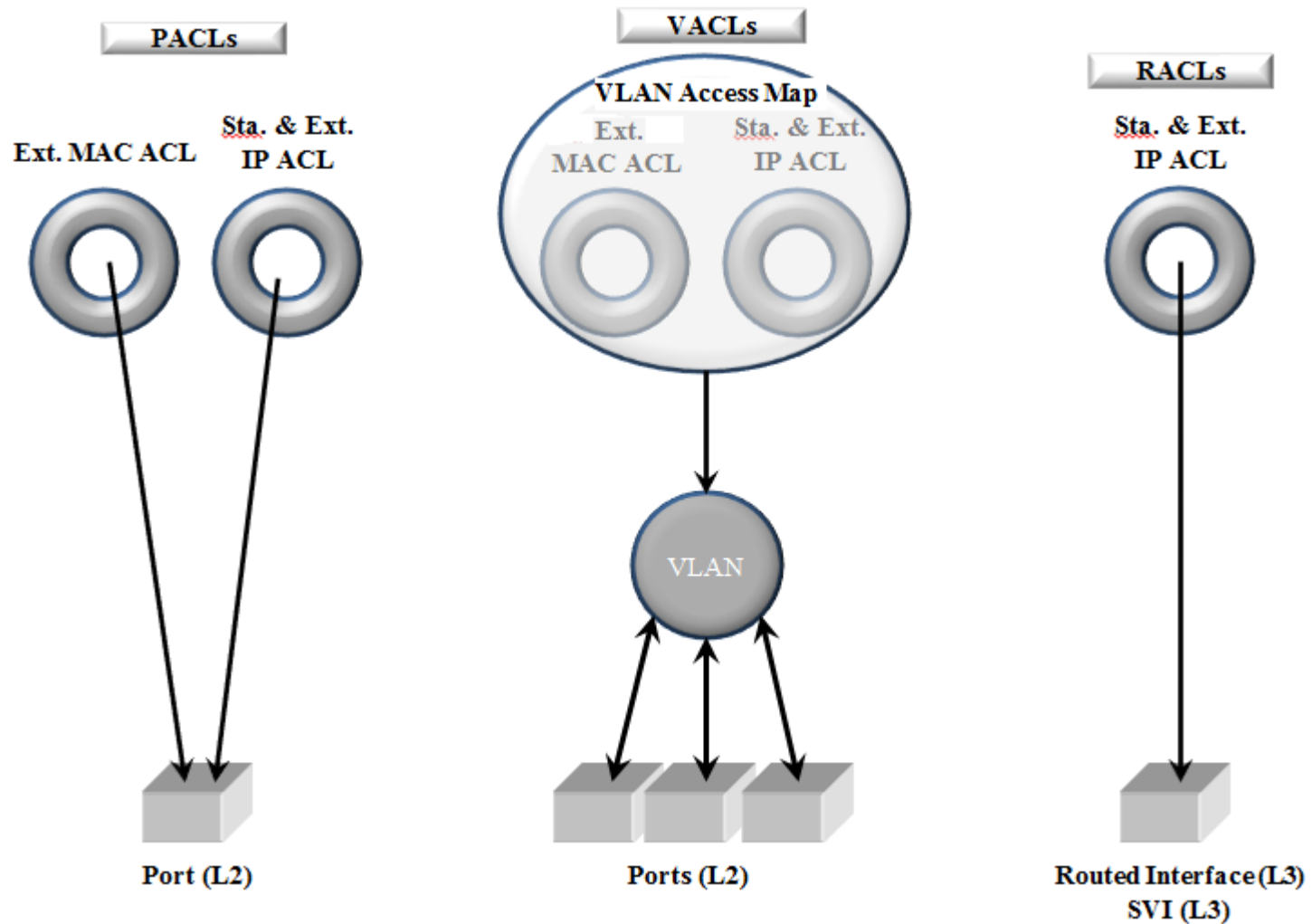
VLAN Attacks Mitigation

- Disable DTP on all ports using the **switchport nonegotiate** command on each port.
- Define access ports and trunk ports explicitly using commands like “**switchport mode access**” and “**switchport mode trunk**” .
- Shutdown all unused ports and assign them all to an unused VLAN (ex. something like 999) != VLAN 1
- Define the native VLAN separately from any data VLANs to use it as an administrative VLAN
- Define explicit VLANs allowed on the trunk links, rather than the default - allow all
- Use PACLs and/or VACLs

PACLs, VACLs and RACLs

Among the four types of access control lists (ACLs) that Cisco switches support, three of them can help mitigate VLAN attacks:

- Traditional Router ACL (RACL)
- QoS ACL
- VACL (also referred to as VLAN access-maps)
- PACL (Port ACL)



PACLs (Port ACLs) → Controlling L2 ↔ L7 Traffic Entering L2 Ports

2/26/15

Port ACLs perform layer 2 or layer 3 addresses access control on all traffic **entering** the specified **Layer 2 port: access, trunk and EtherChannel logical ports. They do not affect Layer 2 control frames received on the port:** (R)STP BPDUs., DTP, VTP, PAgP, LACP, CDP and UDLD. You can apply only one IP ACL and one MAC ACL to a Layer 2 interface.

The CLI syntax for creating a PACL is identical to the syntax for creating an RACL.

An instance of an ACL that is mapped to a Layer 2 port is called a PACL. An instance of an ACL that is mapped to a Layer 3 interface is called a RACL. The same ACL can be mapped to both a Layer 2 port and a Layer 3 interface.

When you apply a PACL to a trunk port, the ACL filters traffic **on all VLANs** present on the trunk port.

When you apply a PACL to a port with voice VLAN, the ACL filters traffic on both data and voice VLANs.

Configuring MAC and IP ACLs:

```
Switch(config)# {mac | ip} access-list {standard | extended} name
```

Configuring MAC ACL:

```
Switch(config)# mac access-list extended name
```

```
Switch(config-ext-macl)# {permit | deny} {hexa-src-mac-addr hexa-src-wildcard-mask | host hexa-src-mac-addr | any}  
                                {hexa-dst-mac-addr hexa-dst-wildcard-mask | host hexa-dst-mac-addr | any}  
                                [{protocol-keyword | ether-type-number ether-type-wildcard-mask}  
                                [cos cos_value]] [vlan vlan_ID]
```

// *protocol-keyword*: a named protocol; for example: arp, stp, dot1q, trill, ip, ipv6

// *ether-type-number ether-type-wildcard-mask*: the protocol within the Ethernet packet with an hexadecimal 0 <=> FFFF

// *vlan* and *cos* are not allowed when MAC ACL is used for VACL filtering

Applying a MAC or IP ACL on an L2 interface:

```
Switch(config-if)# {mac | ip} access-group {name | nbr} {in | out}
```

// => PACL if layer 2 interface - *nbr* and *out* only for IP ACLs

// => RACL if layer 3 interface

Changing the way PACLs interact with other ACLs:

```
Switch(config-if)# access-group mode {prefer port | merge}
```

// **Prefer port mode**: If a PACL is configured on a Layer 2 interface, the PACL takes effect and overwrites the

// effect of other ACLs (RACL and VACL). If no PACL feature is configured on the Layer 2 interface, other features

// applicable to the interface are merged and are applied on the interface.

// This mode is necessary to apply a PACL on a **trunk port**

// **Merge mode**: In this mode, the PACL, VACL, and RACLs are merged in the ingress direction. This is the default

// access group mode. A PACL can be configured on a trunk port only after prefer port mode has been selected.

// Trunk ports do not support merge mode.

VACLs (VLAN ACLs) → Controlling L2 ↔ L7 Traffic Within One VLAN

2/26/15

VACLs control layer 2 or layer 3 addresses access to the VLAN of all packets (bridged and routed), **inbound AND outbound**. Packets can either enter the VLAN through a Layer 2 port or through a Layer 3 port after being routed. You can also use VACLs to filter traffic between devices in the same VLAN. VACLs use standard and extended IP and extended MAC named ACLs and VLAN access maps applied to VLANs for all VACL actions or WAN interfaces only for VACL capture.

VACLs are much like route-maps in that they use match and set statements to define what actions are taken. In this case, the set statements are “action” directives, which include forward, drop, and redirect (to up to 5 interfaces, including port-channels). Also like route-maps, VACL statements are ordered.

Note: We must be very cautious when applying VACLs because they affect absolutely all traffic, including Layer 2 control frames (RSTP BPDUs, ...), IP routing protocols control packets (OSPF, EIGRP, ...), FHRP control packets (HSRP, VRRP, GLBP) and so on...

In particular, in order to avoid STP loops, we must make sure that these BPDUs are allowed through the VLAN(s) on which the VACL is applied. The same remark applies to Layer 3-7 traffic. There are 2 ways to implement them: either by allowing all traffic at the end of the VACL (easiest method), or by forwarding each traffic type separately and dropping all other traffic at the end of the VACL (complicated and hazardous method).

Configuring VLAN Access Map:

```
Switch(config)# vlan access-map name [seq]
Switch((config-access-map)# match ip address {std-or-ext-acl-nbr | std-or-ext-acl-name} {std-or-ext-acl...}
Switch((config-access-map)# match mac address ext-acl-name ... ext-acl-name...
Switch((config-access-map)# action {drop} | {forward [vlan id | capture]} | {redirect int ... int...}
// The "drop" and "forward" actions are self-explanatory
// The "forward vlan" action implements PBF - Policy Based Forwarding, bridging(!) between VLANs
// The "forward capture" action sets the capture bit for the forwarded packets so that ports with the capture
// function enabled can receive the packets.
// The "redirect int" must be in the VLAN for which the VACL access-map is configured.
```

Allowing All Traffic at the End of VACL, Including Layer 2 Control BPDUs and Layer 3-7 Traffic

```
Switch(config)# vlan access-map name [last-seq]
Switch((config-access-map)# action forward
```

OR Allowing Layer 2 Control BPDUs and Layer 3-7 Traffic Separately: A Complete List of Ether Types is available [here](#)

```
Switch(config)# mac access-list extended PVST+
Switch(config-ext-macl)# permit any any Isap 0xAAAA 0x0
Switch(config)# mac access-list extended IEEE-STP-and-ISL-PVST
Switch(config-ext-macl)# permit any any Isap 0x4242 0x0
Switch(config)# mac access-list extended 802.1Q
Switch(config-ext-macl)# permit any any 0x8100 0x0
Switch(config)# mac access-list extended IP
Switch(config-ext-macl)# permit any any 0x800 0x0
Switch(config)# mac access-list extended IP-ARP
Switch(config-ext-macl)# permit any any 0x806 0x0
Switch(config)# mac access-list extended IPv6
Switch(config-ext-macl)# permit any any 0x86DD 0x0
```

Applying the VLAN Access Map on one or several VLANs:

```
Switch(config)# vlan filter access-map-name {vlan-list vlan-id,vlan-id - vlan-id | interface int...int...}
// Applies the specified VLAN access map on the list of VLANs or on the WAN interface(s), inbound AND outbound
// You can apply only one VLAN access map to each VLAN or WAN interface.
// VACLs applied to VLANs are active only for VLANs with a Layer 3 VLAN interface configured
// You cannot apply a VACL to a secondary private VLAN. VACLs applied to primary private VLANs also apply to their
// secondary private VLANs.
```

Displaying the VLAN Access Map configuration or filters:

```
Switch(config)# show vlan {access-map | filter}
```

Example:

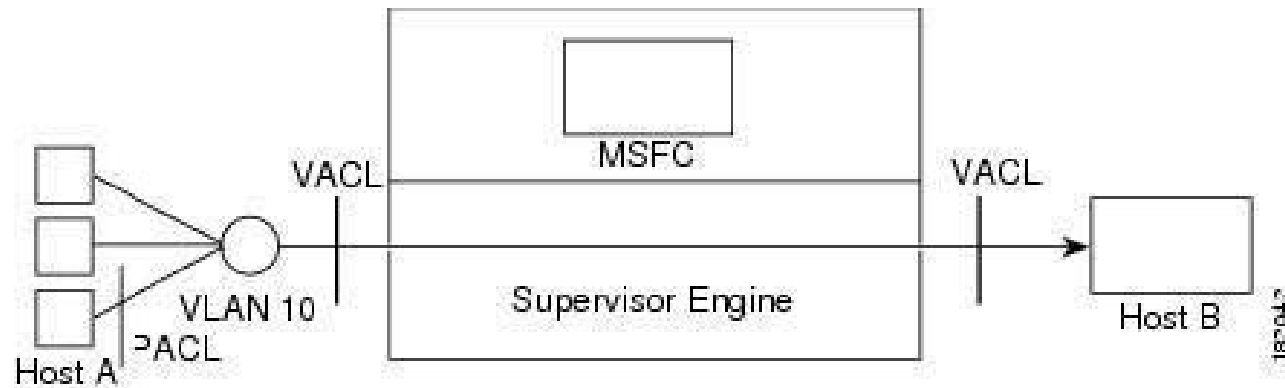
```
Switch(config)# access-list 100 permit ip 10.1.1.0 0.0.0.255 any
Switch(config)# mac access-list extended SERVER
Switch(config-ext-mac)# permit any host 0000.1111.2222
Switch(config)# vlan access-map TEST 10
Switch(config-map)# match ip address 100
Switch(config-map)# action drop
Switch(config-map)# vlan access-map TEST 20
Switch(config-map)# match mac address SERVER
Switch(config-map)# action drop
Switch(config-map)# vlan access-map TEST 30
Switch(config-map)# action forward
Switch(config)# vlan filter TEST vlan-list 14,17
```

Note that traffic not explicitly defined within the VLAN access map will be dropped because of the implicit deny feature at the end of the list. Also, it is important to remember that VLAN maps and router ACLs can be used in combination.

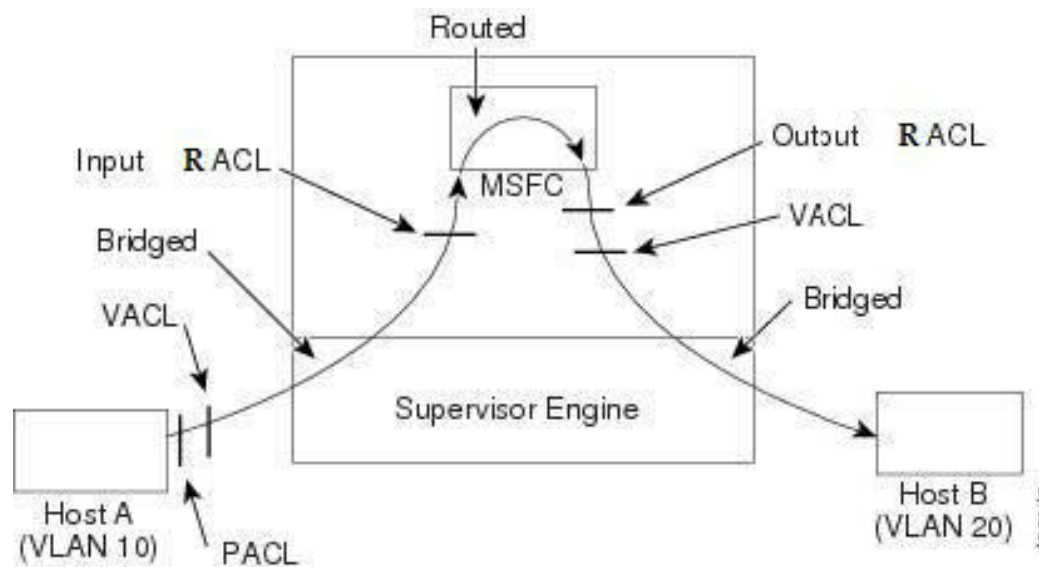
RACLs (Routed ACLs) → Controlling L3 ↔ L7 Traffic Between Different Subnets

IP ACLs are applied to ingress and/or egress Layer 3 interfaces. See CCNP ROUTE Summary for more details (link on first page).

Applying ACLs on Bridged Packets in merge mode



Applying ACLs on Routed Packets in merge mode



Spoofing Attacks

DHCP Spoofing → Mitigated by DHCP Snooping / DHCPv6 Guard

DHCP spoofing attacks occur when an attacker responds to DHCP requests, listing themselves as the default gateway or DNS server. This positions them to intercept all traffic before forwarding it on to the real network gateway.

In the **DHCP Starvation attack**, the attacker floods the DHCP server with requests, filling up the available IP space (DoS attack).

In the **DHCP Starvation variation**, the attacker keeps the same source MAC address but uses different MAC addresses inside the DHCP request ("Client Hardware Address") and provoques same DHCP starvation if the server's implementation reads the MAC address from that field rather than from the Source MAC address field of the packets.

IP Spoofing → Mitigated by DHCP Snooping + IP Source Guard IPv6 Snooping + IPv6 Source Guard + IPv6 Prefix Guard

IP spoofing refers to the creation of IP packets with a forged source IP address, with the purpose of concealing the identity of the sender or impersonating another computing system, such as the gateway or a DNS server.

IP spoofing is most frequently used in denial-of-service attacks. In such attacks, the goal is to flood the victim with overwhelming amounts of traffic, and the attacker does not care about receiving responses to the attack packets. Each spoofed packet appears to come from a different address, and it hides the true source of the attack.

ARP Spoofing → Mitigated by DHCP Snooping + Dynamic ARP Inspection IPv6 Snooping + IPv6 Destination Guard

ARP spoofing is another man-in-the-middle attack exploiting the ARP protocol. An attacker sends out an unsolicited ARP message - called "gratuitous ARP" - giving the IP address of the local gateway or DNS Server with its own MAC address. Local machines then overwrite their ARP tables and all traffic is forwarded through the attacker.

Spoofing Attacks Mitigation

2/26/15

DHCP Snooping → inspects DHCP traffic

DHCPv6 Guard → inspects DHCPv6 traffic

One option for preventing DHCP spoofing attacks is to statically assign ARP entries into the DHCP server so that dynamically created ARP packets cannot interfere. A more manageable solution is to use DHCP snooping. DHCP snooping protects against DHCP spoofing attacks and is a security feature that when enabled, only ports that uplink to an authorized DHCP server are trusted and allowed to pass **all DHCP traffic**. All other ports are untrusted (default) and can only send **DHCP requests**. If a DHCP response (“offer”) is heard on an untrusted interface, it is **shutdown**.

The DHCP snooping database contains the following connections: **host MAC address ↔ Leased IP address ↔ VLAN id ↔ switch port id (option 82)**

DHCP Snooping Configuration

DHCP snooping must be first configured globally, then on specific VLANs, and finally on interfaces. Remember to configure only ports that connect directly to or uplink to a DHCP server as trusted.

Globally:

```
Switch(config)# ip dhcp snooping
```

Optionally DHCP Option 82:

```
Switch(config)# ip dhcp snooping information option
```

```
// Switch inserts its MAC address and the port id where the DHCP DISCOVER has been received
```

```
// DHCP relay inserts its address as 'giaddr' inside the DHCP relayed packets to help the server(s) identify the
```

```
// correct DHCP pool
```

Then on VLAN(s):

```
Switch(config)# ip dhcp snooping vlan vlan-range
```

Then on port(s):

```
Switch(config-if)# ip dhcp snooping trust
```

```
// This Layer 2 port is connected to a DHCP server, or to a switch or router
```

```
Switch(config-if)# ip dhcp snooping limit rate pkts-per-sec
```

```
// Limits DoS attacks (1 - 2048)
```

Verification:

```
Switch# show ip dhcp snooping
```

```
Switch# show ip dhcp snooping [binding | database | statistics]
```

IP Source Guard → inspects all IPv4 traffic

IPv6 Source/Prefix Guard → inspects all IPv6 traffic

IP Source Guard can be applied to access ports. IP Source Guard keeps track of the host IP address and/or MAC address associated with each port – usually in conjunction with DHCP snooping enabled. If traffic sourced from another address enters that interface, it is dropped.

Port security alone with one/specific/a few MAC addresses defined on each interface can be used if IP Source Guard is not supported.

To Configure IP Source Guard on Port(s):

```
Switch(config-if)# ip verify source
// Uses just IP address filtering
Switch(config-if)# ip verify source port-security
// Uses IP + MAC filtering
// Port security must be enabled on the interface
```

To Configure IP Source Binding

```
Switch(config-if)# ip source binding mac-address vlan vlan-id ip-address interface interface-id
// Static association MAC address ↔ VLAN id ↔ IP address ↔ interface id
```

Verification:

```
Switch# show ip verify source
Switch# show ip source binding
```

Dynamic ARP Inspection → Inspects ARP traffic

IPv6 Snooping / IPv6 Destination Guard → Inspects ND traffic

Dynamic ARP Inspection (DAI) is a security mechanism that works with DHCP snooping database to define trusted and untrusted interfaces. DAI intercepts, logs, and drops ARP messages on untrusted ports that do not match the DHCP snooping MAC <-> IP bindings. All traffic that matches is passed; all traffic that does not match is dropped.

DAI is supported on **access ports, trunk ports, EtherChannels, and private VLAN interfaces**. Dynamic ARP Inspection should be only applied to **ingress** interfaces. All access ports should be untrusted and all trunks (including connections to routers) should be configured as trusted. Enable DAI on one or more VLANs, and then configure the trusted interfaces. It matches IP and MAC by default.

DAI allows the configuration of static ARP ACLs to support systems that use statically configured IP addresses and that do not rely on DHCP. DAI can also be configured to drop ARP packets with invalid IP addresses, such as 0.0.0.0 or 255.255.255.255, and ARP packets containing MAC addresses in their payloads that do not match the addresses specified in the Ethernet headers.

On VLAN(s):

```
Switch(config)# ip arp inspection vlan vlan-range
```

Then on port(s):

```
Switch(config-if)# ip arp inspection trust
// Trusted interfaces: no DAI
// Untrusted interfaces : DAI applied
Switch(config-if)# ip arp inspection limit rate frames-per-second
// Limits DoS attacks (1 - 2048)
```

Verification:

```
Switch# show ip arp inspection {vlan vlan-range | interfaces | statistics}
```

RA Guard → Inspects RA Traffic

Secure ND (SEND) → Secures ND Traffic *but not implemented by Cisco anymore*

All IPv6 First-Hop Security techniques implemented by Cisco are detailed in the **IPv6 First-Hop Security Configuration Guide** and **IPv6 Secure ND**. Another way to design IPv6 layer 2 security is to use **TOR L3 switches with one VLAN per port and uRPF**:
[Why is IPv6 layer-2 security so complex \(and how to fix it\)](#)

Port-Based Authentication

2/26/15

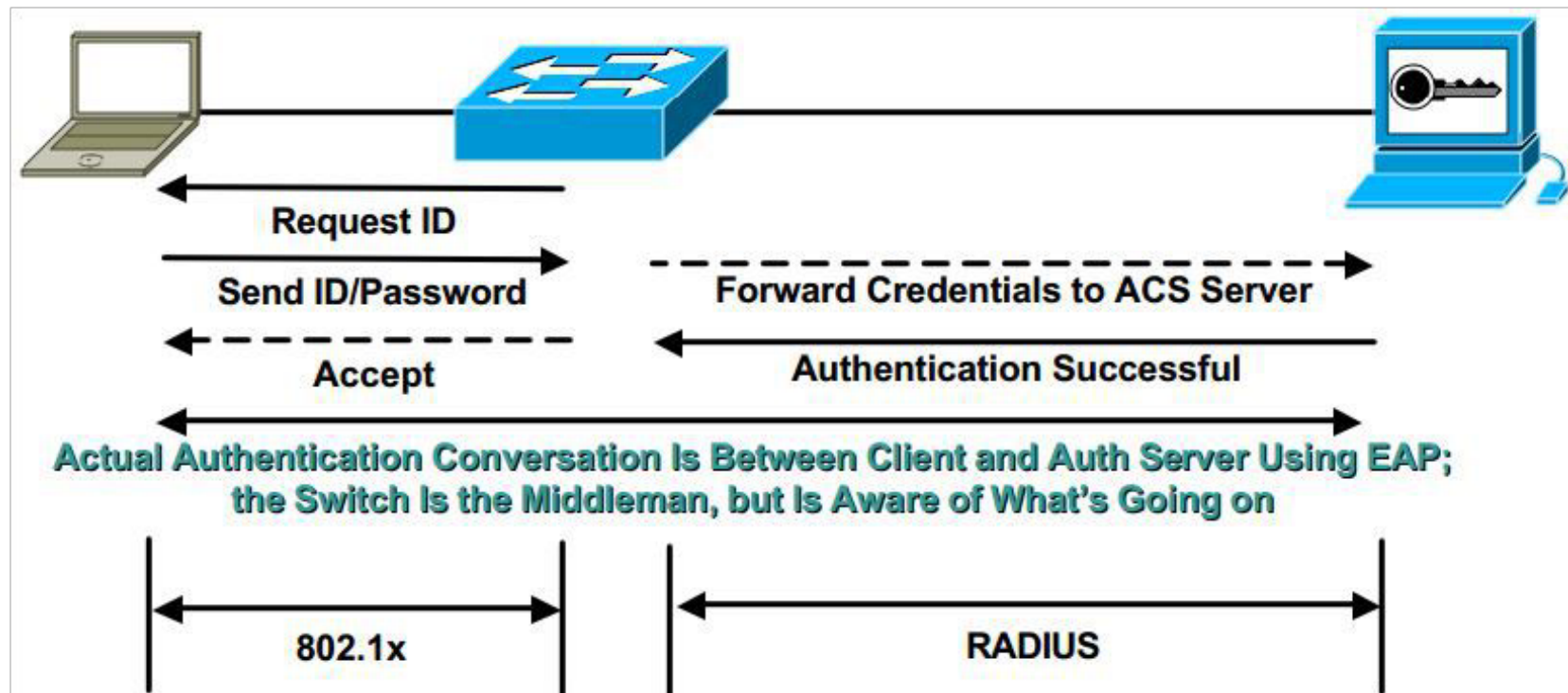
802.1X is a security protocol designed to authenticate devices like computers to access ports on a switch.

When a device connects to an 802.1X enabled access port, it goes through the following steps:

1. It begins in the unauthorized state – only allowing EAPoL (Extensible Authentication Protocol over LAN), CDP, and STP communication.
2. The switch requests authentication or the client sends an EAPoL frame to begin authentication.
3. The switch forwards the client's authentication information to a RADIUS server and acts as a proxy.
4. If authentication is successful, the port transitions to authorized state and traffic is permitted.

802.1X requires three different devices to be configured for port-based authentication to work properly:

- Client (or host) – must be running 802.1X compliant system software
- Switch (or authenticator) – controls physical access and acts as a proxy
- Authentication server – Performs the actual authentication of the clients; Cisco ACS (Access Control System): RADIUS server



Although the Official Certification Guide states that the **RADIUS** security system with **Extensible Authentication Protocol (EAP) extensions is the only supported authentication server (the exam follows the OCG...)**, configuration with a **TACACS+ server has been included for reference.**

To enable port-based authentication using 802.1X:

```
Switch(config)# aaa new-model
// Enables AAA globally, with default lists applied to the VTYs
Switch(config)# aaa authentication dot1x default {group {radius | tacacs+} | local}
// Local: uses database local to the switch
Switch(config)# dot1x system-auth-control
// Globally enables 802.1X on the switch
Switch(config)# {radius-server | tacacs-server} host server-ip-address key string
```

To enable 802.1X on an individual interface

```
Switch(config-if)# dot1x port-control [auto | force-authorized | force-unauthorized]
// auto: Enables 802.1X port-based authentication and causes the port to begin in the unauthorized state, allowing
// only EAPoL, CDP and STP communication
// force-authorized: Disables 802.1X on the interface and causes the port to change to the authorized state without
// any authentication exchange required. This can be applied to interfaces that you inherently trust are secure or
// more likely do not support any type of 802.1X exchange.
// force-unauthorized: Denies all access through this interface by forcing the port to change to the unauthorized
// state, ignoring all attempts by the client to authenticate.
```

To verify:

```
Switch# show aaa [servers | sessions]
Switch# show dot1x [all]
```

Limitations

The 802.1X protocol is supported on **Layer 2 static access ports, voice VLAN ports, and Layer 3 routed ports**, but it is **not supported on these port types**:

- Dynamic access (VQP) ports
- Dynamic (auto - desirable) ports
- Trunk ports
- EtherChannel ports
- Switched Port Analyzer (SPAN) and Remote SPAN (RSPAN) destination ports.

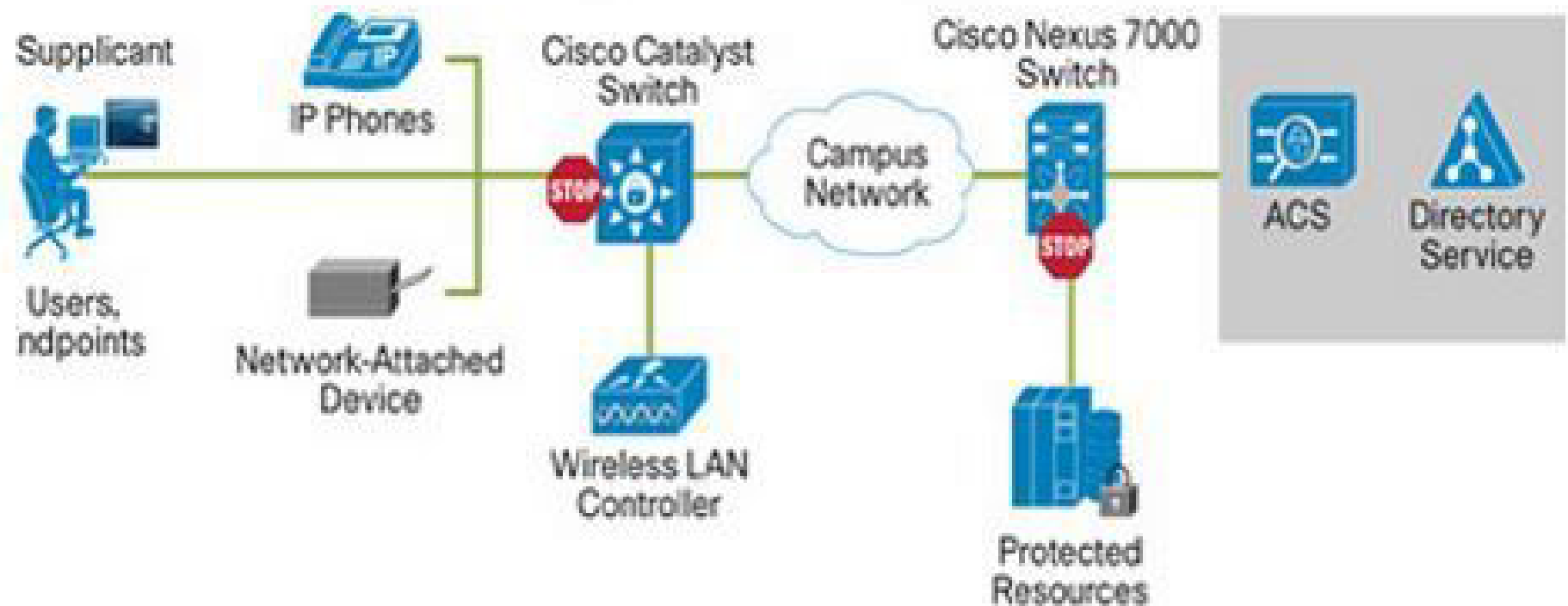
In case the workstations are NOT 802.1X compliant, a LDAP (Lightweight Directory Access Protocol) server could be used instead of a RADIUS server to authenticate them.

ACS

Cisco ACS (Access Control System) offers central management of access policies for user and device administration and for wireless and wired 802.1X network access scenarios. It supports:

- **RADIUS** (Remote Access Dial In User Service): provides centralized Authentication, Authorization, and Accounting (AAA) management of **users**. It uses UDP ports 1645-1646 or 1812-1813.
- **TACACS+** (Terminal Access Controller Access-Control System +): provides centralized Authentication, Authorization, and Accounting (AAA) management of **network devices**. It uses TCP port 49.

Typical ACS 802.1X Deployment



General Security Recommendations

- Use strong passwords that are not susceptible to a dictionary attack (preferably using numbers and symbols and capital letters)
- Use SSH instead of Telnet
- Physically secure the switch
- Use banners that warn intruders against unauthorized access
- Remove unused services (ex. finger, TCP and UDP small servers, service config, and HTTP server)
- Configure a Syslog server
- Disable DTP (Dynamic Trunking Protocol) – define trunks explicitly
- Disable CDP when it is not required: no cdp run (disables it globally on the switch). CDP is for example **required for Auto QoS and Cisco IP phones.**

Security Features Summary

Attacks and Mitigations

Attacks		Mitigation
MAC	MAC Flooding	DHCP Snooping + IP Source Guard / Port Security
	MAC Spoofing	DHCP Snooping + IP Source Guard / DAI / PVLANs
VLAN	Switch Spoofing	Disable DTP + Shutdown unused ports / VACLs
	802.1Q Double-Tagging	Upgrade Switch Software to drop frames with native VLAN Tag / VACLs
IP	ARP Spoofing	DHCP Snooping + DAI
	IP Spoofing	DHCP Snooping + IP Source Guard
DHCP	DHCP Spoofing	DHCP Snooping

Security Features Configuration and Verification

Port Security	1)N/A 2)#switchport port-security #switchport port-security maximum [vlan] #switchport port-security mac-address [sticky] [vlan] #switchport port-security aging {static (time minutes) type {absolute inactivity}} #switchport port-security violation {shutdown restrict protect}	1)2)#show port-security [interface interface-id] [address vlan]
DHCP Snooping	1)#ip dhcp snooping #ip dhcp snooping information option #ip dhcp snooping vlan vlan-range 2)#ip dhcp snooping trust #ip dhcp snooping limit rate packets-per-sec	1) 2) #show ip dhcp snooping [binding database statistics]
IP Source Guard	1)#ip verify source [port-security] #ip source binding mac-address vlan vlan-id ip-address interface interface-id 2)N/A	1) #show ip verify source #show ip source binding 2))N/A
DAI	1)#ip arp inspection vlan vlan-range 2)#ip arp inspection trust #ip arp inspection limit rate frames-per-second	1) 2) #show ip arp inspection [interfaces [int-id vlan vlan-range statistics]
802.1X Authentication	1)#aaa new-model #aaa authentication dot1x default {local group (radius tacacs+)} #{radius-server tacacs-server } host ip-address key string #dot1x system-auth-control 2)#dot1x port-control {auto force-authorized force-unauthorized}	1)#show aaa {servers sessions} 2)#show dot1x [all]

*: 1) Globally => (config)
2) Interface => (config-if)

PACLs	1) #ip access-list {standard extended} <i>name</i> #mac access-list extended <i>name</i> #{permit deny} {<i>hexa-src-mac-addr hexa-src-wildcard-mask</i> host <i>hexa-src-mac-addr</i> any} {<i>hexa-dst-mac-addr hexa-dst-wildcard-mask</i> host <i>hexa-dst-mac-addr</i> any} [<i>protocol-keyword</i> {<i>ethertype wildcard-mask</i>}] 2) #{mac ip} access-group {<i>name</i> <i>nbr</i>} {in out} #access-group mode {prefer port merge}	
VACLs	1) #vlan access-map <i>name</i> [seq] <i>seq</i> #match ip address {<i>std-or-ext-acl-nbr</i> <i>std-or-ext-acl-name</i>} #match mac address <i>ext-acl-name</i> #action {drop} {forward [vlan <i>id</i> capture]} {redirect <i>int</i>} #vlan filter <i>access-map-name</i> {vlan-list <i>vlan-range</i> interface <i>int</i>} 2) N/A	1) show vlan {access-map filter} 2) N/A
RACLs	cf. CCNP ROUTE Summary	

*: 1) Globally => (config)
 2) Interface => (config-if)

Cisco
642
813

Chapter 6

Inter- VLAN Routing

Inter-VLAN Routing

VLANs require a layer 3 device between them to communicate. Cisco recommends using layer 3 routing at the distribution layer or core layer of the multilayer switched network to terminate local VLANs, isolate network problems, and avoid access layer issues from affecting the core.

There are 3 inter-VLAN routing device options:

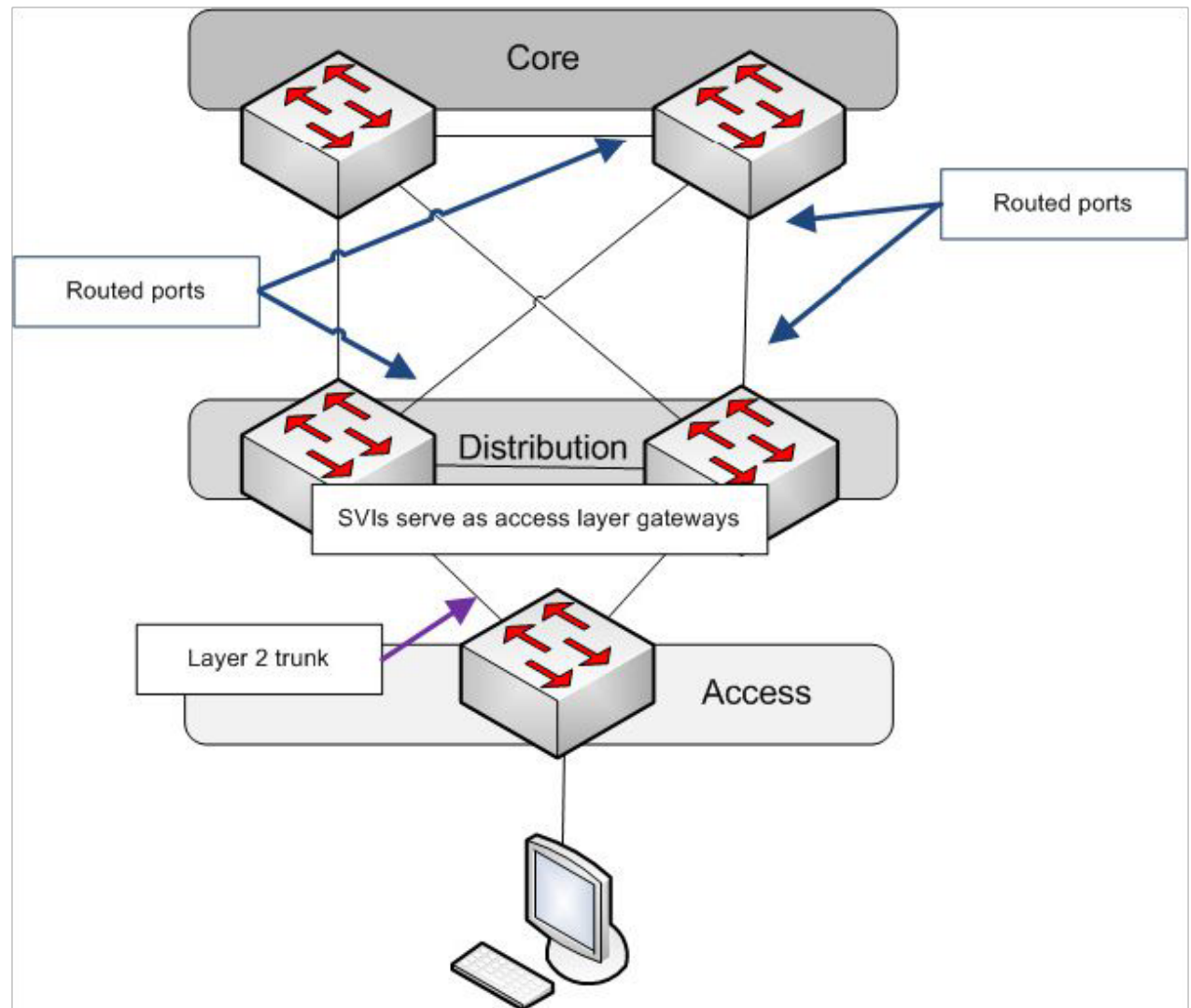
- layer 3 multilayer Catalyst/Nexus switch
- external router that allows trunking (router-on-a-stick)
- external router with enough interfaces for every VLAN (this doesn't scale and is very expensive)

All Catalyst/Nexus multilayer switches support the following types of layer 3 interfaces:

Routed port – a pure layer 3 port similar to that on a router

Switch virtual interface (SVI) – virtual routed VLAN interface in a switch for inter-VLAN routing

Bridge virtual interface (BVI) – a layer 3 interface on a router that groups a set of layer 2 interfaces that gets bridged - equivalent of an SVI in a router to span a vlan there.



External Router (Router-on-a-Stick)

A layer two switch can be connected to a single router to allow inter-VLAN communication either using a single physical link as a trunk with multiple sub-interfaces (a.k.a. router-on-a-stick) or using separate physical links between the switch and router for each individual VLAN.

Router-on-a-stick Example:

```
interface FastEthernet 0/1
no ip address
duplex auto
speed auto
```

```
interface FastEthernet 0/1.10
description data vlan
encapsulation dot1q 10
ip address 10.1.10.254 255.255.255.0
```

```
interface FastEthernet 0/1.20
description mgmt vlan
encapsulation dot1q 20
ip address 10.1.20.254 255.255.255.0
```

```
interface FastEthernet 0/1.55
description native vlan
encapsulation dot1q 55 native
ip address 10.1.55.254 255.255.255.0
```

Advantages

- Works with almost all switches because the switches do not have to support layer 3, just VLANs and trunking
- Simple configuration (one switch port, one router interface)

Disadvantages

- Router is a single point of failure
- If the trunk becomes congested, it can affect every VLAN
- Slightly higher latency because traffic must leave and re-enter the switch and the router makes the traffic decisions in software (which is slower than hardware)
- The added processing on the router will add overhead

Inter-VLAN Routing with an External Router

Implementation Planning

- Need to know how many VLANs require routing, the VLAN IDs, and what ports connect to the router
- Every router sub-interface must be configured with the same type of frame encapsulation (usually 802.1q) as the switch side of the link
- Make sure the native VLAN is the same on both ends. Now a sub-interface on the router can be created for the native VLAN, also if it is a sub-interface – make sure to define its encapsulation type with the "**encapsulation dot1q vlan-id native**" command.
- It is best practice to match the sub-interface ID with the VLAN ID

Configuring Router-on-a-stick

1. Enable trunking on the switch port
2. Enable the router interface with the "**no shut**" command
3. Create the sub-interfaces on the router for each VLAN
4. Configure IPs and encapsulation on each sub-interface as they relate to their VLANs

*Note: There **cannot** be a trunk negotiation (DTP) between the switch and the router, so the trunk configuration on the switch side is only static.*

Example Router Interface Configuration:

```
Router(config)# interface FastEthernet0/0
Router(config-if)# no shutdown
Router(config)# interface FastEthernet 0/0.1
Router(config-subif) description VLAN 1
Router(config-subif)# encapsulation dot1Q 1 native
Router(config-subif)# ip address 10.1.1.1 255.255.255.0
Router(config-subif)# exit
Router(config)# interface FastEthernet 0/0.2
Router(config-subif)# description VLAN 2
Router(config-subif)# encapsulation dot1Q 2
Router(config-subif)# ip address 10.2.2.1 255.255.255.0
Router(config-subif)# exit
Router(config)# end
```

Example switch trunk interface configuration

```
switch(config)# interface FastEthernet 4/2
switch(config-if)# switchport trunk encapsulation dot1q
switch(config-if)# switchport mode trunk
```

Switch Virtual Interfaces

Remember that Cisco recommends using layer 2 between access and distribution layers and layer 3 routing between distribution and core layers. SVIs are virtual VLAN interfaces on multilayer switches; one SVI is created for each VLAN to be routed and it performs the process for all the packets associated with that VLAN. The only SVI created by default is for VLAN 1. The rest must be created manually using the command:

```
Switch(conf)# interface vlan vlan_id
```

SVIs are commonly used for:

- Default gateways for users within the VLAN
- Virtual route between VLANs
- Provides an IP address for connectivity to the switch itself
- Can be used as an interface for routing protocols

An SVI is considered “up” when at least one (access or trunk) interface in its associated VLAN is active and forwarding traffic. If all interfaces within that VLAN are down, the SVI goes down to prevent creating a routing black hole.

Advantages

- Fast because all performed in hardware
- No need for external links for routing
- Low latency (doesn't need to leave the switch)

Disadvantage

- May require a more expensive switch

Inter-VLAN Routing with SVIs

Implementation Planning

- Identify which VLANs require layer 3 gateways as you may not want all VLANs to be routable within the organization
- Make sure VLANs are first created on the switch, then make the SVIs
- Find out what IPs need to be configured on each SVI interface, then use the “**no shutdown**” command to enable them
- Configure any routing protocols that are required
- Determine if any switch ports should be excluded from contributing to the SVI line-state up-and-down calculation
- Verify that the SVIs are UP/UP with “**show int vlan id**” or “**show ip int brief**”

Configuring SVIs

1. Enable IP routing
2. Create the VLANs
3. Create the SVIs
4. Assign an IP address to each SVI
5. Enable the interfaces
6. Optional – Enable an IP routing protocol
7. Verify the SVIs state

Example Configuration:

```
Switch# configure terminal
Switch(config)# ip routing
Switch(config)# vlan 10
Switch(config)# interface vlan 10
Switch(config-if)# ip address 10.10.1.1 255.0.0.0
Switch(config-if)# no shutdown
Switch(config)# router rip
Switch(config-router)# network 10.0.0.0
```

Note: Routing protocols are only required to allow different devices to communicate across different VLANs or networks. They are not required to route between SVIs on the same switch because the switch sees the SVIs as connected interfaces.

SVI Autostate

An SVI is automatically created when the following conditions are met:

- The VLAN is active and exists in the VLAN database
- The VLAN interface exists and is not administratively shut down
- At least a single access or trunk port on the switch belongs to the VLAN, is in the UP/UP state, and is in the spanning-tree forwarding state.

If there are multiple ports on the switch in the same VLAN, the default action is to take down the SVI interface if all of the ports in that VLAN are shut down.

The command "switchport autostate exclude", when applied to a port, will allow the VLAN to go down if all the other ports in the VLAN go down except the one "autostate exclude" was applied to. This is often desirable when traffic analyzers are attached to a switch. They will stay up, but are just passive monitors, so if all other devices in the VLAN go down – this port would prevent the VLAN from going down, so autostate exclude is applied to allow the VLAN to still go down.

Routed Ports

Routed ports are physical ports on the switch that act much like a router interface would with an IP address configured. Routed ports are not associated with any particular VLAN and do not run layer 2 protocols like STP and VTP.

Note: Routed interfaces also do not support sub-interfaces.

Routed ports are point-to-point links that usually connect core switches to other core switches or distribution layer switches (if the distribution layer is running layer 3). They can also be used when a switch has only a single switch port per VLAN or subnet.

Make sure when configuring a routed port that you use the "**no switchport**" command to make sure the interface is configured to operate at layer 3. Also make sure to assign IP addresses and any other layer 3 information required. Check that routing protocols are configured.

Advantages

- A multilayer switch can have both SVIs and routed ports configured
- Multilayer switches forward all layer 2 and 3 traffic in hardware, so it is very fast

Inter-VLAN Routing with Routed Ports

1. Select the interface
2. Convert to layer 3 port (“**no switchport**” command)
3. Add an IP address
4. Enable the interface (“**no shut**” command)
5. *There must be at least one SVI on the switch to communicate with the routed interface.*

Example:

```
Core(config)# interface GigabitEthernet 1/1
Core(config-if)# no switchport
Core(config-if)# ip address 10.10.1.1 255.255.255.252
Core(config-if)# exit
```

Verifying Inter-VLAN

```
#show ip interface {interface_type_port | vlan svi_number}
#show interface {interface_type_port | vlan svi_number}
#show running interface {type_port | vlan svi_number}
#show vlan
#show interfaces trunk
```

Troubleshooting Inter-VLAN Problems

Here is a list to run through when identifying an issue related to inter-VLAN routing:

- Correct VLANs on switches and trunks
- Correct routes
- Correct primary and secondary root bridges
- Correct IP addresses and masks

Routing Protocol Configuration

Unlike routers, multilayer switches do not automatically route until a layer 3 interface is defined or an SVI is created. Routing can be configured just like on an actual router, using static routes and dynamic routing protocols. If routing is required, make sure the global "**ip routing**" command has first been applied. You may be required to do some dynamic routing protocol configuration on a multilayer switch within the SWITCH exam, so make sure you brush up on your routing protocol basics.

Example:

```
Switch(config)# ip routing
Switch(config)# router eigrp 20
Switch(config-router)# no auto-summary
Switch(config-router)# network 10.0.0.0
Switch(config-router)# exit
```

To verify a routing protocol is behaving as expected, use the "**show ip route**" command to display the active routing table routes.

"**Show ip protocols**" will allow you to see the routing protocols currently running on the device.

Multilayer Switching

A Multilayer switch can perform both layer two switching as well as inter-VLAN routing. Knowing the order of events within the switch will help you understand how the many forwarding and filtering options interact.

Multilayer Switching, or MLS, is a fairly general term used to describe features that enable very efficient routing of traffic between VLANs and routed ports.

Switch Forwarding Architectures

There are four different ways packets are switched on a layer 3 switch or router:

Process Switching

Each packet is examined by the internal processor and is handled in software. This is the slowest option (only used in routers).

Route Caching (old method also known as “fast switching”, “flow-based switching” or “demand-based switching”)

The route processor tracks a flow first packet, setting up a “shortcut” for the remaining packets to avoid software-based routing, instead being immediately forwarded in hardware. This method is faster than process switching and is done in both routers and layer 3 switches.

Cisco Express Forwarding (a.k.a. CEF or “topology-based switching”)

Layer 3 routing table dynamically populates a single database of the entire network topology in hardware (the **FIB with adjacency table**) for fast and efficient lookup. This is the second fastest method and is the default option within Cisco routers and multilayer switches.

Distributed CEF

One **FIB / adjacency table per line card**: it is the fastest routing method.

Cisco Express Forwarding

Cisco Express Forwarding, or CEF, is the specific implementation of MLS Cisco uses on their multilayer switches. It supports per destination load-sharing. All modern Catalyst/Nexus switches use CEF by default, so no manual configuration is necessary.

CEF Exceptions

Some types of traffic are not able to bypass the processor using CEF. Some examples include:

- CDP packets
- ARP packets
- ICMP response (TTL expired, MTU exceeded, etc.)
- IP broadcasts – multicast (DHCP request – Routing Protocol Updates)
- Packets that must be fragmented
- Packets with IP header options
- Anything that must be encrypted or tunneled
- Packets triggering NAT
- Most non-IP packets

Distributed CEF

Distributed CEF, commonly denoted dCEF, speeds up CEF switching even more by running a FIB table on each of a switch's line cards. Because the FIB lookup occurs directly on the line card itself, it no longer has to cross the switch backplane and it allows for multiple L3 data plane operations to be performed simultaneously within a single chassis-based system. The main route processor of the switch is responsible for generating a central master FIB and adjacency table and distributing these tables out to each line card dCEF.

This is currently the fastest method of implementing CEF on Cisco switches.

Switching methods in order from fastest to slowest: dCEF, CEF, fast switching, process switching.

CEF Load-balancing

Cisco Express Forwarding load balancing is based on a combination of source and destination packet information; it allows you to optimize resources by distributing traffic over multiple paths.

Per-Destination Load-Balancing: Per {IP Source ↔ IP Destination} Pair

It is actually **per source-destination based load-balancing**. Packets for a given source-destination host pair are guaranteed to take the same path, even if multiple paths are available. Traffic streams destined for different pairs tend to take different paths.

Per-destination load balancing is **enabled by default** when you enable Cisco Express Forwarding.

You can use per-destination load balancing to ensure that packets for a given host pair **arrive in order**.

Per-Packet Load-Balancing: Round-Robin

It allows the L3 switch to send successive data packets over different paths without regard to individual hosts or user sessions. It uses the **round-robin** method to determine which path each packet takes to the destination. Per-packet load balancing ensures that the traffic is balanced over multiple links. Packets for a given source-destination host pair might take different paths. Per-packet load balancing can therefore introduce **reordering** of packets. This type of load balancing is **inappropriate for certain types of data traffic** (such as VOIP) that depend on packets arriving at the destination in sequence. Use per-packet load balancing to help ensure that a path for a single source-destination host pair does not get overloaded. If the bulk of the data passing through parallel links is for a single pair, per-destination load balancing overloads a single link while other links have very little traffic. Enabling per-packet load-balancing allows you to use alternate paths to the same busy destination. So this method is inappropriate in most situations and is **not recommended**.

Load-Balancing Algorithms

- **Original algorithm:** it is **not recommended** because it introduces some distortions in load sharing across multiple routers!
- **Universal algorithm (default):** it allows each router on the network to make a different load sharing decision for each source-destination address pair.
- **Tunnel algorithm:** it is designed to balance the per-packet load when only a few source and destination pairs are involved.
- **Include-ports algorithm:** it allows you to use the Layer 4 source and destination ports as part of the load-balancing decision.

Configuring CEF

```
Switch(config)# ip cef [distributed]
Switch(config-if)# ip load-sharing {per-destination | per-packet}
Switch(config)# ip cef load-sharing algorithm {original | universal | tunnel | include-ports {source | destination | source destination}}
```

Verifying CEF

```
Switch# show ip cef [vlan vlan-id] [detail | summary]
```

```
Switch# show adjacency [detail]
```

Troubleshooting CEF

```
Switch# show cef [drop | not-cef-swirched]
// legacy commands since IOS 12.2 (25)
Switch# debug ip cef
```

Layer 2 Forwarding Process

(STP information not included here)

INPUT	OUTPUT
1. Receive frame	1. Apply outbound QoS ACL
2. Verify integrity	2. Apply outbound VACL
3. Apply inbound PACL, VACL and QOS ACL <-> TCAMs	3. Select outbound port
4. Update MAC Address Table <-> 1 CAM/VLAN	4. Place in port queue
5. Destination MAC Lookup in CAM	5. Rewrite
	6. Forward frame

Layer 3 Forwarding Process

INPUT	ROUTING	OUTPUT
1. Receive frame	1. Apply input RACL <-> TCAM	1. Apply outbound QoS ACL
2. Verify integrity	2. Route if entry is in CEF cache (FIB)	2. Apply outbound VACL
3. Apply inbound PACL, VACL and QOS ACL <-> TCAMs	3. Identify exit interface and next hop address using FIB	3. Select outbound interface
4. Update MAC Address Table <-> 1 CAM/VLAN	4. Apply outbound RACL	4. Place in interface queue
5. Destination MAC Lookup in CAM		5. Rewrite packet (src. + dest. MAC, decrement TTL in IP header, compute IP checksum and FCS,)
		6. Forward frame

CAM

The CAM table stores information about frames that pass through the switch for more intelligent forwarding.

Unlike RAM in which the process supplies a memory address and the RAM returns the data word stored at that address, a CAM is designed such that the process supplies a data word and the CAM searches its entire memory to see if that data word is stored anywhere in it. If the data word is found, the CAM returns a list of one or more storage addresses where the word was found and other associated pieces of data → must find **an exact match to find a hit**.

The CAM table (1 CAM table per VLAN) stores three pieces of information about traffic:

- MAC address
- Inbound port
- Type (static/dynamic)

Frames passing through the switch first enter the ingress queue, then proceed simultaneously to the Security TCAMs (ACLs), QoS TCAM, and L2 Forwarding Table (CAM). Afterwards, they all then enter the egress queue before exiting an interface.

Displaying the CAM Table

```
Switch# sh mac address-table [static | dynamic] [count] [vlan vlan-id] [interface type/number] [address mac-address]
// dynamic: Allows you to view the contents of the switch CAM table (ones learned through passing frames)
// vlan: Shows the CAM table entries according to VLAN assignments. So if you want to see how many hosts the
// switch knows about in a particular VLAN, this lays it out in a nice table format.
```

*Troubleshooting tip: If you need to find out where a particular device is attached to the network, you can run the "**sh mac address-table dynamic address** xxxx.xxxx.xxxx" command at the core of the network, determining which ports it is connected to (and thus downstream switch). Continue the process until you reach the final access switch that the device is attached to.*

Switch Database Manager

```
Switch # show sdm prefer
// Shows max number of entries in the MAC address table and max number of routes in the RIB. These max can be changed.
```

TCAM

The TCAM (Ternary CAM) stores ACLs, layer 3 and up information.

It allows a third matching state of "X" or "Don't Care" for one or more bits in the stored dataword, thus adding flexibility to the search → an exact match is not required, **the longest match is considered a hit**. The TCAM is organized by masks – each mask has 8 value patterns associated with it.

Note that each {mask-value} pair is evaluated simultaneously (in parallel) looking for the longest match in a single lookup.

The TCAMs are used for PACL, VACL, QOS ACL and RACLs, and the FIB.

FIB + Adjacency Tables

The FIB, or **Forwarding Information Base**, is what allows CEF to switch layer three traffic so quickly. It is created in hardware using the existing routing table to create a single route cache, allowing the packets to be forwarded directly the very first time they are seen on the switch.

The FIB uses destination IP address and mask (certainly class-based mask derived from the destination IP address) as table indexes. Also contains next-hop IP and MAC so no other lookup is necessary.

There are **6 FIB entry types**:

- **Resolved** Any prefix with a resolved adjacency for the next hop
- **Receive** Contains a prefix with a mask of 32 corresponding to one of the IP addresses of the L3 switch and an IP address of the broadcast subnet
- **Connected** Prefix associated with a connected network
- **Drop** All these packets are dropped
- **Default** The default route which matches all subnets; only present if there is a default route present in the routing table
- **Wildcard** This matches all entries; only present if there is no default entry - packet is dropped - or if the FIB table is full - packet is sent to the L3 engine.

In the rare case in which the FIB table is full, the wildcard entry is still present but, instead of dropping packets that match it, they are forwarded to the routing engine. This only occurs if you have more than a 256K prefixes in the FIB and if you cannot store the complete routing table and ARP adjacencies. You then need to have the default mechanism sent to the routing engine since it can have a routing entry that is not present in the FIB.

CEF uses another table, the **adjacency table**, along with the FIB to quickly forward packets. While the FIB stores the routing information, the adjacency table is derived from the ARP table and stores the layer 2 next-hop address and frame header rewrite information for all FIB entries. The control plane is what controls and coordinates all of this information, which is physically separate from the data plane (the actual layer 2 forwarding). This further allows performance improvements.

To recap, the FIB is responsible for maintaining the next-hop IP address for all known routes and the adjacency tables maintain the layer 2 information. The adjacency tables link to the FIB entries, so combined they provide all the layer 2 and 3 next hop information necessary to dramatically increase packet switching speed. When the adjacency table is full, a TCAM entry points to the L3 engine to redirect the adjacency.

There are 7 adjacency categories:

- **Cache** Contains the correct outbound interface and the correct MAC address
- **Receive** Handles packets whose final destinations include the router itself (own interfaces, broadcasts and multicasts)
- **Glean** Unknown MAC address. Packets that trigger this entry will generate an ARP request
- **Punt** Deals with packets that require special handling or can not be switched by CEF. Such packets are forwarded to the layer 3
- **Drop** Packets pointing to this entry are dropped, but the prefix will be checked
- **Null** Handles packets destined to a NULL interface
- **Discard** FIB entries pointing to this type of adjacency will be discarded

Cisco
642
813

Chapter 7

IP SLA

IP Service Level Agreement

Service level agreements or SLAs are contractual agreements usually between a customer and service provider that spell out the minimum acceptable levels of service. SLAs are often attached to WAN and MPLS links because any downtime can significantly affect business performance/profits.

In terms of the exam, Cisco's SLA attempts to measure latency, jitter (variation of delay), and packet loss for a given link. Cisco does this by enabling IOS to send synthetic traffic to a specific host computer or router that is or isn't configured to respond. The router can then use it to determine one way jitter, delay, and packet loss.

Common IP SLA Functions

- Active edge-to-edge network availability monitoring
- Network performance monitoring
- VoIP, video, and VPN monitoring
- IP health assessment
- MPLS monitoring
- Troubleshooting

IP SLA can measure the following statistics:

- Network latency (delay) and response time (RTT)
- Packet loss
- Jitter and voice quality scoring
- End-to-end network connectivity

IP SLA Operations

Multiple IP SLA operations (measurements) can run in a network at the same time. The reporting tools use SNMP to fetch the data so they can report on it.

The source router needs to be configured with a target device, protocol, and sometimes a UDP/TCP port number for each IP SLA operation. The source router uses the IP SLA control protocol to confirm communication with the responding host before the source sends the test messages (jitter operations).

To increase security, the responder can use an MD5 hash to authenticate the message from the source, securing the exchange.

When the operation is complete, the results are stored in the IP SLA MIB on the source and can be retrieved via SNMP (or by traps which can be conditionally set to send alerts if thresholds are exceeded).

Almost the entire configuration occurs on the source router. The source sends the probe packets that test whatever protocols the administrator chooses.

Note: Although any IP device can be a responder, another IP SLA router running IOS is preferred because the measurement accuracy will be improved and it is required if you want to measure jitter.

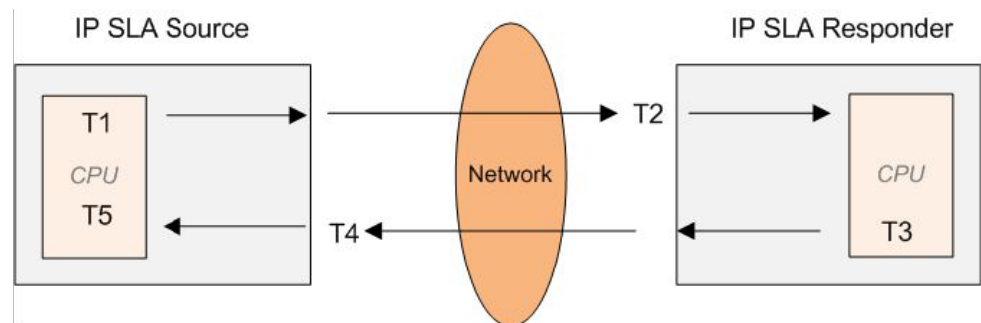
You will find a complete description of IP SLA in the [CCNP ROUTE Summary](#).

IP SLA Operation Breakdown

1. Source sends an IP SLA control message *in jitter operations* to the responder using UDP port 1967. The control message carries the protocol, port, and duration defined when the operation was configured on the source router.
 - o If MD5 is enabled, the checksum is sent with the control message.
 - o If authentication is enabled, the responder verifies it. If authentication fails, the responder returns an authentication failure message.
 - o If a response is not received from the responder, the source will attempt to retransmit until it eventually times out.
2. The responder sends a confirmation message back to the source router and listens on the specified port.
3. If the response from the control message is OK, it begins sending probe packets.
4. The responder responds to the incoming probe packets for the predetermined time.

The diagram to the right outlines the timestamp process IP SLA uses to calculate round trip time (RTT) accurately.

1. The source sends a packet at time T1
2. The responder records both the receipt time (T2) and the transmitted time (T3). Because there can be delay between when the router receives the packet and when a confirmation is sent back out the interface, it tracks the difference in time (sub-milliseconds). The source later subtracts this difference from the total RTT because it was not time in transit, but rather router processing time.



An additional benefit of so many timestamps is the ability to track one-way delay, jitter, and packet loss. Remember that traffic behavior can be asynchronous. Also, make sure that both devices are using the same source for clock information. The *same NTP server is a requirement* for many of these functions.

Configuring IP SLA

1. Configure the source router
2. Activate IP SLA on the source
3. Configure the tracking object on the source
4. Configure the responder

Example Source Configuration:

```
Switch(config)# ip sla 10
// number indicates the IP SLA test identifier
Switch(config-sla)# type echo prot ipIcmpEcho 192.168.1.10 source-int fa0/1
Switch(config-sla)# frequency 20
// number of seconds between each operation
Switch(config)# exit
Switch(config)# ip sla schedule 10 life forever start-time now
Switch(config)# track 1 ip sla 10 reachability
OR
Switch(config)# track 1 rtr 10
// Tracks a Response Time Reporter (old IOS releases)
```

Example Responder Configuration:

```
Switch2(config)# ip sla responder [{tcp-connect | udp-echo} ipaddress source-ip-address port port-number]
```

Verifying IP SLA

```
Switch# show ip sla statistics
Switch# show ip sla configuration [operationID]
Switch# show ip sla application
// Shows which IP SLA operation types are supported in your software image
Switch# show ip sla responder
```

Cisco
642
813

Chapter 8

High Availability

High Availability

High availability is an organizational objective that enables resilience by increasing network availability and includes the following components:

- Redundancy
- Technology
- People (ex. skills, training)
- Processes (ex. change control)
- Tools (ex. network management, documentation)

Review of Failover Times

- RSTP converges in about 1 second
- EtherChannel can failover in approximately 1 second (when a single link in the bundle fails, it redirects traffic to the other links)
- Default HSRP timers are 3 seconds for hellos and 10 seconds for hold time but best practice says to change hellos to 1 sec. so convergence takes less than 3 seconds. Hellos timer can be lowered to 300 ms so that convergence takes less than 1 second.
- EIGRP and OSPF can both achieve sub-second convergence time

Optimal Redundancy

Redundancy is not only a question of added cost vs. uptime and resiliency, but also a question of complexity. The more hardware and software deployed in the name of redundancy adds administrative overhead and complexity, which is tough to put numbers on.

Cisco recommends:

- Redundant switches at the core and distribution layers with fully redundant links
 - Access switches should have redundant links to redundant distribution switches
 - Avoiding single points of failure as much as possible
- This can be achieved at the access layer with help from SSO (for layer 2) and potentially NSF (for layer 3)

Redundant Supervisor Engines

Providing redundant switch supervisor engines adds another level of high-availability for critical distribution and core layer devices. Redundant switch supervisor engine options are only available on Cisco Catalyst 4500 and 6500/6800 families and Nexus 7000/7700/9500 Series of switches (modular platforms). The Control plane (Route Switch Processors - RSPs or Supervisor Engines - SEs) and Data plane (with CEF shared between RSPs or SEs) are separated.

RSP is the newer version of SE. It performs:

- fabric switching
- local/remote management

RSP and SE integrate these daughter cards:

- **PFC** (Policy Feature Card): **forwarding plane in hardware** (Layer 2 switching and Layer 3 routing)
- **MSFC** (Multilayer Switch Feature Card): **control plane in software** (RSTP/VTP/CDP/Routing Protocols...), integrating RP (Route Processor) and SP (Switch Processor)

The three redundancy options are available on RSPs and SEs:

- **RPR** (Route Processor Redundancy) and RPR+
- **SSO** (Stateful Switchover)
- **NSF** (Non-Stop Forwarding)

Route Processor Redundancy

RPR was the first form of supervisor engine redundancy and is no longer the preferred option. The primary reason is the time required to failover to the backup supervisor engine.

RPR – 2 to 4 minutes on 6500 (<60 seconds on 4500)

RPR+ – takes between 30-60 seconds

RPR also **does not synchronize dynamic routing information** with the redundant supervisor engine, so all dynamic routing state information is lost upon failover. Also, upon failover the FIB tables are cleared so all dynamic routing protocols must reconverge. Only static routes will remain intact as they are manually configured.

Stateful SwitchOver

SSO is designed to minimize disruption while transitioning layer 2 services during a supervisor failover. Even a clock synchronization failure between supervisors is enough to cause a failover with SSO.

The redundant supervisor starts up in a fully initialized state and syncs with the startup and running configuration of the active supervisor engine. All subsequent changes are then also updated, allowing for seamless continuation of all supported layer two protocols.

SSO recognizes the link status of every port, so links that were active before the switchover remain active after. Neighboring devices do not see the link go down and spanning-tree remains unaffected.

On the 6500s, the switchover takes between 0-3 seconds. On the 4500 series switches it takes less than a second. Layer 3 information must be relearned however, which includes rebuilding ARP tables and layer 3 CEF and adjacency tables.

Configuring RPR/SSO

```
Switch# configure terminal
Switch(config)# redundancy
Switch(config-red)# mode {rpr | sso}
```

Verifying RPR/SSO

```
Switch# show redundancy [states]
```

Non-Stop Forwarding

Non-stop Forwarding, or NSF, is another redundancy protocol **designed to accompany SSO**. Unlike SSO, which allows seamless layer 2 transitions during a failover, NSF is designed to optimize layer 3 reconvergence after a failover. It allows a stateful switchover with redundant RSPs. When both are used, zero or near zero packets are lost during the transition. NSF also helps avoid route flapping problems by using the FIB table for failover.

NSF works by continuing to forward CEF flows while layer 3 routing protocols reconverge behind the scenes. The standby RSP maintains a copy of the CEF entries and in the event of a failover, it uses those entries to prevent a loss of traffic. After the routing has reconverged and new FIB/Adjacency tables are built, the old CEF entries are removed.

Changes have been made to many modern routing protocols (EIGRP, OSPF, IS-IS, BGP) so that upon switchover, an NSF-enabled router sends special packets that trigger routing updates from the NSF-aware neighbors without resetting the peer relationship and preventing route flapping and changes:

- **EIGRP**: Restart bit
- **OSPF**: Local Link Signaling
- **BGP**: Graceful Restart Capability

In summary, NSF improves L3 network availability and stability.

Use **NSF with SSO** in locations where you do not have a duplicate switch for failover, such as at the user access or Enterprise network edge. Otherwise it can actually cause longer convergence.

Configuring NSF

The configuration is different for EIGRP, IS-IS, and OSPF than for BGP. See the examples below:

Example:

```
Switch(config)# router {{ospf 100} | {eigrp 100}}
Switch(config-router)# nsf
Switch(config)# router bgp 10
Switch(config-router)# bgp graceful-restart
```

Hot Standby Router Protocol

Several first hop redundancy protocols (FHRP) exist including IRDP, HSRP, VRRP, and GLBP.

Default gateways are essential for devices to communicate with devices outside their local network. If the gateway is unavailable for any reason, external conversations cease. In an effort to mitigate that situation, first hop redundancy protocols have been developed to provide pairs of gateways, often one active and the other in standby, to allow an always-up default gateway.

HSRP ([RFC 2281](#)) is a redundancy protocol developed by Cisco to solve this problem. HSRP provides a virtual MAC and IP address that represents a set (2 or more) of physical routers. The virtual IP will be used as the default gateway address for the segment. The virtual IP will respond to any ARP requests for the MAC address of the default gateway with the virtual MAC address.

The Active router sends hellos (**v1: multicast 224.0.0.2 // UDP port 1985 and v2: 224.0.0.102 // UDP port 2029**) to the standby router(s) to let them know it is still up. If a standby router stops receiving hellos during hold time from the active router, it assumes the role of active and takes over forwarding packets for the network - all transparent to the end systems.

HSRP Groups → 1 Virtual IP and 1 Virtual MAC per HSRP Group

The virtual MAC used is always 0x**0000.0c07.acxx** where xx is the HSRP group ID. For example, if you see a message with 0x 0000.0c07.ac0b, the HSRP group number would be 11. The 0b HEX values after the .0c07.ac is 11 in base 10 format ("0x *number*" means that *number* is hexadecimal).

In HSRP version 2, the Virtual MAC Address related to the Virtual IP is 0x **0000.0c9f.fxxx** where xxx is the 0-4095 group ID.

There can be only a single active and single standby router in a HSRP group. After two routers, the rest stay in **Listen** state and wait for the active or standby to go down before contending for the active and standby position. The active router processes packets sent to the virtual router.

The active router in the HSRP group is determined by an election process. The router with the highest HSRP priority configured wins and if no specific priority has been set, the router with the highest IP address is elected as the active router. A new election will only occur if the active router is removed; the same is true for the standby router. This default behavior can be changed with the preempt command.

State	Definition
Initial	This is the state at the start. This state indicates that HSRP does not run. This state is entered through a configuration change or when an interface first becomes available.
Learn	The router has not determined the virtual IP address and has not yet seen an authenticated hello message from the active router. In this state, the router still waits to hear from the active router.
Listen	The router knows the virtual IP address, but the router is neither the active router nor the standby router. It listens for hello messages from those routers.
Speak	The router sends periodic hello messages and actively participates in the election of the active and/or standby router. A router cannot enter speak state unless the router has the virtual IP address.
Standby	The router is a candidate to become the next active router and sends periodic hello messages. With the exclusion of transient conditions, there is, at most, one router in the group in standby state.
Active	The router currently forwards packets that are sent to the group virtual MAC address. The router sends periodic hello messages. With the exclusion of transient conditions, there must be, at most one router in active state in the group.

** Hellos are sent in the Speak, Standby and Active states.

Before Active state, all members of the group multicast Hellos with their own BIA (Burned-In Address) MAC ↔ own IP address. They don't know who the speaker for the group is.

Once the Active router is chosen, then it:

- broadcasts gratuitous ARP with group vMAC ↔ group vIP
- multicasts Hellos with group vMAC ↔ own IP address

In each ARP response from the Active router, the Source MAC and the ARP Sender MAC Address are the vMAC address of the group. The multicasts regularly sent by the Active router give its direction to all the switches in the VLAN.

HSRP Architecture Guidelines

HSRP can only be configured on a layer 3 interface including SVIs, routed interfaces, and L3 Etherchannels:

- 1 HSRP Group per VLAN: *simple flow management*
- Multiple HSRP groups per VLAN: *load-balancing* between the routers within that VLAN
- Match *primary* (R)STP / MST RB and *Active* router on same device /VLAN
- Match *secondary* (R)STP / MST RB and *Standby* router on the other device /VLAN
- Link with DHCP configuration (default router with vIP, excluded addresses)

HSRP Virtual IP

The group number is only required if you plan on implementing more than one HSRP group on the router. If none is specified, group number 0 will be used.

```
Switch(config-if)# standby group-number ip ip-address
```

HSRP Priority

A priority value can be set to force a router to become the active router in the group. The default is **100**, and it can be manually set between 0 and 255. Higher wins. If the priority is the same, the router with the highest IP address will become active for that standby group.

```
Switch(config-if)# standby group-number priority priority-value
```

HSRP Preemption

Remember that if two routers are manually booted up at the same time, if the one with the lower priority boots up first – it will become the active router in the group even though its priority is lower. That is because it will not see any other routers when it begins the election process and will transition straight to active. Once the other router comes up, it will not automatically become active. To change this, use the **preempt** command on the router you want to be able to preempt the Active router, i.e become Active if the priority or IP address is higher than the current Active router. It is better to configure this preemption feature on all members of the group..

```
Switch(config-if)# standby group-number preempt [delay minimum seconds]
// The seconds argument causes the local router to postpone taking over the active role for a minimum number of
// seconds since the previously active router has failed. The range is from 0 to 3600 seconds (1 hour).
// The default is 0 seconds (no delay).
```

HSRP Authentication

The default password is cisco if none is specified and the password string must be the same on all members of the standby group.

```
Switch(config-if)# standby group-number authentication {password | md5 {key-chain chain | key-string [0 | 7] password}}
```

HSRP Timers

HSRP uses two important timers between the Active/Standby routers. Hello timers are used to exchange HSRP information while the hold down timer is used to determine how long before a router is declared to be down in a group. The default hello times are **3 seconds** and the default hold down timer is **10 seconds**. That means there could be up to a 10 second delay before the standby router begins forwarding traffic if the Active goes down. To tune the timers (in seconds [or msec]):

```
Switch(config-if)# standby group-number timers [msec] hello-time [msec] hold-time
```

Note: If you are noticing the HSRP states frequently change, you may have a physical layer problems, a spanning-tree loop or the timers are set too low. If you notice the output, “Standby router is unknown expired”, you likely have an HSRP misconfiguration or a physical layer issue.

Convergence Delay: Layer 2 + Layer 3 delays → STP/RSTP delays + HSRP hold time

HSRP Tracking

Tracking a critical uplink interface can force a re-election by decrementing the Active router's priority value by a set amount (**default 10**).

Or tracking an object linked to IP SLA or IP route (checking Internet connectivity for example) can force the Active router to become Standby thus letting another router takeover the link to Internet.

```
Switch(config-if)# standby group-number track {interface | tracking-object-id [decrement priority-to-decrement]}  
// Some old IOS releases do not need the "decrement" keyword before priority-to-decrement
```

HSRP Versions

HSRP comes in two versions, 1 and 2. The most significant difference is that v1 only allows up to 255 group numbers and v2 allows up to 4095 – making it now possible to correspond group numbers with VLAN IDs.

```
Switch(config-if)# standby version [1|2]
```

HSRP Verification

```
Switch# show standby [{vlan id} | {interface id} | all | brief | neighbors]  
Switch# debug standby [errors | events]
```

HSRP for IPv6

HSRPv2 supports IPv6. An HSRP IPv6 group has a vMAC address that is derived from the HSRP group number, and a vIPv6 link-local address that is, by default, derived from the HSRP vMAC address. Periodic RAs are sent by HSRP Active router for the vIPv6 link-local address when the HSRP group is Active. Hosts default gateway IPv6 address is dynamically configured from the RAs.

HSRPv6 uses a different virtual MAC address block than does HSRP for IP: 0005.73A0.0000 through 0005.73A0.0FFF (4096 addresses)

UDP Port number 2029 (same as HSRP for IPv4) has been assigned to HSRPv6.

On MLAG topologies, you must enable both L3 devices to be Active.

Virtual Router Redundancy Protocol

VRRP is an open standard redundancy protocol ([RFC 5798](#) **obsoletes RFCs 3768 and 2338**) that is similar to Cisco's HSRP. One difference is that the virtual IP can either be a virtual one (as is the case with HSRP) or it can be the actual IP address of the active router: "*IP address owner*".

The VRRP "Master" forwards the traffic and is chosen because it owns the real IP address or has the highest priority (default is again 100). One of the "Backup" routers takes over if the Master fails. Priority values are between 1-255. If the Master router fails, it advertises a priority of 0, forcing an election amongst the Backup routers without waiting for the hold down timer to expire.

The Master router sends hellos (**multicast 224.0.0.18 // VRRP IP protocol 112**) to the Backup router(s) to let them know it is still up. If a Backup router stops receiving hellos during hold time from the Master router, it assumes the role of Master and takes over forwarding packets for the network - all transparent to the end systems.

Note: Not supported by 35xx switchs, but by 45xx, 65xx/68xx and Nexus Series

VRRP Groups → 1 Virtual IP and 1 Virtual MAC per VRRP Group

The virtual MAC used is always 0x **0000.5e00.01xx** where xx is the VRID, Virtual Router IDentifier, aka VRRP Group ID.

Once the Master router is chosen, then it:

- broadcasts gratuitous ARP with group vMAC ↔ group vIP
- multicasts Hellos with group vMAC ↔ own IP address

In each ARP response from the Master router, the Source MAC and the ARP Sender MAC Address are the vMAC address of the group. The multicasts regularly sent by the Master router give its direction to all the switches in the VLAN.

VRRP Architecture Guidelines

VRRP can only be configured on a layer 3 interface including SVIs, routed interfaces, and L3 Etherchannels:

- 1 VRRP Group per VLAN: *simple flow management*
- Multiple VRRP groups per VLAN: *load-balancing* between the routers within that VLAN
- Match *primary* (R)STP / MST RB and *Master* router on same device /VLAN
- Match *secondary* (R)STP / MST RB and *Backup* router on the other device /VLAN
- Link with DHCP configuration (default router with vIP, excluded addresses)

VRRP Virtual IP

```
Switch(config-if)# vrrp group-number ip virtual-ip-address [secondary]
// You can configure the primary IP address of a VRRP group with the same address as the interface.
// Then, the router that has the interface IP address is always the master router when UP: IP address owner :
```

VRRP Priority

A priority value can be set to force a router to become the Master router in the group. The default is **100**, and it can be manually set between 0 and 255. Higher wins. If the priority is the same, the router with the highest IP address will become Master.

```
Switch(config-if)# vrrp group-number priority priority-value
```

VRRP Preemption

To be able to preempt the Master router, i.e become Master if the priority or IP address is higher than the current Master router (**enabled by default**):

```
Switch(config-if)# vrrp group-number preempt [delay minimum seconds]
// Number of seconds that the router will delay before issuing an advertisement claiming master ownership.
// The default delay is 0 seconds.
```

VRRP Authentication

2/26/15

The password string must be the same on all members of the group.

```
Switch(config-if)# vrrp group-number authentication {password | md5 {key-chain chain | key-string [0 | 7] password}}
```

VRRP Timers

- Advertisements, or hellos – default **1 second**
- Master down interval = 3 * (the advertisement timer) + **skew**
- **Skew** time = $(256 - \text{priority-of-local-backup-router}) / 256$.

Used to ensure that two backup routers do not switch to the master state at the same time in case of a failover situation.

Setting the advertisement timer on the master:

```
Switch(config-if)# vrrp group-number timers advertise [msec] advertisements-timer
```

Learning the advertisement timer from the master:

```
Switch(config-if)# vrrp group-number timers learn  
// By default, the local router uses the advertisement timer as configured by the "vrrp timers advertise" command.
```

VRRP Tracking

VRRP cannot track interface changes, but can track IP SLA / IP route / interface object groups. The **default decremented value is 10**.

```
Switch(config-if)# vrrp group-number track tracking-object-id [ decrement priority-to-decrement]
```

VRRP Verification

```
Switch# show vrrp [interface | all | brief]  
Switch# debug vrrp {all | authentication | error | events | ha | state}
```

VRRP for IPv6

VRRPv3 supports IPv6. The functioning is similar to HSRP for IPv6.

Gateway Load Balancing Protocol

One of the major limitations to both HSRP and VRRP is that a single router handles traffic for the whole group, leaving the others inactive until the master router fails. GLBP or Gateway Load Balancing Protocol (Cisco proprietary - no RFC) solves this dilemma by load balancing traffic over *up to four gateways*, maximizing bandwidth. One virtual IP is used, but each participating router uses a *different virtual MAC* address which is used to respond to ARP requests.

*Notes: GLBP is only supported on Cisco Catalyst 45xx, 65xx/68xx, and Nexus Series.
GLBP supports IPv6.*

GLBP Groups → 1 Virtual IP per GLBP Group (per VLAN) and 1 Virtual MAC per AVF

The routers running GLBP elect a single **Active Virtual Gateway (AVG)**, which manages the load balancing and responds to ARPs sent to the vIP. The highest priority router wins; in a tie the highest IP address wins. Group members send hello multicasts every **3 seconds** at **multicast address 224.0.0.102 and UDP port 3222**; if a router goes down, another will answer its requests.

The job of the AVG is to assign virtual MAC addresses to each member of the GLBP group and to assign each network host to one of the GLBP routers. The routers that receive the vMAC address assignment are the **Active Virtual Forwarders**, or AVFs.

Once the Active Gateway is chosen, then the AVG and each AVF:

- *do not* broadcast gratuitous ARP because the group vIP is shared among multiple vMACs and the hosts are configured with the group vIP.
- multicast Hellos with its own vMAC ↔ own IP address

In each ARP response from the Active AVG router, the Source MAC address is its BIA MAC address and the ARP Sender MAC Address field is the vMAC address (0x **0007.b400.xxyy** - xx: group-id and yy: router-id) of the AVF assigned to this host by the AVG..

The multicasts regularly sent by the Active AVF routers give their directions to all the switches in the VLAN.

In case of an AVF failure, another AVF takes over its vMAC so that the clients once served by the failed gateway will not experience any communication disruption.

Up to 1024 different groups can be defined per VLAN.

GLBP Architecture Guidelines

GLBP can only be configured on a layer 3 interface including SVIs, routed interfaces, and L3 Etherchannels:

- 1 GLBP group per VLAN: *load-balancing* between the routers within that VLAN
- Link with DHCP configuration (default router with vIP, excluded addresses)

GLBP Virtual IP

```
Switch(config-if)# glbp group-number ip virtual-ip-address
```

GLBP Priority → AVG Election

A priority value can be set to force a router to become the AVG router in the group. The default is **100**, and it can be manually set between 0 and 255. Higher wins. If the priority is the same, the router with the highest IP address will become AVG.

```
Switch(config-if)# glbp group-number priority priority-value
```

GLBP AVG Preemption → AVG Election

To be able to preempt the AVG router, i.e become AVG if the priority or IP address is higher than the current AVG router:

```
Switch(config-if)# glbp group-number preempt [delay minimum seconds]  
// delay minimum seconds: Specifies a minimum number of seconds that the router will delay before taking over  
// the role of AVG. The range is from 0 to 3600 seconds with a default delay of 30 seconds(!).
```

Note. On 3725 IOS 12.4(15)T5, avoid delay = 0s: otherwise, the AVG election process is disturbed (bug) – the AVG does not relinquish its role when it should be preempted by another gateway with a higher priority or IP address.

GLBP Load-Balancing

Three load sharing options

Weighted load balancing- based on preconfigured weights assigned to AVFs (cf. next paragraph); the amount of load directed to an AVF is dependant upon its weighting value. Weighted can be used if there are disparities in the capabilities of gateways in the GLBP group.

Host-dependent load balancing – each host always uses the same specific AVF . Host Dependent is needed when an application requires traffic flows to be tracked (for example when using NAT).

Round-robin load balancing – Each vMAC is used to respond in turn. Round Robin is recommended as the **default**, it is suitable for all other requirements.

```
Switch(config-if)# glbp group-number load-balancing {round-robin | host-dependent | weighted}
```

GLBP Weighting and Tracking → Impact on AVF Role and Election

The weighting value of a virtual gateway is a measure of the forwarding capacity of the gateway.

If a tracked interface on the router fails, the weighting value of the router may fall from the maximum value to **strictly below the lower threshold (1 by default)**, causing the router to give up its role as a virtual forwarder. Other gateways will detect the event using Hold timer expiration, and immediately try to take over the failed AVF virtual MAC address. Among the competitors, the **AVF with highest weight value wins**, and the remaining AVFs will switch back to “Listen” state. At this point, the new AVF will start accepting packets for two virtual MAC addresses: its own, and the one it has obtained from the failed AVF.

When the weighting value of the failed router rises **above or at the same level as the upper threshold** (100 by default), the router can resume its Active Virtual Forwarder role.

```
Switch(config-if)# glbp group-number weighting max-weighting-value [lower lower-value] [upper upper-value]
// Default max-weighting-value is 100
// Default lower-value is 1
// Default upper-value is max-weighting-value
Switch(config-if)# glbp group-number weighting track object-number [decrement weighting-value-to-decrement]
// When multiple tracked interfaces are down, the priority is reduced by a cumulative amount
// Default weighting-value-to-decrement is 10
```

GLBP AVF Preemption → AVF Election

2/26/15

```
Switch(config-if)# glbp group-number forwarder preempt [delay minimum seconds]
// Configures a router to take over as AVF if the current AVF falls strictly below its low weighting threshold.
// By default, already the case with seconds = 30s;
```

Note. **On 3725 IOS 12.4(15)T5, avoid delay = 0s: otherwise, the AVF election process is disturbed (bug)** – AVF does not relinquish its AVF role when its weight falls strictly below its lower threshold and it should be preempted by another AVF.

GLBP Authentication

The password string must be the same on all members of the group.

```
Switch(config-if)# glbp group-number authentication {password | md5 {key-chain chain | key-string [0 | 7] password}}
```

GLBP Timers

By default, the Hello timer is **3s** and the hold time is **10s**.

```
Switch(config-if)# glbp group-number timers [msec] hello [msec] hold
```

GLBP Verification

```
Switch# show glbp [{vlan id} | interface | brief]
Switch# debug glbp {errors | events | terse}
```

IRDP

Some newer hosts use the ICMP Router Discovery Protocol (**RFC 1256**) to find a new router when a route becomes available.

A host running IRDP listens for hello multicast messages from its configured router and uses an alternate router when that router is no longer available.

ICMP Router Discovery Protocol (IRDP) allows hosts to locate routers that can be used as a gateway to reach IP-based devices on other networks. When the device running IRDP operates as a host, *router Solicitation* packets are sent (ICMP Type 10). When the device running IRDP operates as a router, *router Advertisement* packets (ICMP Type 9) are generated.

Configuring the DHCP Server

Assigning IP Addresses from a Network Pool

```
Switch(config)# service dhcp
Switch(config)# ip dhcp pool network-pool-name
Switch(dhcp-config)# network subnet-ip-address [mask | /prefix-length]
Switch(dhcp-config)# default-router router1-ip-address router2-ip-address ... router8-ip-address
// These IP addresses are NOT allocated to hosts from the pool
Switch(dhcp-config)# dns-server dns-server1-ip-address dns-server2-ip-address ... dns-server8-ip-address
Switch(dhcp-config)# domain-name domain
Switch(dhcp-config)# lease days hours mins seconds | infinite
// Default: 24 hours
Switch(config)# ip dhcp excluded-address first-ip-address [last-ip-address]
Switch(config)# ip dhcp database url timeout seconds write-delay seconds
// Database location
```

Configuring Options: RFC 2132

```
Switch(dhcp-config)# option 43 hex hex-string
// Option 43: used for handing out IP addresses of WLCs (other than Cisco Aironet 1000 Series) to LWAPs
// TLV Format: 0xF1 - Number of WLCs * 4 - Hexadecimal IP addresses
Switch(dhcp-config)# option 60 ascii "VCI string"
// Option 60: used by LWAPs to identify themselves to the DHCP server with a Vendor Class Identifier (VCI)
// Only the LWAPs configured with option 60 will be able to use option 43. You must configure a separate DHCP pool
// for each access point type. If option 60 is not used on DHCP Server, all LWAPs will be served with option 43.
Switch(dhcp-config)# option 150 ip {tftp-server-1 [tftp-server-2]}
// Option 150: Defines the TFTP Server IP addresses (can be some Unified Communication Manager Express UCME router
// IP addresses) from which the Cisco IP phones can download the image configuration file
```

Assigning one IP Address per client-id

```
Switch(config)# service dhcp
Switch(config)# ip dhcp pool host-pool-name
Switch(dhcp-config)# client-identifier client-id <<<
// client-id same as the one from "#debug ip dhcp server packet" <<<
Switch(dhcp-config)# host host-ip-address [mask | /prefix-length] <<<
Switch(dhcp-config)# default-router router1-ip-address router2-ip-address ... router8-ip-address
Switch(dhcp-config)# dns-server dns-server1-ip-address dns-server2-ip-address ... dns-server8-ip-address
Switch(dhcp-config)# domain-name domain
Switch(dhcp-config)# client-name hostname <<<
Switch(dhcp-config)# lease days hours mins seconds | infinite
Switch(config)# ip dhcp database url timeout seconds write-delay seconds
```

Configuring the DHCP Relay

```
Switch(config)# service dhcp
// Necessary to insert the DHCP Relay IP address into the relayed DHCP Discover
Switch(config-if)# ip helper-address server-ip-address
// This router relays some UDP broadcasts received on the specified interface towards that server
// By default, DHCP, BOOTP, DNS, TFTP, TACACS are relayed when the aforementioned command is used
Switch(config)# ip forward-protocol udp [port]
// Controlling which protocols are relayed by this router
Switch(config)# ip dhcp relay information trust-all
// Preventing the DHCP relay to discard the DHCP packets even if the gateway address (giaddr) is set to all zeros
```

Configuring the DHCP Client

```
Switch(config-if)# ip address dhcp [client-id interface]
// client-id: Optionally uses this interface MAC address as client-id
// Beware: Cisco adds '01' before client MAC address to form client-id
Switch(config-if)# ip dhcp client client-id {interface | {ascii string} | {hex number}}
// Another way to configure the client-id
Switch(config-if)# ip dhcp client class-id {string | {hex number}}
// Option 60: a string or hexadecimal number
```

Verifying DHCP on the Server

```
Switch# show ip dhcp binding
// All leased IP addresses ↔ client-id ↔ lease ↔ type
Switch# show ip dhcp pool
// All DHCP pools with current index and number of leased IP addresses
Switch# show ip dhcp statistics
Switch# show ip dhcp database
Switch# show ip dhcp conflict

Switch# debug ip dhcp server {events | packet}
```

High Availability Design Guide

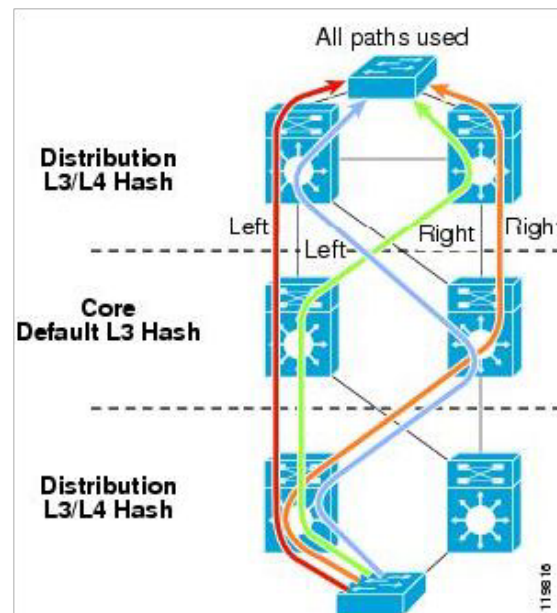
2/26/15

Access Layer Tuning

- Limit VLANs to a single closet whenever possible: **Local VLAN model**,
- If STP is required, use **Rapid PVST+**,
- Set trunks to on/on with no negotiate (2s tuned away), **prune unused VLANs**, and use VTP transparent mode,
- **Consider EIGRP/OSPF Routing in the access layer.**

Distribution Layer Tuning

- Use **equal-cost redundant connections to the core for fastest convergence** and to avoid black holes,
- Connect distribution nodes to facilitate **summarization** and L2 VLANs spanning multiple access layer switches where required, with **L2 link between the distribution layer switches in the latter case**,
- Utilize **GLBP/VRRP/HSRP millisecond timers**,
- Ensure that the distribution node has connectivity to the core before it preempts its HSRP/VRRP/GLBP standby/backup peer so that traffic is not dropped while connectivity to the core is established,
- **Tune EtherChannel (L3/L4 information) and CEF (L3 information in the core, L3/L4 information at the distribution layer) load balancing** to ensure optimum utilization of redundant, equal-cost links.



Core Layer Tuning

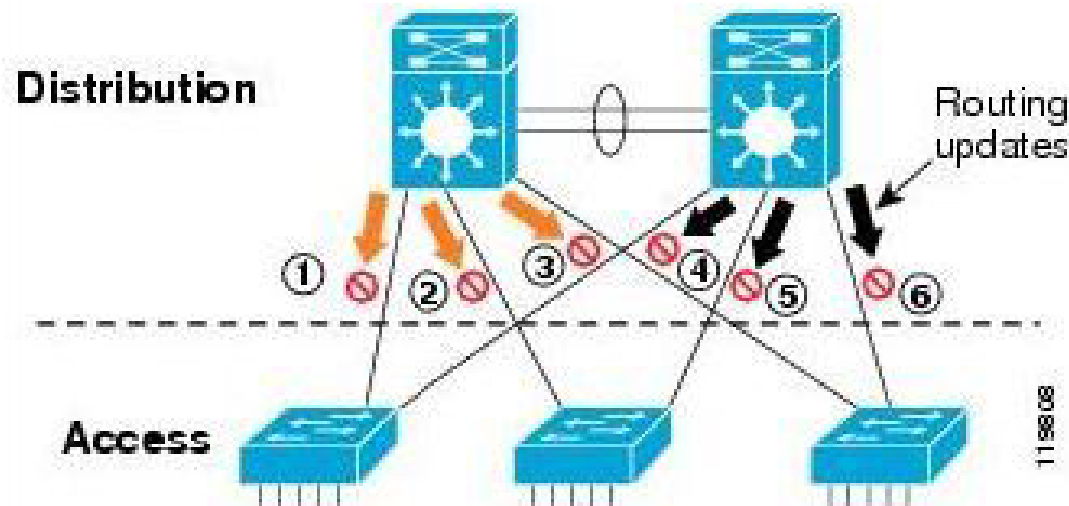
Topologies with **redundant equal-cost load sharing links** are the most deterministic and optimized for convergence measured in milliseconds.

General Design Considerations

- **Design for deterministic convergence—triangles, not squares.**
Physical link up/down is faster than timer-based convergence.
- **An L3 link is required between the distribution nodes and between the core nodes,**
- **Limiting L3 Peering to Transit Links:**

The distribution routers, based on the default configuration, can establish a peer relationship through the access layer for each VLAN supported by the distribution pair. This can cause unexpected and unwanted IGP behavior. And it consumes extra memory, CPU and bandwidth.

Layer 3 Peer Relationships



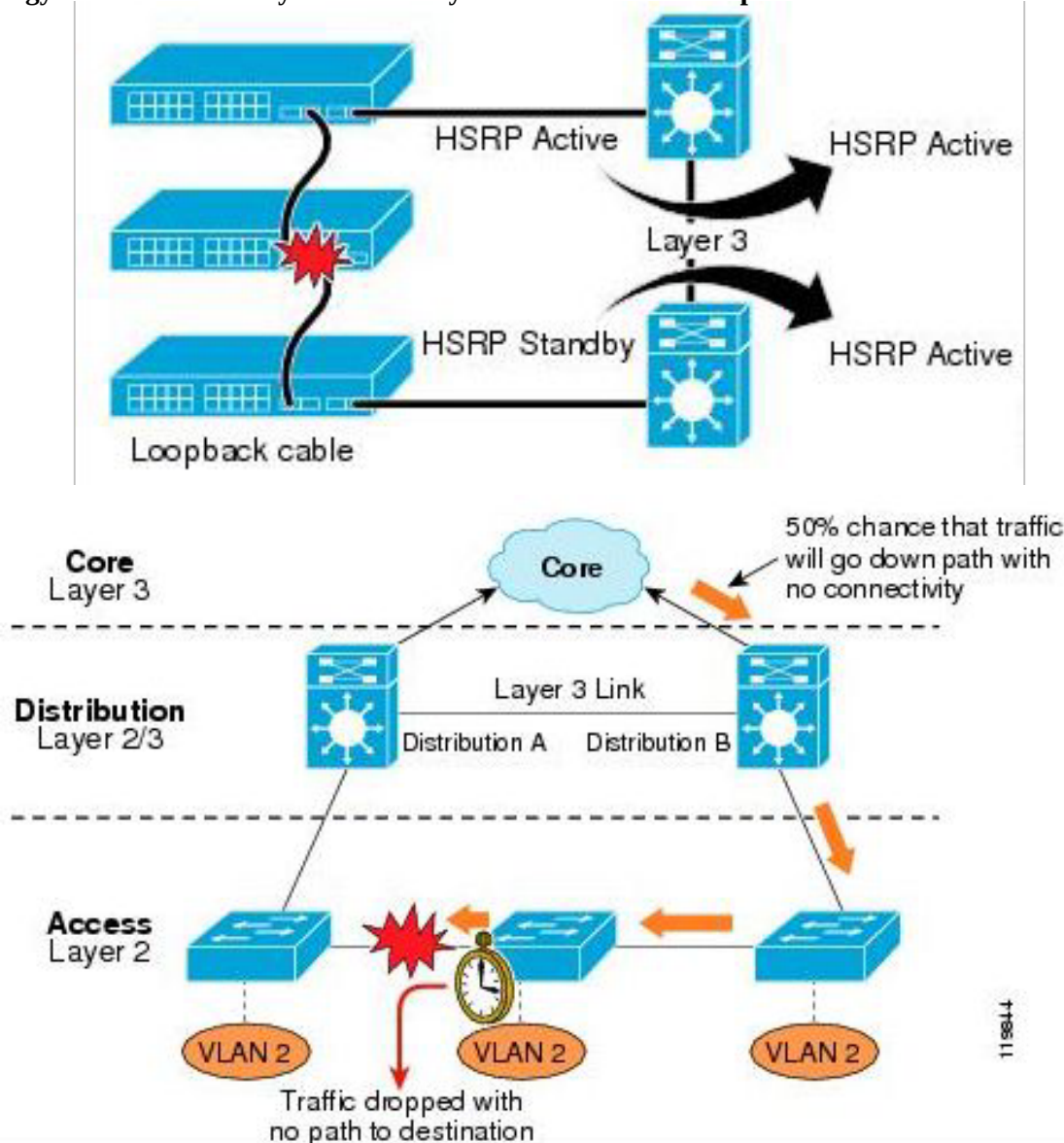
- Only span VLANs across multiple access layer switches if you must.
- Keep redundancy simple.
- Use UDLD to protect against one-way up/up connections,
- Deploy QoS end-to-end.

- **Avoid daisy chaining stackable switches;** stacks are good, **StackWise and chassis solutions are better:**

Daisy-chained fixed configuration implementations add complexity. Without careful consideration, discontinuous VLAN/subnets, routing black holes, and active/active HSRP/GLPB situations can exist:

In case a link between access switches fails and a VLAN spans the daisy chained access switches, return path traffic has a 50/50 chance of arriving on an access switch that does not have physical connectivity to the half of the stack where the traffic is destined. Traffic is dropped when it arrives on the wrong access switch.

Use StackWise technology in the Cisco Catalyst 3750 family or modular chassis implementations to avoid these complications.



- **Avoid asymmetric routing and unicast flooding; do not span VLANs across the access layer if possible.**

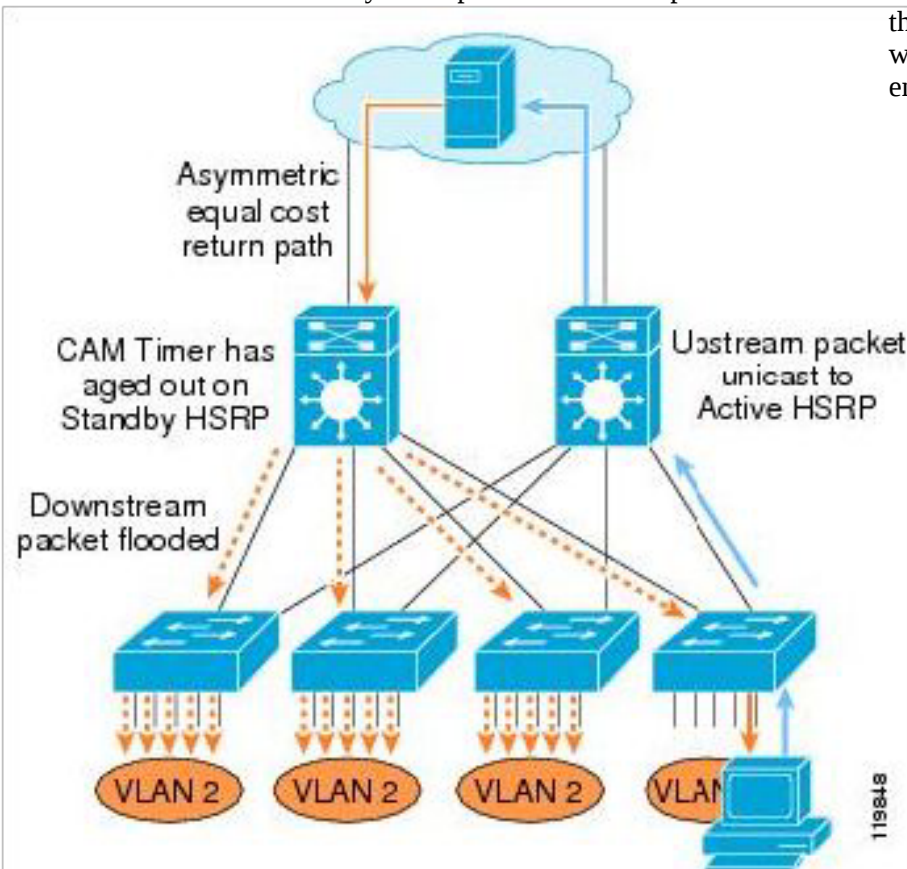
Traffic returning through the standby HSRP, backup VRRP, or alternate/non-forwarding GLBP peer can be flooded to all ports in the target VLAN when you use a topology in which VLANs are spanned across multiple access layer switches. This can have **significant impact on performance**.

In the following topology, the **CAM table entry ages out on the standby/backup/alternate FHRP router**. This occurs because the **ARP (4 hours) and CAM (5 mn) default aging timers are different**. The CAM timer expires because no traffic is sent upstream towards the standby/backup/alternate FHRP peer after the end point initially ARPed for its default gateway. When the CAM entry has aged out and is removed, the standby/backup/alternate FHRP peer must forward the return path traffic to all ports in the common VLAN.

The corresponding access layer switches also do not have a CAM entry for the target MAC, and they also broadcast the return traffic on all ports in the common VLAN. This traffic flooding can have a performance impact on the connected end stations because they may receive a large amount of traffic that is not intended for them.

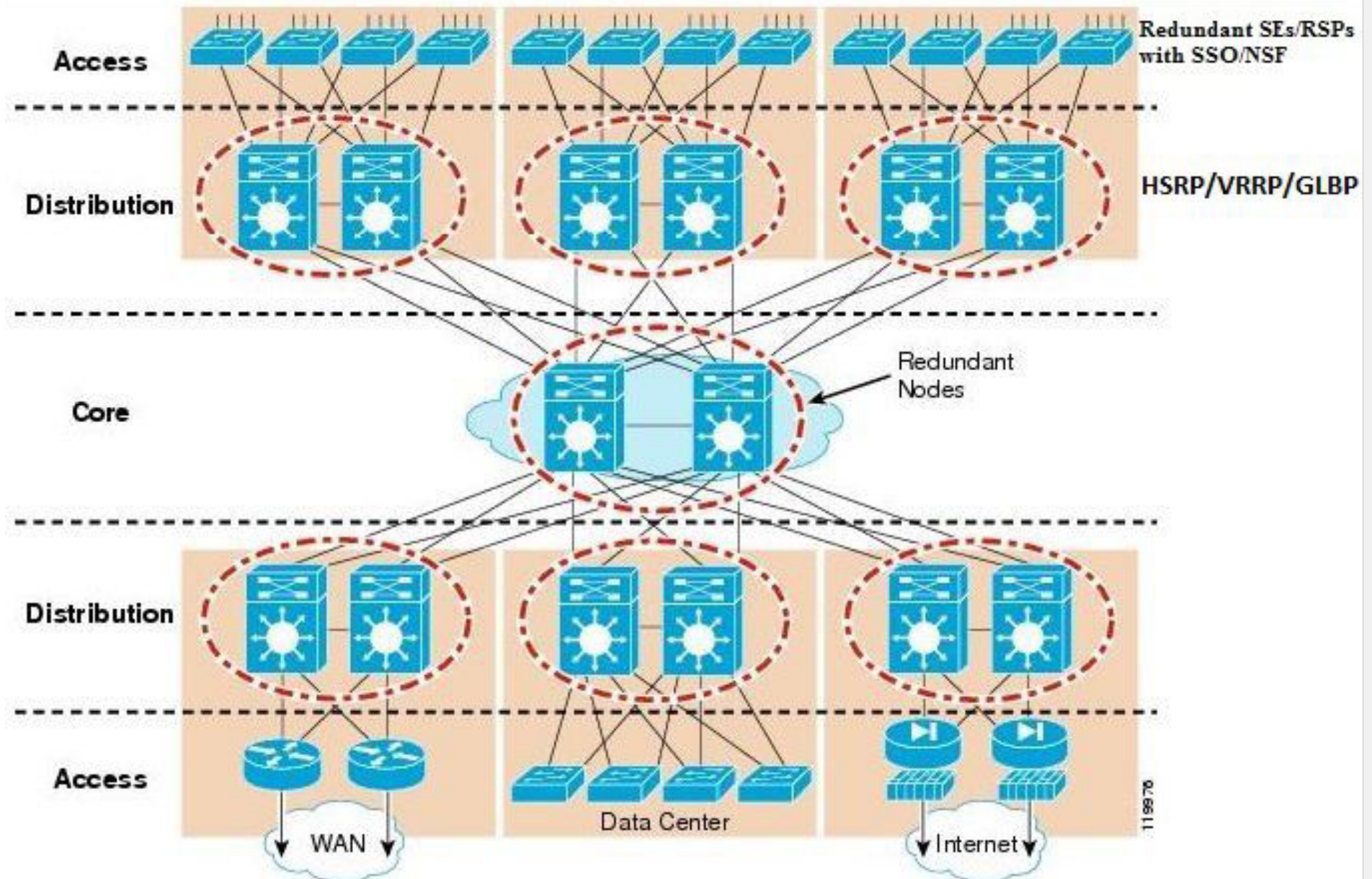
If you must implement a topology where VLANs span more than one access layer switch, **my recommended solution is to use a routed access layer with VXLAN**. **Tuning the ARP timer to be equal to or less than the CAM aging timer on the FHRP routers would make this problem much worse by generating the same broadcast flooding issue every 5 mn for example.**

A shorter ARP cache timer would cause the standby/backup/alternate FHRP peer to ARP for the target IP address before the CAM entry timer expires and the MAC entry is removed. The subsequent ARP response would repopulate the CAM table before the CAM entry is aged out and removed but **at a very high cost**.



```
Switch(config-if)# arp timeout seconds
// Default: 14400s (4 hours)
```

Hierarchical Network Design Model



Cisco
642
813

Chapter 9

Wireless

Wireless LANs (WLAN) transmit using Microwaves (300 MHz - 300 GHz), often through an access point. One interesting point is that for the spectrums covered here (2.4 GHz and 5 GHz), there are usually no additional RF licenses required. They are limited in physical transmission distance (ex. within a floor, department, or campus) and are considered by Cisco an extension of the wired campus network.

The Cisco Unified Wireless Network

Cisco's wireless architecture model includes 5 layers (current release is 7.5):

1. **Client devices** – Wireless end clients (ex. laptop, smart phones)
2. **Mobility platform** – Access points and wireless bridges
3. **Network unification** – Existing wired network (Ex. routers, switch, WLAN controllers)
4. **Network management** – WLAN location, management and security (Cisco Wireless Control Systems [WCS], replaced by Cisco Prime NCS (Network Control System) is Cisco's solution here)
5. **Mobility services** (also called **Unified Advanced Services**) – advanced products and services like wireless IP phones, RF firewalls, and location appliances

Note: Cisco offers wireless IP phones with the same feature set found in desk phones, including optional LEAP authentication.

The Cisco Compatible Extensions Program

The Cisco Compatible Extensions (CCX) program ensures the widespread availability of client devices that are interoperable with a Cisco WLAN infrastructure. You may notice a "Cisco Compatible" sticker on the device or its packaging.

Wireless LAN Attributes

Wireless access points provide client connectivity similar to what a *hub* would do in a wired infrastructure. Radio waves are the physical medium as opposed to wires and the same network and application layer protocols can run over a WLAN network (ex. IP, HTTP, etc.).

Some specific considerations

- WLANs use Carrier Sense Multiple-Access/Collision Avoidance (CSMA/CA)
- It is half duplex.
- Collisions can't be detected (can't listen while sending) (as of today, but there's been a recent development in that field of research) ⇒ CSMA/CA tries to avoid collisions before they happen with *Distributed Coordinated Function* (DCF) algorithm with following characteristics:
 - Random Backoff timers
 - *Optional* RTS (request to send) and CTS (clear to send) messages
 - Recipients of messages need to send an ACK if transmission is OK.

There is innovation in progress in decoding original transmission within interfered signals.

- RF is susceptible to interference, distortion, and noise often caused by physical structures and specific materials.
- WLAN design should include the fact that clients are often mobile and use batteries.
- Different countries have unique rules and standards regarding RF implementations.
- Antennas are characterized by polarization, gain, and directionality and antenna power is measured in dBi

SSIDs

Service Set Identifiers or SSIDs map a network, by VLAN, to a specific segment of users. The segment of users can have specific QoS or security assigned to them when they associate with the SSID. The SSIDs are often broadcasted by wireless access points, but can also be simply statically configured on a host device.

Notes: SSID names are case sensitive, so inconsistent case in an SSID configuration can result in a failed connection.

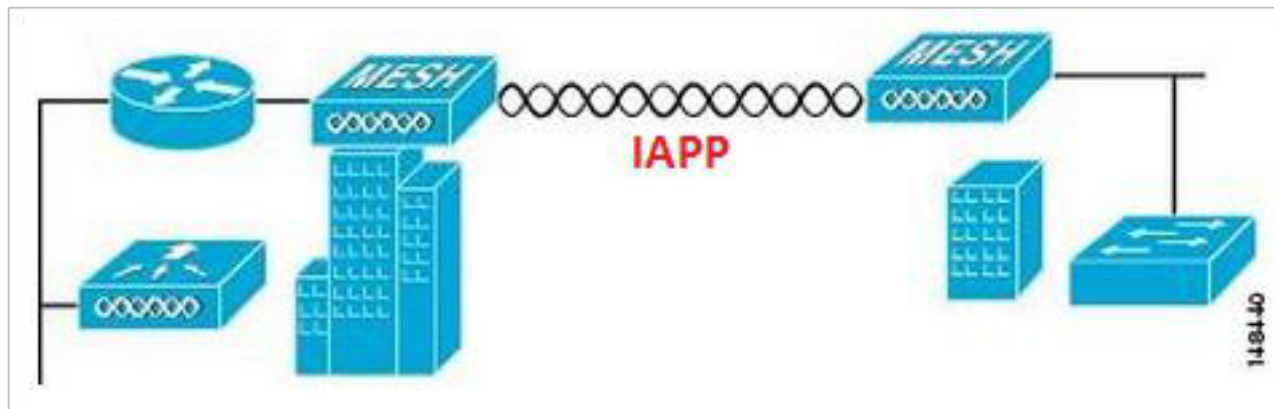
Another important point regarding SSID configuration is when an autonomous AP is hosting multiple SSIDs (and in turn multiple VLANs), the link back to the switch must be a trunk that supports all the VLANs.

Wireless Topologies

There are three main types of WLAN topologies used today:

- Client access (think end devices connecting to an AP)
- Point-to point (ex. building-to-building)
- Mesh

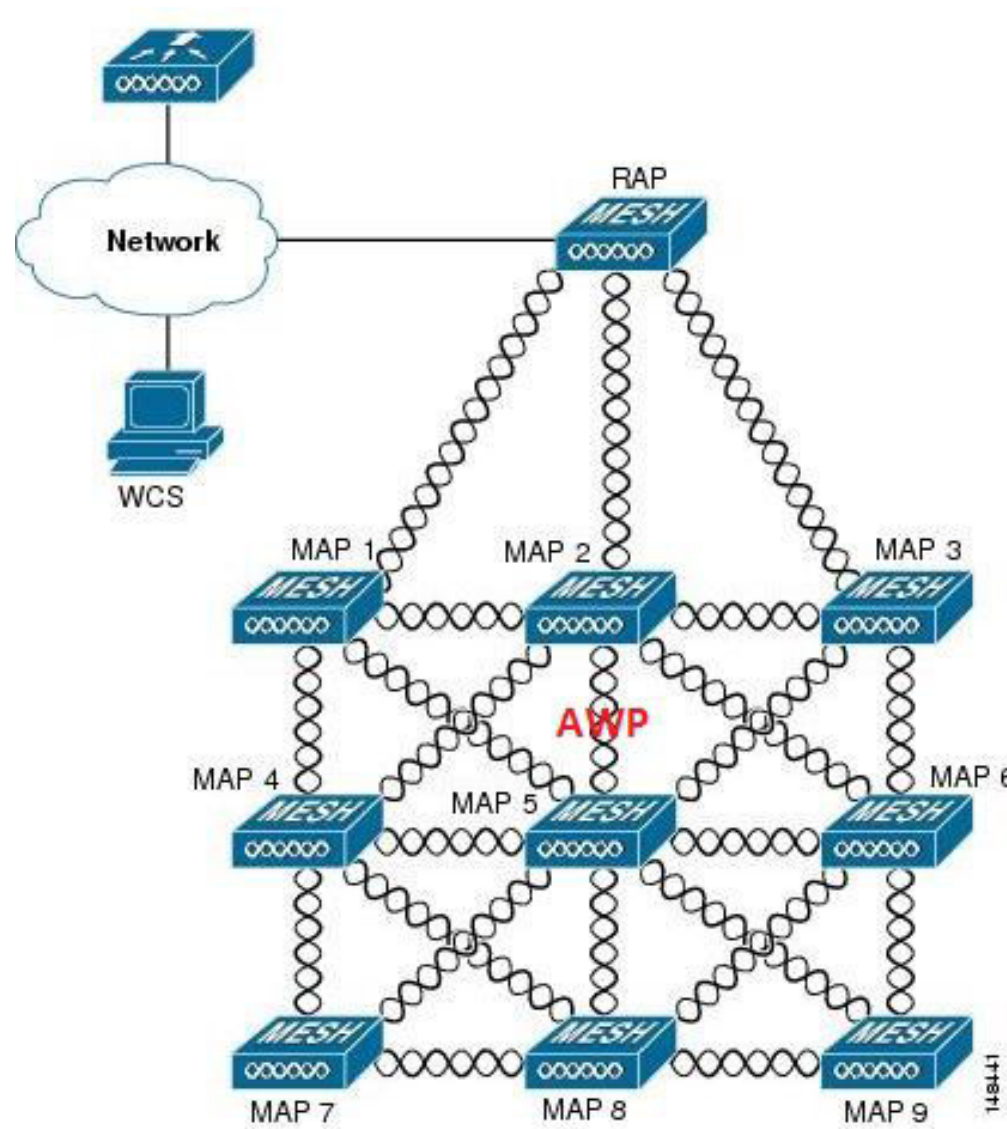
Point-to-Point



IAPP: Inter-Access Point Protocol => Allows Layer 2 Roaming

Mesh

Wireless mesh networks are usually designed for long distances. Only the APs on the edges of the mesh network connect to the wired infrastructure – the rest hops AP to AP, each acting as a repeater. Each intermediate AP has multiple paths through the mesh network, all coordinated by the **Adaptive Wireless Path** (AWP) protocol. AWP chooses the best path for traffic to the wired network and also select a backup path in case the preferred path fails.



RAP: Rooftop Access Point

MAP: Mesh Access Point

There are two modes of connection:

Infrastructure

- **Basic Service Set [BSS]** – One AP to connect clients, so the signal range (known as its microcell) must encompass all clients
- **Extended Service Set [ESS]** – Multiple APs with overlapping microcells connected by a common distribution system
 - Microcells should overlap by 10-15% for data
 - Microcells should overlap by 15-20% for voice
 - Each AP should use a different channel
- **BSSID:** MAC Address

Ad-Hoc

- **Independent Basic Service Set [IBSS]**
- Clients communicate directly with each other without the use of an access point
- Limited in range and function

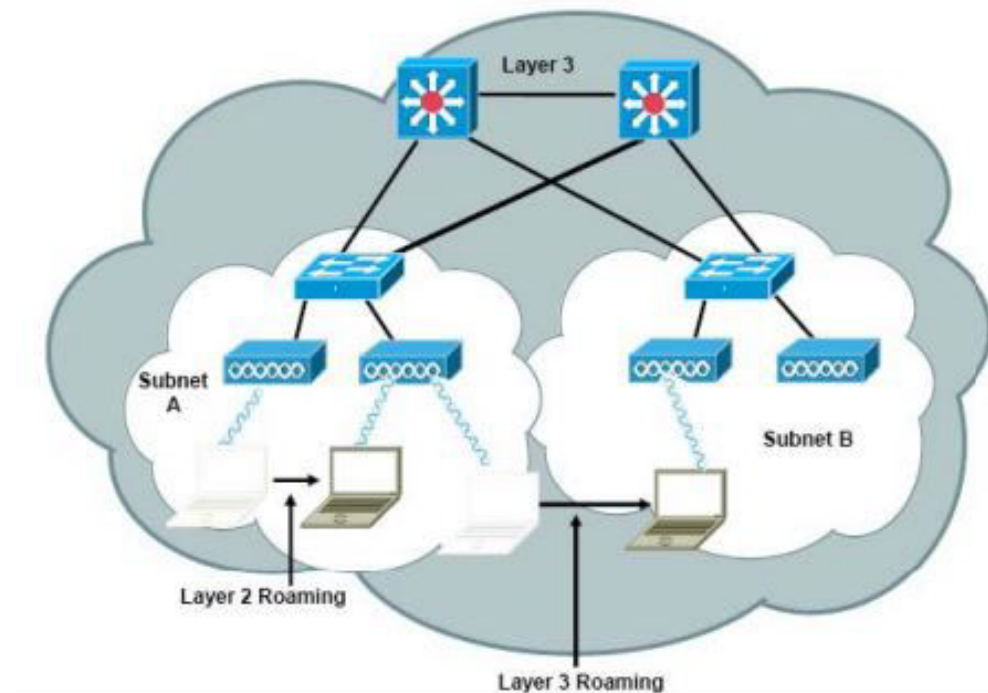
Client Connectivity

The following steps define how clients connect to an access point. Keep in mind that APs send out beacons with SSID information at regular intervals unless configured otherwise.

1. **Clients sends probe request** and listens for probe responses and beacons
2. AP replies to the request with a **probe response**
3. **Client then initiates an association with the access point.** During the association, 802.1X or PSK authentication and any other necessary **security information** is passed to the AP:
 - WEP: WEP/RC4
 - WPA: TKIP/RC4
 - WPA 2: CCMP/AES
4. **AP accepts the association.** MAC address and SSID information is exchanged between the two.
5. **AP adds client's MAC to its association table**

When ESS Infrastructure mode is in use, clients can roam (associate with another AP) between APs, but the access points must be configured with the same SSID/VLAN information and security settings. **Layer 2 roaming is done on autonomous APs using Inter-Access Point Protocol (IAPP)** with multicast. **Layer 2/3 roaming is performed on LWAPs using wireless LAN controllers.**

Note: VoIP over WLAN is susceptible to latency and jitter problems. This is a particular issue when roaming between APs, so short roaming times are critical.



A client will automatically attempt to roam if any of the following are present:

- Data rate is reduced
- Misses too many beacons from the associated AP
- Max data retry count is exceeded
- If configured to search for another AP at regular intervals

Client L2 roaming requires the following configured:

- All APs should be configured with identical VLANs
- All APs should be configured with identical subnets
- All APs should be configured with identical SSIDs
- All APs should be configured with identical security settings

WLAN Components

Cisco supports two different types of wireless access points, **autonomous and lightweight**. Autonomous systems are able to provide wireless services independently and lightweight models work in combination with a wireless controller. Both variations can receive their power from Power over Ethernet (**PoE**) switches or midspan power injectors which inject power into a cable run. Both of these options are important because they prevent the need for electrical outlets near an AP, giving more flexible location options. Note that access points can require up to 15 Watts of power (802.3af), so if you are running PoE, make sure the switch can power the number of APs connected.

The Cisco Aironet 1250 Series AP needs a switch that provides enhanced power (20 Watts - 802.3at a.k.a **PoE+** which delivers up to 30 Watts per port).

Autonomous APs

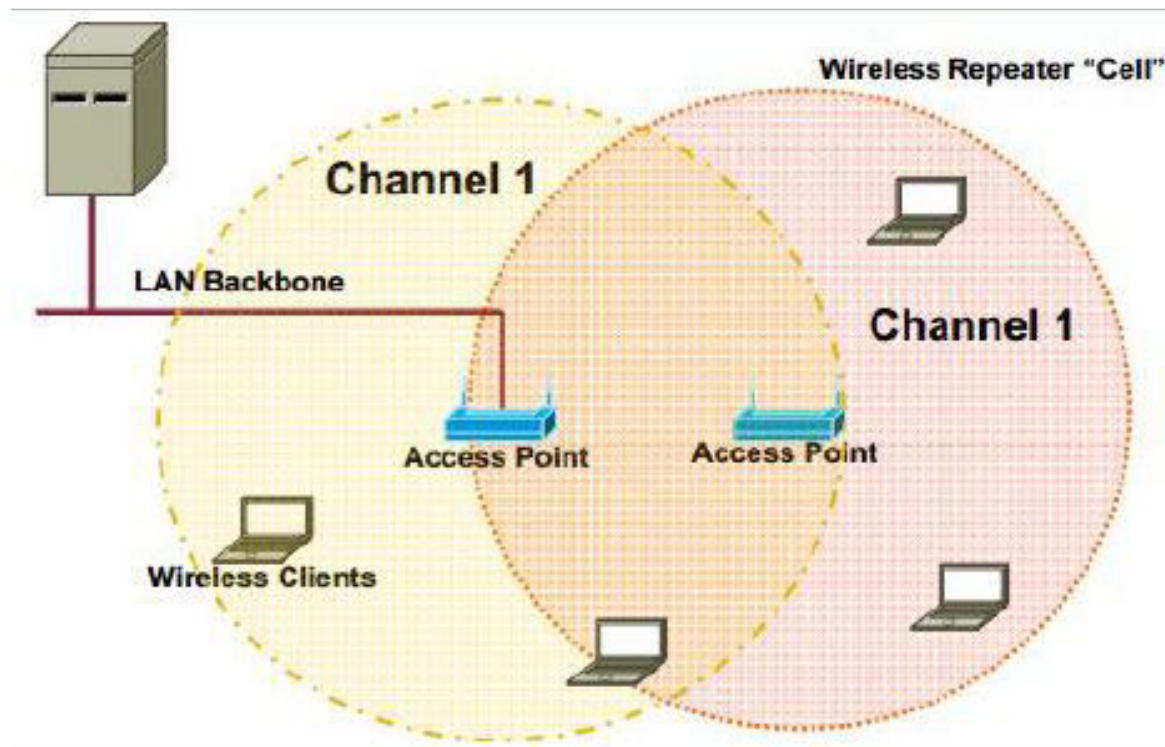
2/26/15

Autonomous APs run Cisco IOS and are configured directly. The traffic flows from client, to autonomous AP, to connected switch, to the rest of the network. If roaming is a requirement, make sure proper VLANs and SSIDs are configured (make sure a management VLAN is included). Also, **only layer 2 roaming is possible on autonomous APs**. Make sure the switch has power and remember to configure the connected switch interface as a trunk if you are using multiple VLANs. Redundancy is provided by multiple APs.

Repeaters

Repeaters are access points configured to extend the radio range of an existing wireless network. The repeater AP is not connected to the wired LAN, instead it is in the signal range of an AP connected to the wired LAN. Lightweight access points cannot be configured as repeaters since they need to contact a WLC first. The SSID must match on both the root access point and the repeater AP and the recommended coverage overlap between the AP connected to the wired LAN and the repeater AP is 50%.

Because repeaters are also configured on the same channel as the LAN-connected AP, every additional repeater that is added to the chain on the same channel effectively makes the throughput of that network decrease because wireless works in half-duplex mode. If any AP is transmitting, everyone else must wait their turn before they can transmit their message.



Note: A better solution is to use PLC (Power Line Communication) adapter sets (such as HomePlug) which do not need to be in the range of the root AP; the throughput of the root network is not modified by the repeater network as long as they do not overlap.

Lightweight APs → Plug-and-Play devices

2/26/15

When using lightweight access points, the AP and the wireless LAN controller (WLC) share the functions of layer 2, the MAC layer (sometimes referred to as “split” MAC). The management controller includes a Wireless Control System (WCS) and location-tracking appliance. Redundancy consists of multiple WLCs.

The AP handles real-time processes and the WLC handles processes that are not time sensitive.

Examples of LWAP handled functions include:

- All client traffic is sent through LWAPP/CAPWAP tunnel
- Client authentication and association **only in H-REAP mode**
- Client data traffic switching **only in H-REAP mode**
- Wireless 802.11 PHY/MAC

Controllers provide a single point of management which can be a big advantage in large-scale deployments.

Examples of WLC handled functions include:

- LWAP configuration and firmware management
- Client authentication and association
- Client data traffic switching
- L2/L3 Roaming

LWAPP → Replaced by CAPWAP (Control And Provisioning of Wireless Access Points)

LWAPP (Lightweight Access Point Protocol) provides access point discovery, information exchange, and configuration.

L2 LWAPP (*deprecated*) tunnels use EtherType code 0xBBB.

L3 LWAPP (which *should rather be called a L4 LWAPP*) encapsulates control traffic using UDP port 1024 as the source and UDP port 12223 as the destination (port 12222 for data traffic). Layer 3 LWAPP uses an IP/UDP frame that requires the Cisco LWAP to get its IP address from a DHCP server.

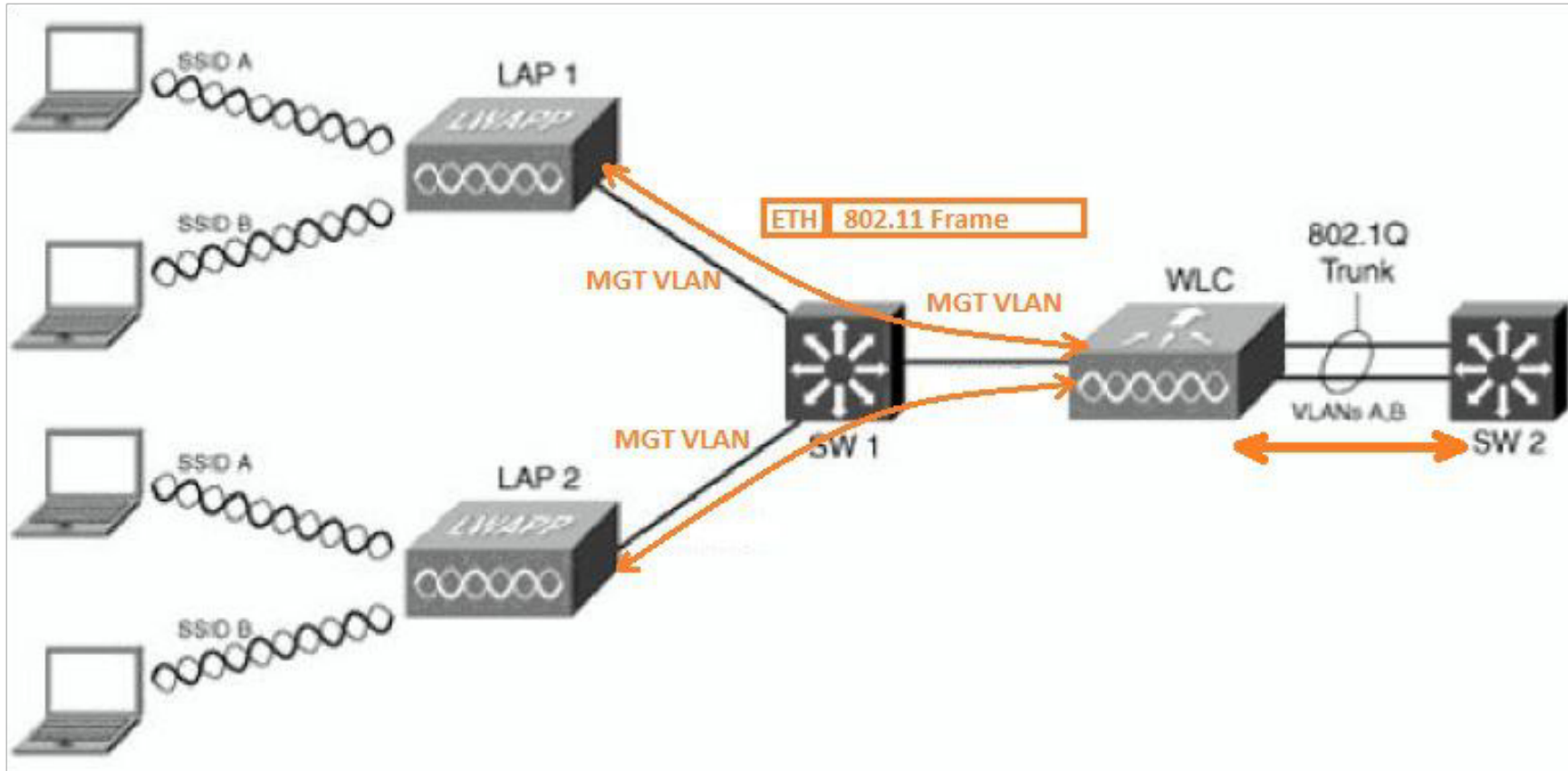
The split MAC function is performed by LWAPP which uses **AES-encrypted** control messages, but does not encrypt data traffic.

- Control traffic for LWAPP is encapsulated and encrypted
- Data traffic for LWAPP is encapsulated but not encrypted

All the client (802.11) packets are encapsulated in a LWAPP packet by the LWAP and sent to the WLC. WLC decapsulates the LWAPP packet and acts based on the destination IP address in the 802.11 packet. If the destination is one of the wireless clients associated to the WLC, it encapsulates the packet again with the LWAPP and sends it to the LWAP of the client, where it is decapsulated and sent to the wireless client. If the destination is on the wired side of the network, it removes the 802.11 header, adds the Ethernet header, and forwards the packet to the connected switch, from where it is sent to the wired client. When a packet comes from the wired side, WLC removes the Ethernet header, adds the 802.11 header, encapsulates it with LWAPP, and sends it to the LWAP, where it is decapsulated, and the 802.11 packet is delivered to the wireless client.

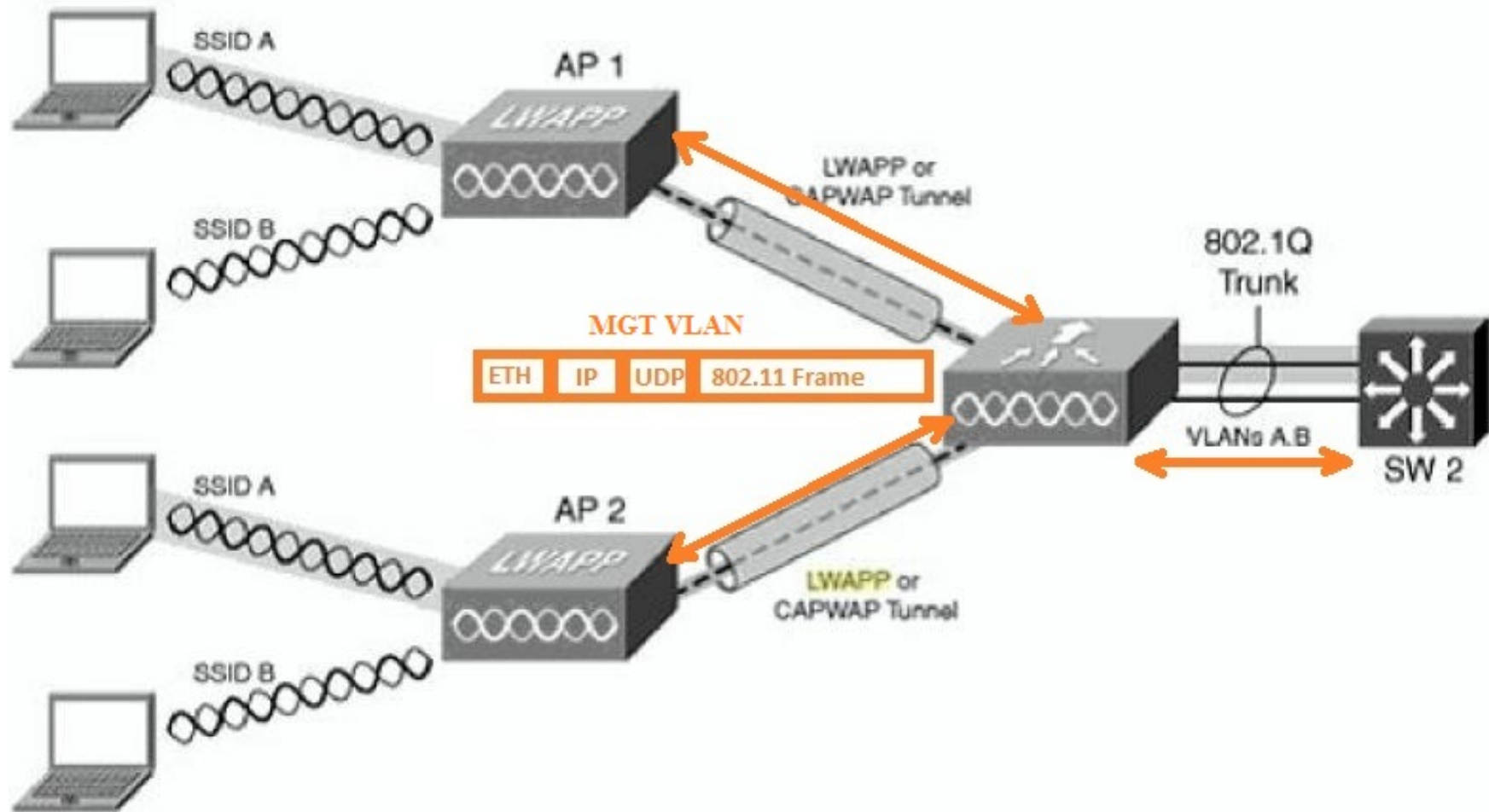
Layer 2 LWAPP/CAPWAP Tunnel is *legacy* mode, implies a flat layer 2 network between the LWAPs and the WLC(s):

- the APs do not require an IP address.
- LWAP receives 802.11 frame and encapsulates into an Ethernet frame towards WLC
- implies LWAP and WLC must be in the same VLAN and subnet.
- WLC places WLAN traffic into appropriate VLANs after having deencapsulated traffic received from LWAPs.



Layer 3 LWAPP/CAPWAP Tunnel is preferred (actually rather a **L4 tunnel**):

- LWAP receives 802.11 frame and encapsulates into Ethernet/IP/UDP towards WLC
- implies WLC can be anywhere as long as it is reachable
- WLC places WLAN traffic into appropriate VLANs after having deencapsulated traffic received from LWAPs



Lightweight APs use this process to discover their controller:

1. **The AP requests a DHCP address** – the response includes the management IP of one or more WLCs using DHCP Option 43.
2. **The AP sends a Discovery Request message** (using L3 CAPWAP or LWAPP) to each WLC.
3. **The WLC responds** (using CAPWAP or LWAPP) with a **Discovery Response** that includes the number of APs associated with it.
4. **The AP sends a Join Request to the chosen WLC** (cf. Selection criteria below) with its own digitally signed X.509 certificate.
5. **The WLC responds with a Join Response message** with its own digitally signed X.509 certificate. Then, the AP and WLC produce encryption keys for future control messages. The WLC then configures the AP (SSID, channels, security settings, etc.).

Step 2 (discovery request) explained (more information [here](#)):

Access points must be discovered by a controller before they can become an active part of the network.

Layer 3 CAPWAP or LWAPP WLC:

- *broadcast discovery* request message on the local IP subnet intended to WLC(s) located on the same local or remote subnet (accessible through *ip-helper*)
- *unicast discovery* request to IP addresses of the WLCs advertised by already registered neighbor LWAP(s) through neighbor messages that are sent over the air (**OTAP** – Over The Air Provisioning); this feature must be supported and enabled.
- *unicast discovery* request to IP addresses of the primary, secondary, and tertiary WLCs previously stored in the AP nonvolatile memory (**priming**)
- *unicast discovery* request to IP addresses learned by DHCP option 43
- *unicast discovery* request to IP addresses learned by DNS resolution of CISCO-CAPWAP-CONTROLLER.localdomain or CISCO-LWAPP-CONTROLLER.localdomain

The selected WLC is based on the following:

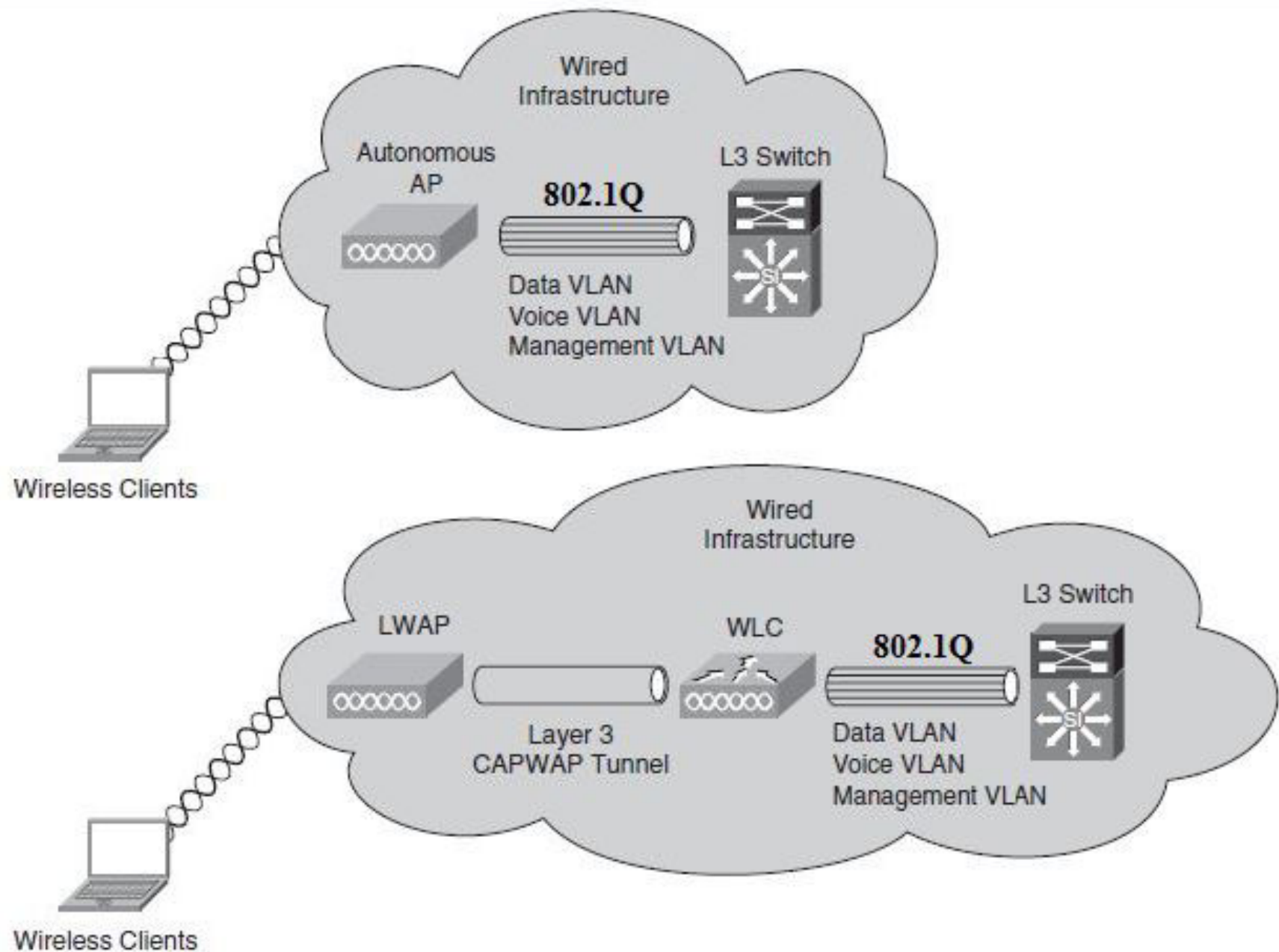
- Previously configured primary, secondary, or tertiary WLCs
- The WLC configured as the master controller
- The WLC with the most excess capacity for AP associations

CAPWAP

A newer IETF-standard that can perform the same function is CAPWAP (Control And Provisioning of Wireless Access Points protocol). Both CAPWAP and LWAPP use UDP and the controller does not have to be in the same subnet as the APs, just reachable through IP.

CAPWAP is similar to LWAPP except for the following differences:

- CAPWAP uses **DTLS** (Datagram Transport Layer Security) for authentication and encryption to protect control and optionally data traffic between APs and controllers. LWAPP uses AES.
- CAPWAP has a dynamic maximum transmission unit (MTU) discovery mechanism.
- LWAP CAPWAP control and data messages use an ephemeral UDP source port that is derived from a hash between the LWAP MAC addresses.
- LWAP CAPWAP control messages run over UDP destination port 5246.
- LWAP CAPWAP data messages use UDP destination port 5247.
- LWAP starts the discovery process to find a WLC with a CAPWAP request. After 1 mn without a response, the LWAP starts a LWAPP discovery process.
- WLC selection by a CAPWAP LWAP is a design decision. This is configurable in the WLC.



Autonomous AP Versus CAPWAP LWAP with WLC

Lightweight AP Planning

When using lightweight access points, the traffic flows from the client to the AP through the switched network, to the WLC, and finally from there to its destination. Because the traffic always goes from AP to the controller, it is important that the AP has layer 2 or 3 connectivity to the WLC.

While the controllers can be distributed across the network (ex. a single controller in each building), Cisco recommends a centralized approach co-locating them for example together in your data center. Simplified management and user mobility are the reasons.

VLAN and SSID assignments must be configured on the controller in a controller-based environment as opposed to the autonomous model.

A management VLAN is used to communicate between the AP and controller. The interface on the switch connected to the AP should be an access port using the management VLAN ID. The interface on the switch 'upstreamly' connected to the controller should be a trunk to forward traffic for multiple subnets.

Etherchannels (portchannels) are often used to connect WLCs to the upstream switch for redundancy and bandwidth.

When using LWAPP on a lightweight AP, the console port provides read-only access to the device.

As with the autonomous model, you should make sure the AP has power from either PoE or a power injector.

WLC Configuration

WLCs can be configured by command-line or through a web browser and GUI interface. There are two commands that enable the web interface modes on the controller:

To enable HTTP access

```
config network webmode {enable | disable}
```

To enable HTTPS access

```
config network secureweb {enable | disable}
```

Note: Cisco WLAN controllers can be a dedicated appliance, module for 6500 series switches and 7600 routers, or integrated into 3750G switches. Also, while we aren't going to get into configuring an AP, you should be aware that the virtual interface on a WLC is often used for a DHCP relay.

Hybrid Remote Edge Access Point (H-REAP) aka FlexConnect

If the wireless controllers are located across the WAN, some significant problems can result. The traffic would have to travel over the WAN to the controller and back again. Also, if the WAN link goes down or flaps, the APs quickly lose functionality. H-REAP is designed to address these problems.

Connected mode

- **central authentication, central switching (Split-MAC)**—In this state, the controller handles client authentication, and all client data is tunneled back to the controller.
- **central authentication, local switching (Local MAC)**—In this state, the controller handles client authentication, and the FlexConnect AP switches data packets locally. That prevents local traffic from having to cross the WAN. Also, it doesn't have to be local traffic – you can configure any VLAN on any AP you want to stay off the controller, but local VLANs make the most sense. The AP sends only control and remote traffic to the controller.
- **local authentication, local switching (Local MAC)**—In this state, the FlexConnect AP handles client authentication and switches client data packets locally. The AP can be used as a EAP server.

Note: In this mode, the connection between the AP and the switch should be a trunk to carry all the VLANs.

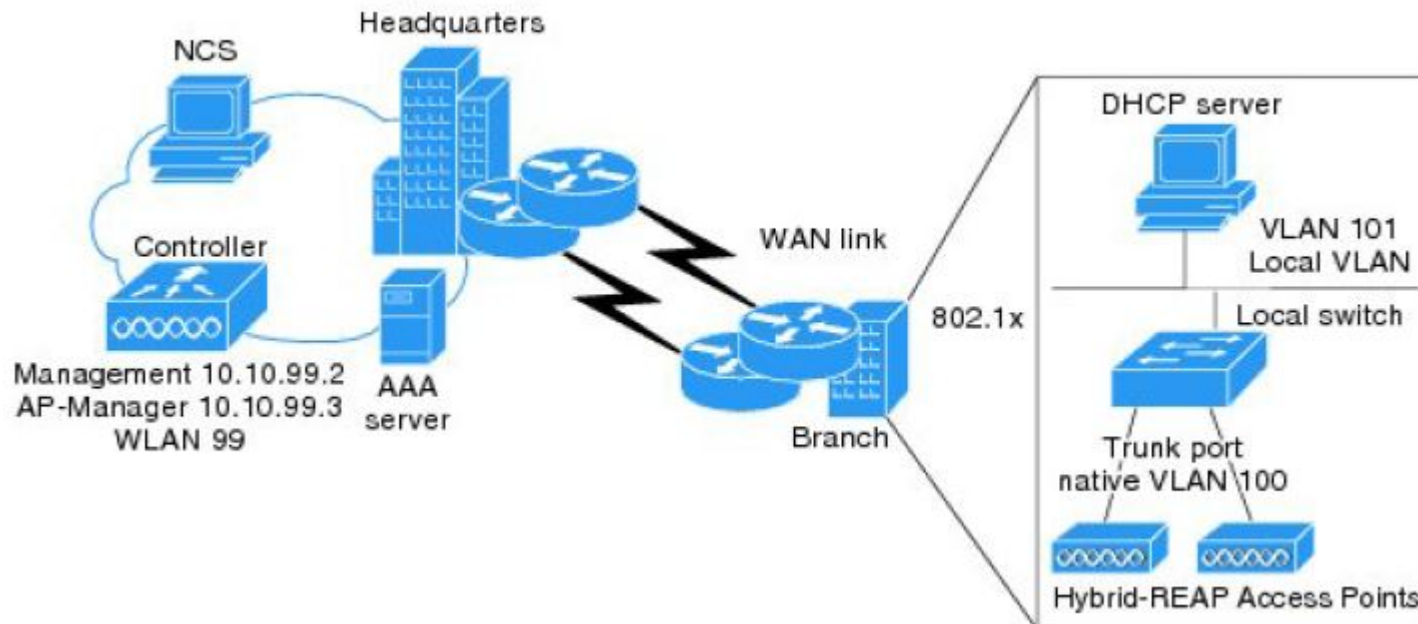
Standalone (disconnected) mode

- **local authentication, local switching (Local MAC)**—In this state, the FlexConnect AP handles client authentication and switches client data packets locally. The AP can be used as a EAP server.

Since the controller becomes unreachable, local traffic is still sent to the local switch, but remote destinations won't be reachable as the WAN would be down.

Note: FlexConnect (H-REAP) is configured on the controllers, not the APs.

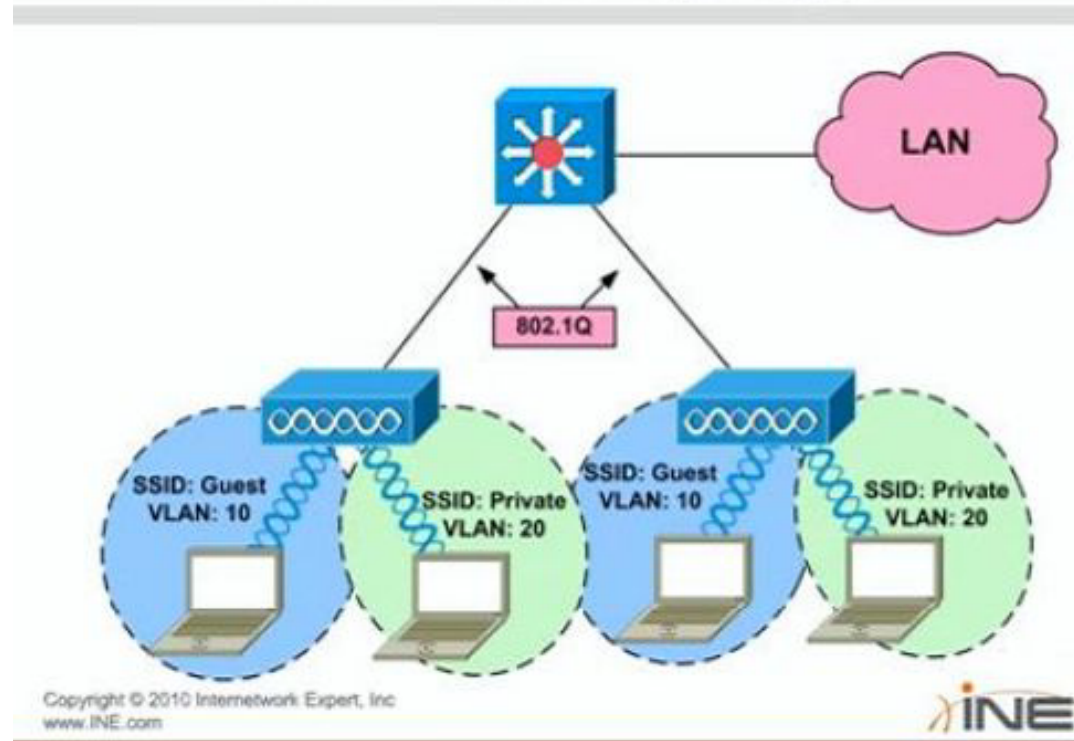
cf. WLC design for more details about REAP and H-REAP modes



Switch Configuration for Wireless

For an autonomous AP, configure it as an access port for a single VLAN or a trunk port for multiple VLANs. Trust CoS if the link is a trunk, DSCP if it is an access port. Set the trunk native VLAN to the AP management VLAN. Prioritize voice if you are using wireless VoIP phones.

WLAN VLAN Topology



For a controller-based AP, generally use an access port (except for FlexConnect) and place it in the management VLAN, even if there are multiple SSIDs per LWAP. Trust DSCP and again prioritize voice if you are using wireless VoIP phones.

Configure the upstream switch port connected to the controller as a trunk port (limited to only wireless and management VLANs). Trust CoS on the port and again prioritize voice if you are using wireless VoIP phones.

Cisco
642
813

Chapter 10

VoIP

& QoS

Voice over IP (VoIP) is becoming more and more common in the enterprise world by replacing traditional TDM phone systems with feature-rich IP-based communication servers.

Some benefits of converged voice, video, and data into a single network include:

- Expense reducer – if only a single cable drop is required per user, cabling and network provisioning costs go down. PSTN costs also go down as more calls can use the existing data network and not the public phone service.
- Efficiencies in bandwidth – for example, if a voice call is not in progress, data can be transmitted on the same link. That's not the case with traditional phone lines.
- Innovative features- VoIP allows new services to be added including unifying several modes of communication (ex. voicemail, email, IM). Service providers can also sell new services and provide more flexible pricing arrangements.

Features/Network Requirements	VOIP	Video	Data
Bandwidth	Low	High	High
Delay	Low	Low	Not crucial
Jitter	Low	Low	Not crucial
Packet loss	Low	Low	Not crucial
Security	Medium	Medium	High
Management	High	Medium	Medium
Availability	High	High	High
PoE	✓	Sometimes	Sometimes

AVVID

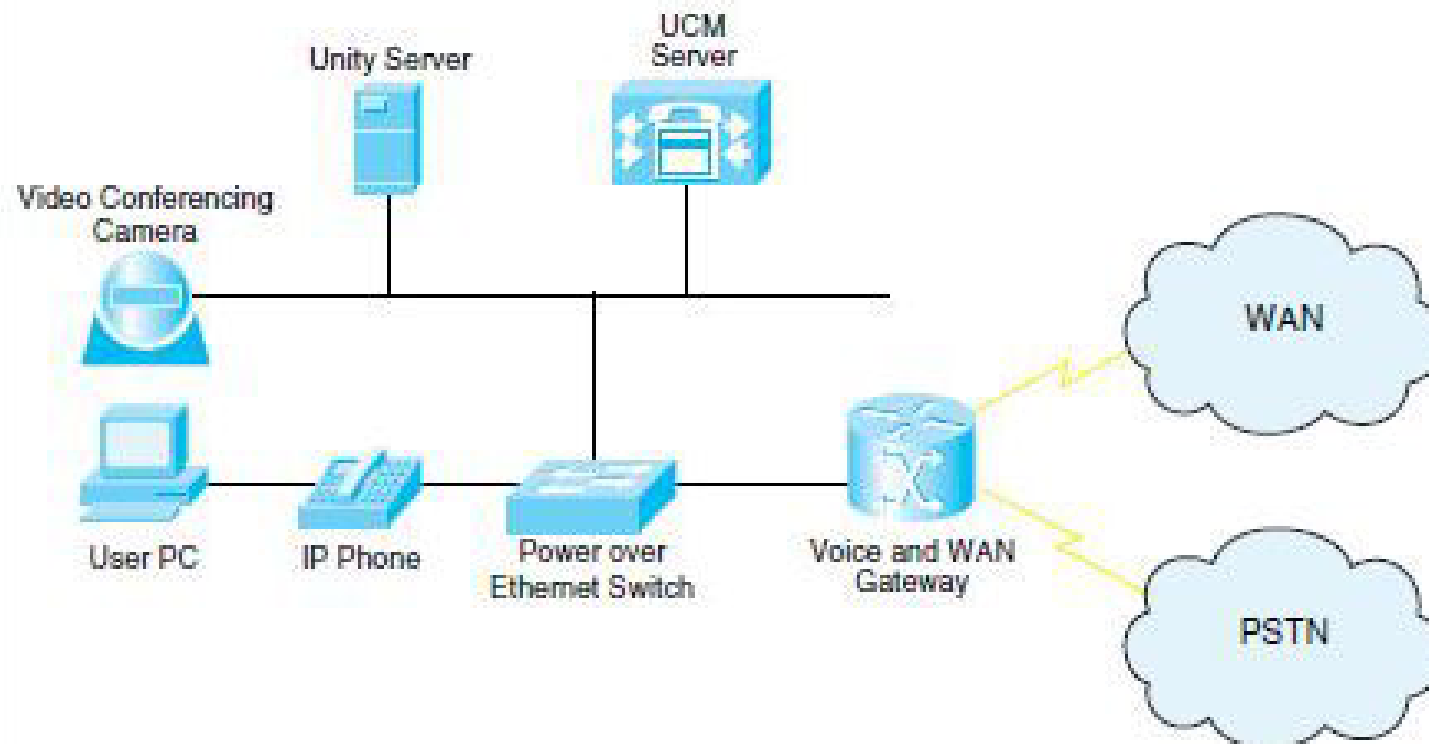
Architecture for voice, video and integrated data, more commonly referred to by Cisco as AVVID, was an all-encompassing blueprint for converged enterprise networks pitched by Cisco. While it was originally intended to include a very large cross-section of product families, it has been primarily focused on Cisco's VoIP products. AVVID addresses:

- High availability
- QoS
- Security
- Mobility
- Scalability

VoIP Components

- IP Phones – Provides voice and applications to users
- Cisco Unified Communications Manager (UCM) – Essentially an IP PBX
- Voice Gateways – Translate between IP and PSTN
- Gatekeepers - Optional, usually in larger environments. Performs call admission control, allocates bandwidth for calls, and resolves phone numbers to IP addresses
- Video Conferencing Units – Allow voice/video calls
- Multipoint Control Units - Allow multi-point audio and videoconferencing
- Application Servers – Provide application services like Unity Voicemail

Note: Voice traffic comes in two types, call control signaling and voice bearer. The call control portion can use several different protocols to communicate between the phone and UCM and UCM to voice gateway while the voice bearer traffic uses RTP (Real Time Protocol) over UDP.



VoIP Network Requirements

When planning for a VoIP deployment, keep in mind the following factors:

Features like call security, QoS, delay, etc.

Cabling uses at least CAT-5.

Power either **PoE** (aka **power inline**) from the switch or power brick connected to the phone itself.

Bandwidth planning is crucial. Commit no more than 25% capacity to allow for over-subscription and other types of traffic like video, and data. A good rule of thumb is to reserve 24 Kbps of IP network bandwidth per call for 8 Kbps (G.729-like) compressed voice. If G.711 is used, then reserve 80 Kbps of bandwidth.

Network management is important for proactively managing bandwidth and availability.

High availability means redundant links, a FHRP, such as HSRP, VRRP and GLBP, an auto-restart UPS, monitoring, and a response contract.

Call Signaling

There are generally two separate traffic streams when placing a VoIP call. The first is the call control signaling, used to setup, maintain, redirect and tear-down calls. Some examples of **call signaling protocols include H.323, SIP, and MGCP**. Make sure you do not confuse these protocols with the voice compression protocols like G.729 and G.711.

The second is the actual UDP voice traffic itself, which uses RTP (Real-Time Transport Protocol) to encapsulate the traffic.

Bandwidth Considerations

Each call uses somewhere around 21-106 kbs depending on the **CODEC** and **compression technique (G.711, G.722, G.723.1, G.729, proprietary)** used, plus around 150 bps for control traffic. Each voice payload is in the neighborhood of 20-160 bytes.

A good formula for calculating call bandwidth is: (Packet payload + all headers) * Packets per second

- Max one-way delay of 150 ms
- Max average jitter (variable delays) of 30 ms
- Under 1% packet loss
- The sum of every application's bandwidth (voice, video and data) should not exceed 75% of the total available bandwidth for each link. Some bandwidth is necessary for overhead traffic, like Layer 2 control protocols and Layer 3 routing protocol updates.

Voice VLANs (sometimes referred to as **auxiliary VLANs**) are a way for Cisco switches to dynamically tag and assign voice traffic including placing it in its own separate VLAN/subnet. That allows for QoS and security to be applied as well as simplified troubleshooting. Voice VLANs are disabled by default.

Cisco IP phones have a small internal switch that places an 802.1Q tag on the voice traffic and marks the Class of Service (CoS) bits in the tag along with the VLAN ID. Data traffic (from the attached PC) is sent over the access VLAN, while all voice traffic is sent over the voice VLAN on the access port.

Cisco calls this setup a **multi-VLAN access port**:

```
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan id          -----> DATA
Switch(config-if)# switchport voice vlan id          -----> VOICE
Switch(config-if)# switchport voice vlan dot1p       -----> 802.1P priority tagging and VLAN 0.
Switch(config-if)# switchport voice vlan untagged    -----> The telephone sends untagged voice traffic
Switch(config-if)# switchport voice vlan none       -----> Do not instruct the IP telephone about the voice VLAN
```

Note: it is technically possible, although not recommended, to place data and voice in the same VLAN on the same access port.

This whole process of enabling voice VLANs also enables the switch to forward frames with specific 802.1P markings.

802.1P designates how QoS is applied at the MAC layer.

The QoS technique developed by the IEEE 802.1P working group, also known as **Class of Service (CoS)**, is a 3-bit field called the **Priority Code Point (PCP)** within an Ethernet frame header *when using VLAN tagged frames as defined by IEEE 802.1Q or ISL*. It specifies a priority value of between 0 and 7 inclusive that can be used by QoS disciplines to differentiate traffic.

Default Voice VLAN Configuration

- The voice VLAN feature is disabled.
- When the voice VLAN feature is enabled, all untagged traffic is sent according to the default CoS priority of the port.
- The CoS value is not trusted for 802.1P or 802.1Q tagged traffic and the default CoS value is 0 for incoming traffic.
- The IP Phone overrides the priority of all incoming traffic (tagged and untagged) and sets the CoS value to 0.

Voice VLAN Configuration Guidelines

- You can configure Voice VLAN on switch access ports, Dynamic access port (VMPS), Secured port, 802.1X authenticated port or Protected port.
- You *cannot* configure Voice VLAN on *Trunks* (even though the configuration is allowed) *NOR on PVLAN ports*.
- The PortFast feature is automatically enabled when voice VLAN is configured. When you disable voice VLAN, the PortFast feature is not automatically disabled.
- When you enable port security on an interface that is also configured with a voice VLAN, you must set the maximum allowed secure addresses on the port to at least two.
- If any type of port security is enabled on the access port, dynamic port security is automatically enabled on the voice VLAN. You cannot configure port security on a per-VLAN basis.
- You cannot configure static secure or sticky secure MAC addresses on a voice VLAN.

Power over Ethernet

PoE Switches

Two different power standards exist for PoE, **Cisco Inline PoE and IEEE 802.3af**. Both have a mechanism to sense that a powered device is connected to a port. 802.3af relies on the devices to let the switch know how much power it needs, while Cisco's devices can additionally use CDP to send that information. Most phones don't require more than 15 Watts of power, but some PoE equipments still require more. The new **802.3at standard, also known as PoE+**, will specify up to 30 Watts of power. Some current Cisco switches can supply up to 20W.

Switch assumes all PoE devices require 15.4 W of power until the device tells the switch otherwise. If the switch reboots, all PoE devices will receive 15.4 Watts at the same time, which is why you should budget 15.4 W for every PoE device when doing power planning.

Notes: Non-CDP devices always get 15.4 W allocated to them. The Cisco Aironet 1250 Series AP needs a switch that provides 20W and a Gigabit interface.

PoE Configuration

Cisco switches automatically detect and provide power, but if you need to disable it or re-enable it, use the following commands:

```
Switch(config-if)# power inline {never | {{auto | static} [max max-wattage]}}
```

// **never**: Disable device detection, and disable power to the port.
// **auto**: Enable powered-device detection. If enough power is available, automatically allocate power to the PoE port after device detection.
// **static**: Enable powered-device detection. Pre-allocate (reserve) power for a port before the switch discovers the powered device.
// **max max-wattage**: Limit the power allowed on the port. The range is 4000 to 20000 milliwatts. If no value is specified, the maximum is allowed.

To view power information for all ports:

```
Switch# show power inline [interface]
```

Video

Video traffic, from Cisco's perspective, falls into one of three categories:

Many to many

- Examples include Telepresence, WebEx, peer-to-peer video apps
- Data flows client-to-client or MCU-to-client
- Bandwidth requirements for high-def video can be up to 12 Mbps per location (with compression)

Many to few

- Examples include IP surveillance cameras.
- Typically require up to 4 Mbps of bandwidth

Few to many

- Example is Internet streaming from a single source
- Quality not as critical
- Traffic flows from storage to client or server to client

Quality of Service

Quality of Service is a very important part of operating a VoIP platform on a campus network. The ability to prioritize different traffic on the same link makes voice over IP a reality on a shared Ethernet fabric. There are three main drivers for applying QoS: delay, jitter and packet loss.

QoS Inbound Strategies

Classification

Distinguishes one type of traffic from another by:

- **Existing marking:**
 - + layer 2: Ethernet CoS, ATM, MPLS
 - + layer 3: IP precedence or DSCP
- **ACLs** (Access Control Lists): Classifying Flows (*Protocols and UDP/TCP ports*) for Congestion Management Tools
- **Class-Maps**: Classifying Flows (*Layers 3 <-> 7*) for applying policies.
- **PBR** (Policy-Based Routing, using route-maps): Classifying Flows (*Protocols and UDP/TCP ports*), Setting QoS Policy and Routing to Specific Paths
- **NBAR** (Network-Based Application Recognition): Granular Dynamic Identification of Flows with *Deep Packet Inspection* and Protocol Discovery Capability
 - It can recognize applications even when they do not use standard network ports. Furthermore, it matches fields at the application layer.
 - Before NBAR, classification was limited to Layer 4 UDP/TCP port numbers.
- **CAR** (Committed access rate): Uses an ACL to set precedence

Marking

- **layer 2**: Within a frame, placing an **802.1P CoS** (Class Of Service) value within the 802.1Q trunk tag or ISL User field.
- **layer 3: IP Precedence** (first 3 high order bits of ToS byte) or **Differentiated Services Code Point (DSCP)** - first 6 high order bits of DS byte) values in a packet's IP header.

Cos and IPP/DSCP should be synced.

For ex., COS=5 => IPP=5 (voice)

Policing

Decides whether a specific type of traffic is within *predefined bandwidth levels*. If not it is usually dropped:

- **CAR** (Committed Access Rate): Setting QoS Policy for Traffic Exceeding some Bandwidth
- **Class-Based Policing**: Limiting the input transmission rate of a class of traffic based on user-defined criteria..

QoS Outbound Strategies

Congestion Management: Queuing

After traffic has been classified and marked, it can be placed into one of many queues to be sent at different rates and order:

- **FIFO** (First-in, first-out Queuing): Basic Store-and-Forward Capability (default)
- **PQ** (Priority Queuing): **Prioritizing Traffic** with four output queues (high, medium, default, and low priority queues). Higher-priority queues must be emptied before lower-priority queues. PQs drop any oversubscription during times of congestion
- **CQ** (Custom Queuing) (*legacy*): **Garanteeing Bandwidth** with up to 16 individual output queues. Byte size limits are assigned to each queue so that when the limit is reached, it proceeds to the next queue. CQ is fairer than PQ because it allows some level of service to all traffic.
- **WFQ** ((Flow-based) Weighted Fair Queuing): **Creating Fairness Among Flows**; Low-bandwidth traffic has priority over high-bandwidth traffic. High-bandwidth traffic shares the service according to assigned weight values. WFQ is the default QoS mechanism on interfaces below 2.0 Mbps.
- **CBWFQ** (Class-Based Weighted Fair Queuing): **Ensuring Minimum Network Bandwidth**; Traffic that matches the class criteria (ACLs, protocols, and input interfaces.) belongs to that specific class. Each class has a defined queue that corresponds to an output interface. After traffic has been matched and belongs to a specific class, you can modify its characteristics, such as assigning bandwidth, maximum queue limit, and weight. During periods of congestion, the bandwidth assigned to the class is the guaranteed bandwidth that is delivered to the class.
- **LLQ** (Low-Latency Queuing): **Garanteeing real-time traffic** to be serviced first; it adds a strict priority queue (PQ) to CBWFQ. The PQ is sent before other queues are serviced. That gives voice preferential treatment over the other traffic types. Unlike priority queuing, LLQ provides for a maximum threshold on the PQ to prevent lower priority traffic from being starved by the PQ. LLQ is the standard QoS method for many VoIP networks. It is recommended that you place only voice in the PQ. Introducing video or other variable-rate, real-time or non-real-time traffic types could cause unacceptable jitter for the voice traffic.

Congestion Avoidance: TCP Traffic on Non Priority AF and DF Queues

By default, interface queues accept all traffic until they are full and drop everything after that. So, a mechanism is necessary to do two things:

- 1) Try to make sure that the queue does not fill up, so that there is room for high-priority packets
 - 2) Allow some sort of criteria for dropping packets that are of lower priority before dropping higher-priority packets
- **WRED** (Weighted Random Early Detection).
 - **DWRED** (Distributed WRED)
 - **WTD** (Weighted Tail Drop)
 - **DBL** (Dynamic Buffer Limiting)

Traffic Shaping

Defines an artificial maximum throughput for the interface, providing a steady stream that is throttled while congestion mitigation occurs by buffering traffic. In enterprise environments, traffic shaping is used to smooth the flow of traffic going out to the provider. In provider networks, it prevents the provider from dropping traffic that exceeds the contracted rate.

- **GTS** (Generic Traffic Shaping),
- **Class-Based Policing**: Limiting the output transmission rate of a class of traffic based on user-defined criteria..

CoS and DSCP

2/26/15

Differentiated services provides a mechanism to change levels of service based on the value of specific bits in the IP header or the 802.1Q tag. Each hop along the way must be configured to treat the marked traffic the way you want, also known as per-hop behavior (PHB).

As mentioned, there are two ways to mark the QoS values depending on what layer you are marking it at. The first method (layer 2) uses the three 802.1P bits within the 802.1Q tag to set the CoS value. Voice is commonly set to 5 and video 4.

For layer 3, the 8 bit ToS (DS) field within the IP header is used. There are again two options here. IP Precedence can be set using the top 3 bits or DSCP can be set using the top 6 bits. The bottom 2 bits are used for congestion notification. When setting DSCP values, 0 is the default, indicating best-effort delivery.

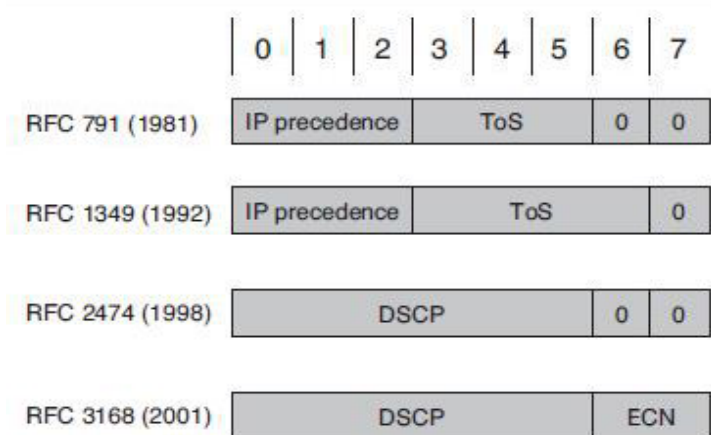
The six bit DSCP code consists of two parts, the first 3 bits define the **DiffServ Assured Forwarding (AF) class** – which matches exactly the *IP precedence* – and the next two bits define the **drop probability**: the higher, the more likely the packet will be dropped by WRED. The sixth bit is unused.

DSCP Name	Binary	Decimal	IP Precedence
CS0	000 000	0	0
CS1	001 000	8	1
AF11	001 010	10	1
AF12	001 100	12	1
AF13	001 110	14	1
CS2	010 000	16	2
AF21	010 010	18	2
AF22	010 100	20	2
AF23	010 110	22	2
CS3	011 000	24	3
AF31	011 010	26	3
AF32	011 100	28	3
AF33	011 110	30	3
CS4	100 000	32	4
AF41	100 010	34	4
AF42	100 100	36	4
AF43	100 110	38	4
CS5	101 000	40	5
EF	101 110	46	5
CS6	110 000	48	6

AF43: $4 \times 8 + 3 \times 2 = 38$

AF53: $5 \times 8 + 3 \times 2 = 46 = \underline{EF}$

IP Precedence = Class Number



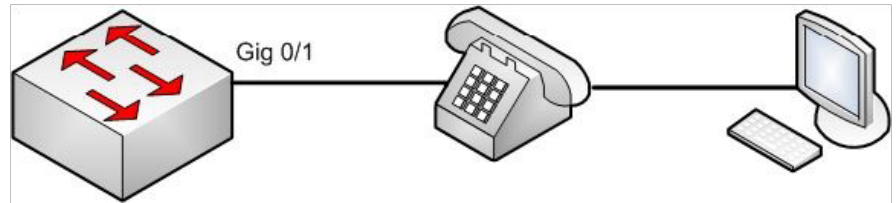
Traffic Type	Layer 2 CoS	Layer 3 IP Precedence	Layer 3 DSCP
Voice RTP	5	5	EF = 46
Voice control	3	3	CS3 = 24
Video conference	4	4	AF41 = 34
Streaming video	4	4	CS4 = 32
Transactional data	2	2	AF21 = 18
Scavenger	1	1	CS1 = 8
Routing Protocols	6	6	CS6 = 48
STP BPDUs	7	7	CS7 = 56

Cisco Recommendations for QoS Marking Values SRND 10.x

Trust Boundaries

The place where a decision about priority marking on incoming frames/packets is done is called the trust boundary. When IP traffic comes into an interface and is already marked, the switch has the following options:

- Trust the DSCP value
- Trust the IP Precedence value
- Trust the CoS value in the frame, on trunk ports
- Classify the traffic based on an IP ACL or MAC ACL, PBR or NBAR.



Cisco recommends marking the traffic as close to the source as possible. IP phones can mark their own traffic and other clients can be marked at the access switch. If that is not an option - mark at the distribution layer, but *never at the core*. Marking slows traffic down, so it has no place being in the core. All devices within the network path should be configured to trust the marking and provide service based on that. Untrusted ports have their marking reset to zero => Best Effort.

QoS for VoIP

Before rolling out VoIP in your environment, think through the following planning steps:

1. **PoE** - Ensure there is enough power for all the phones and there's a UPS backup
2. **Voice VLAN** - Think through the number of VLANs/subnets required, add DHCP scopes for the phones, add voice networks to routing protocols
3. **QoS** - Decide on which marking and queues you plan on using. Cisco recommends implementing AutoQoS and then tuning as needed.
4. **Fast Convergence** - tune RSTP, routing protocols and HSRP/VRRP/GLBP timers
5. **Test Plan** - Test the implementation before rolling it out to real users. Some things to look for include making sure the phone and PC have the correct IP addresses, the phone registers itself, and calls can be made.

QoS Best Practices

QoS policies should be designed from the high-level requirements gathered in the project planning stage. Here are some best practices to keep in mind when designing your QoS policies:

- **Perform QoS in hardware** where there is no performance impact. Cisco routers perform QoS in software. Cisco switches perform QoS in hardware ASICs. If you must perform QoS in software, ensure your devices are sized appropriately and don't become a bottleneck.
- When possible, follow industry standards when deploying QoS policies to align with service providers and business partners.
- **Mark packets as close to the source as possible**
- Enable queuing wherever congestion may occur to guarantee service optimum performance to critical applications. This includes data center and campus LANs as well as public (DMVPN) and private (MPLS) WANs.
- Provide at least 25% bandwidth for Best Effort traffic in the class-default.
- **For real-time traffic such as voice and video, use PQs and be sure to assign ample bandwidth.** However, attempt to limit the priority queues to p % of the overall available bandwidth to **prevent starving other traffic.**
- If multiple applications use PQ, use discrete priority queues for each applications. For example, **VOIP gets its own unique queue, video gets its own queue**, etc.
- **WRED is applied only on AF and DF queues**, which makes CS classes priority higher than AF classes priority.
- **WRED is not enabled on EF and CS queues.** It is not required on scavenger queue as its bandwidth is already set at the minimum.
- Optional: **Tune WRED thresholds consistently**—for example:
 - Set the minimum WRED thresholds for AFx3 to 60% of the queue depth
 - Set the minimum WRED thresholds for AFx2 to 70% of the queue depth
 - Set the minimum WRED thresholds for AFx1 to 80% of the queue depth
 - Set all maximum WRED thresholds to 100%
- Deploy **Link Efficiencies Mechanisms on WANs less than or equal to 768Kbps.**
- **Overprovision video over IP queues by 20% to accommodate bursts**

Auto QoS

Auto QoS, when enabled, configures the switch interfaces using common best-practices including:

- Trust boundaries
- Auto discovery and classification of network applications
- Creates Marking and Queueing policies for those apps
- Configures the switch to support IP phones
- Sets up SNMP traps for network reporting
- Provides a consistent QoS configuration across the environment

Note: Auto QoS uses CDP to function properly with IP phones, so make sure it is not disabled.

Configuring Auto QoS

Automatically configures QoS and trusts the QoS markings of incoming packets

```
Switch(config-if)# auto qos trust {cos | dscp}
```

Automatically configures QoS for VoIP

```
Switch(config-if)# auto qos voip [trust]
// Without "trust": Configures QoS for VoIP and classifies traffic with NBAR
// With "trust": Trust CoS for non routed ports and DSCP for routed ports on incoming traffic
```

Automaticvally configures QoS for VoIP and trusts the QoS markings only if a Cisco phone or Cisco softphone is connected (requires CDP)

```
Switch(config-if)# auto qos voip {cisco-phone | cisco-softphone}
// cisco-phone: 7960 on non-routed ports and CoS is trusted
// cisco-softphone: Cisco SoftPhone on routed ports and DSCP is trusted
```

Automatically configures QoS for NOT trusted interfaces

```
Switch(config-if)# auto qos classify
```

Verifying Auto QoS

Displays the Auto QoS configuration

```
Switch# show auto qos
```

Configuring Manual QoS

By default, QoS is disabled. There is no concept of trusted or untrusted ports because the packets are not modified (the CoS, DSCP, and IP precedence values in the packet are not changed). Traffic is switched in pass-through mode (packets are switched without any rewrites and classified as best effort without any policing).

Enabling manual QoS

All other QoS settings are set to their defaults, traffic is classified as best effort (the DSCP and CoS values are set to 0) without any policing. No policy maps are configured. The default port trust state on all ports is untrusted. The default ingress and egress queue settings are in effect.

```
Switch(config)# mls qos
```

Trusting QoS markings

Effectively moves the trust boundary to the attached device (often an IP phone or server).

Configures the interface to classify incoming packets by using their CoS/DSCP values. For untagged packets, the port default COS value is used.

```
Switch(config-if)# mls qos trust {cos | dscp}
```

Trusting previous QoS configuration only if a Cisco phone is connected

```
Switch(config-if)# mls qos trust device cisco-phone
```

Note: CDP must be configured to detect whether or not a Cisco IP phone is attached to the port.

Defining default CoS value for a port

Used to derive the internal DSCP value

```
Switch(config-if)# mls qos cos value
```

Defining a CoS to DSCP map:

```
Switch(config)# mls qos map cos-dscp {dscp1...dscp8}
// For dscp1...dscp8, enter eight DSCP values that correspond to CoS values 0 to 7.
// Separate each DSCP value with a space. The range is 0 to 62.
```

Defining a DSCP to Cos map:

```
Switch(config)# mls qos map dscp-cos {dscp-list to cos}
// For dscp-list, enter up to eight DSCP values , each separated by a space. The range is 0 to 62.
// For cos, enter a single CoS value to which the DSCP values correspond. The range is 0 to 7.
// One command per cos value.
```

<i>DSCP-to-CoS Default Map</i>								
DSCP	0-7	8-15	16-23	24-31	32-39	40-47	48-55	56-63
CoS	0	1	2	3	4	5	6	7

Moving the trust boundary to the phone

Instructs the IP phone port to set/overwrite CoS value for data coming from a PC attached to the phone. The phone would then be the new trust boundary because it is now doing the marking on the data traffic.

```
Switch(config-if)# switchport priority extend cos cos-value
```

Instructing the phone to trust the CoS priority of the data coming from the attached PC

```
Switch(config-if)# switchport priority extend trust
```

Verifying Manual QoS

Verifying interface parameters

```
Switch# show interfaces interface-id switchport
```

Verifying QoS parameters on an interface

```
Switch# show mls qos [{vlan vlan-id} | {interface interface-id} | {port-channel pc-id}]
```

QoS for Video

Video traffic can change dramatically depending on what kind of compression is used and how static the picture is. Video that is constantly changing will use much more bandwidth and be more bursty than fairly still-image video. Voice traffic is much more steady.

Video should be placed in its own queue, especially if the organization is doing interactive video. Consider creating separate queues for interactive and streaming video if the business uses it. Less than 200 ms of latency is considered acceptable by most standards.

Configuring Auto QoS

Automatically configures QoS for Video and trusts the QoS markings only if a Cisco TelePresence System or Cisco IP camera is connected

```
Switch(config-if)# auto qos video {cts | ip-camera}
```

QOS Commands Summary

	Configuration *	Verification *
Auto QOS	1) 2) #auto qos trust {cos dscp} #auto qos voip [{trust cisco-phone cisco-software-phone}] #auto qos video {cts ip-camera} #auto qos classify	1) #show auto qos 2) #show auto qos [interface]
Manual QOS	1)#mls qos 2)#mls qos trust {cos dscp { device cisco-phone}} #mls qos cos #switchport priority extend {cos trust}	1)#show mls qos 2)#show mls qos [vlan interface port-channel] #show interface switchport

*: 1) Globally
2) Interface

Cisco
642
813

Chapter 11

Syslog, CDP SNMP, RMON & NetFlow

Syslog

2/26/15

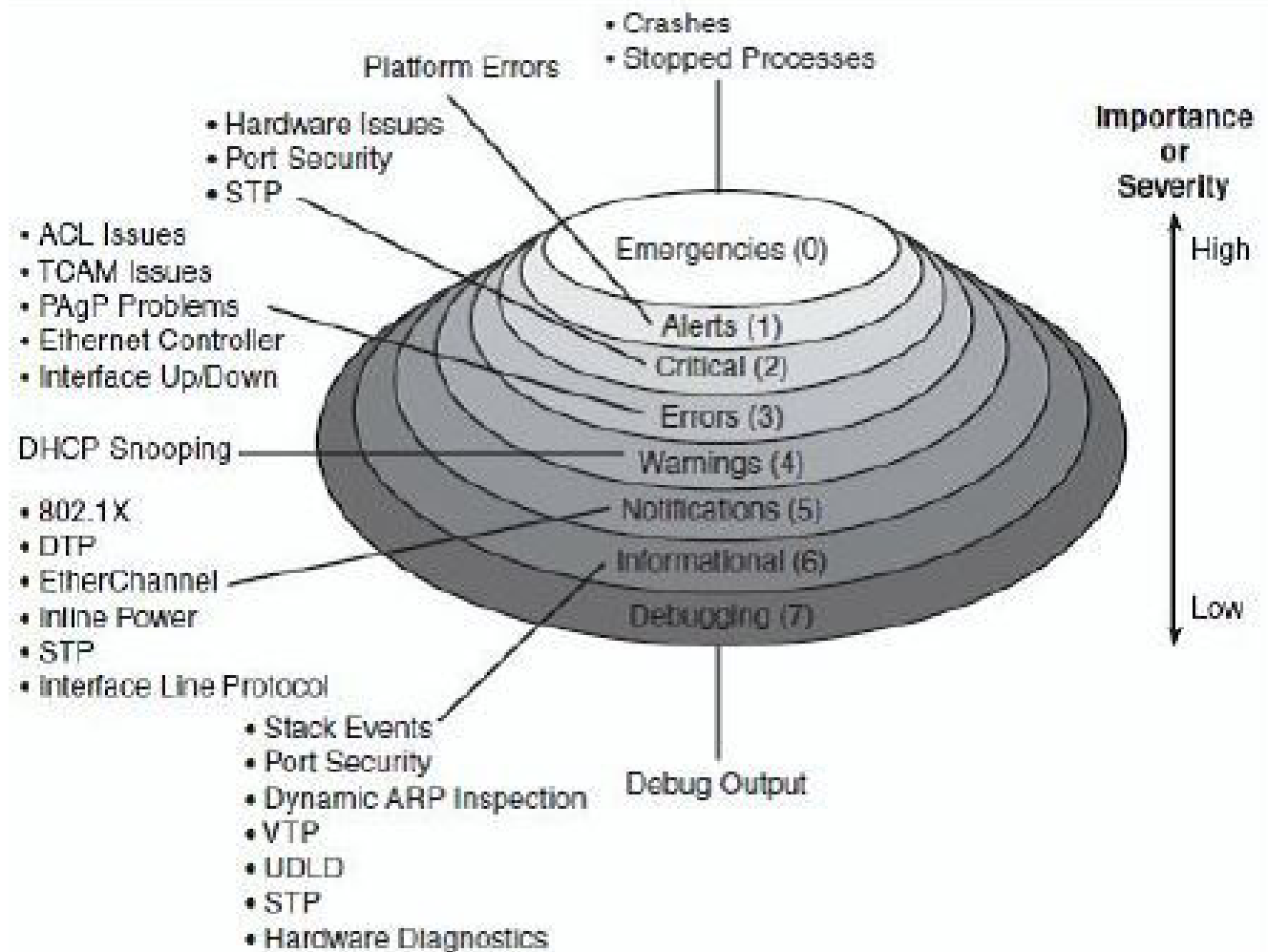
Syslog is a network management protocol that is not unique to Cisco devices, but integrates well within IOS. Syslog allows a network-attached device to report and log error and notification messages either locally or to a remote Syslog server.

Syslog messages are plain text sent using **UDP port 514**.

Every syslog message contains two parts, a facility and a severity level. The severity level goes from 0 to 7 with 0 being the most severe to 7 being simply informational.

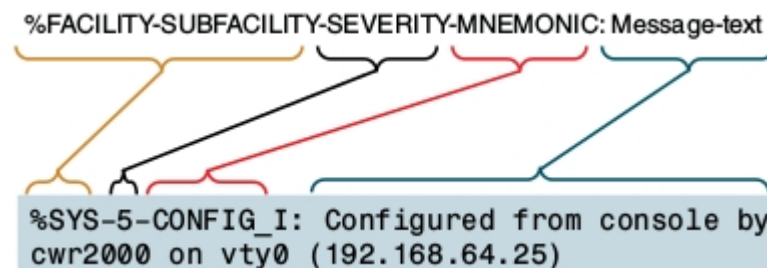
Syslog severity (highest to lowest):

- 0. Emergency** (highest)
- 1. Alert**
- 2. Critical**
- 3. Error**
- 4. Warning**
- 5. Notification**
- 6. Informational**
- 7. Debugging** (lowest)



Message Format

- **Facility:** A code consisting of two or more uppercase letters that indicate the hardware device, protocol, or a module of the system software. There are more than **500** of them.
- **Severity:** A single-digit code from 0 to 7 that reflects the severity of the condition. The lower the number, the more serious the situation.
- **Mnemonic:** A code that uniquely identifies the error message.
- **Message-text:** A text string describing the condition. This portion of the message sometimes contains detailed information about the event, including terminal port numbers, network addresses, or addresses that correspond to locations in the system memory address space.



The previous typical message indicates that the operating system (FACILITY = SYS) is providing a notification (SEVERITY = 5) that it has been configured (MNEMONIC = CONFIG). The message text indicates that a user on vty0 from IP address 192.168.64.25 made this change.

Notes: One of the most common Syslog messages you'll see is line protocol up/down messages after a configuration change has been made in global configuration mode. Also, if ACL logging is enabled, Syslog messages will be generated when packets match ACL parameters. Syslog messages can create large amounts of network bandwidth. It is important to enable only syslog facilities and levels that are of particular importance.

Configuring Syslog

Configuring the local switch to store syslog messages in a buffer:

```
Switch(config)# logging buffered size-in-bytes
// size-in-bytes: 4096 <-> 2147483647
Switch(config)# logging buffered severity-level
// severity-level: 0 <-> 7
```

Configuring the local switch to show syslog messages on the console:

```
Switch(config)# logging console severity-level
```

Adding timestamps to syslog messages:

```
Switch(config)# service timestamps log datetime
[localtime] [show-timezone] [msec] [year]
```

Verifying Syslog

```
Switch(config)# show logging
// Shows the contents of the local log files.
```

Configuring Syslog to export events to an external Syslog server (UDP port 514):

```
Switch(config)# logging server-ip-address
Switch(config)# logging trap severity-level
```

Redirecting the console messages to your remote Telnet /SSH session:

```
Switch# terminal monitor
```

Configuring NTP Servers for accurate time stamping:

```
Switch(config)# ntp server ip-address | FQDN
Switch(config)# clock timezone timezone-name offset-hours
Switch(config)# clock summer-time summer-time-name
recurring [start-week day month hh:mm end-week day month
hh:mm [offset-minutes]]
```

```
Switch# show ntp status | associations
```

CDP

Cisco Discovery Protocol (CDP) is a Cisco-proprietary protocol that can be used to discover only Cisco network devices. CDP is media and protocol independent. The requirement is that the media support SNAP encapsulation. CDP runs at the **data link layer** of the OSI model. CDP uses hello messages; packets are exchanged between neighbors, but CDP information is not forwarded. In addition to routers and switches, IP phones and Cisco Unified Communication Manager (CUCM) servers also advertise CDP information.

It is useful when SNMP community strings are unknown when performing a network discovery.

When displaying CDP neighbors, you can obtain the following information:

- **Local interface:** Local interface that is connected to the discovered neighbor
- **Device ID:** Name of the neighbor device and MAC address or serial number
- **Device IP address:** IP address of the neighbor
- **Hold time:** How long (seconds) to hold the neighbor information
- **Device capabilities:** Type of device discovered: router, switch, transparent bridge, host, IGMP, repeater
- **Version:** IOS or switch OS version
- **Platform:** Router or switch model number
- **Port ID:** Interface of the neighboring device

Network management devices can obtain CDP information for data gathering. CDP should be disabled on untrusted interfaces, such as those that face the Internet, third party networks, or other unsecure networks.

SNMP

2/26/15

SNMP is simply the standard for network monitoring and management . It was initially described in [RFC 1157](#). It is a simple solution that requires little code to implement, which allows vendors to build SNMP agents on their products.

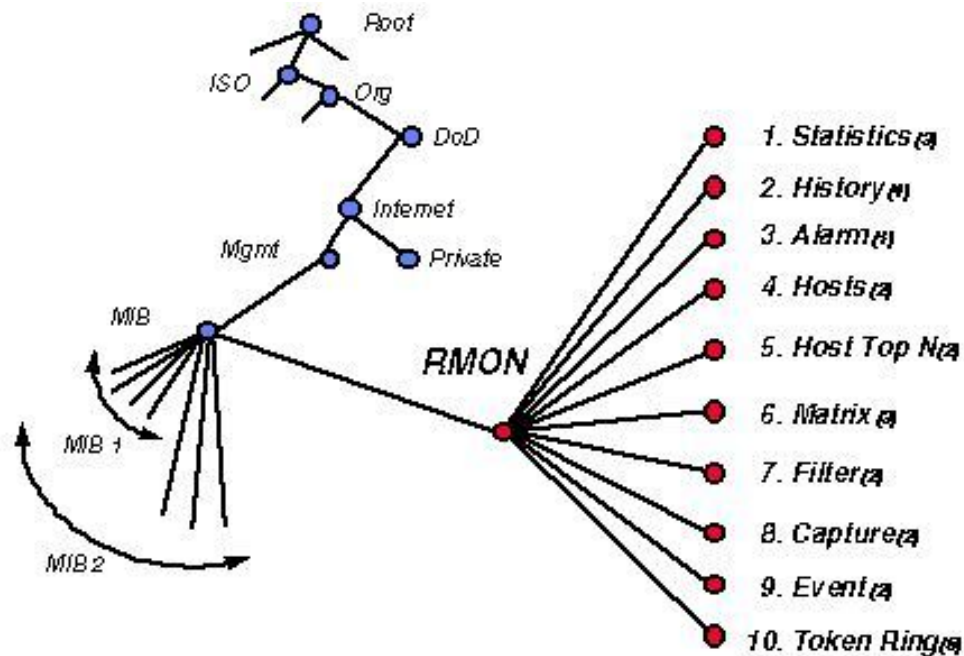
It contains three core elements:

- Network Management System (SNMP Manager)
- SNMP Agents (running inside a managed device)
- MIB Database objects that describes the information requested (inside the agent)

MIB

A Management Information Base (MIB) is a collection of information that is stored on the local agent of the managed device and is accessed by the NMS. MIBs are databases of objects organized in a tree-like structure, with each branch containing similar objects. Each object has a unique object identifier (number) that uniquely identifies the managed object of the MIB hierarchy. Read and write community strings are used to control access to MIB information. The top-level MIB object IDs belong to different standards organizations, and lower-level object IDs are allocated to associated organizations. Standard MIBs are defined by RFCs.

Vendors define private branches that include managed objects for their products. Cisco defines the MIBs under the Cisco head object. A Cisco MIB can be uniquely identified by either the object name, *iso.org.dod.internet.private.enterprise.cisco*, or the equivalent object descriptor, 1.3.6.1.4.1.9.



RMON MIB provides more information than other MIBs, but more sophisticated data collection devices (network probes) are needed. In short, RMON MIB is designed for "*flow-based*" monitoring, while other SNMP MIBs are often used for "*device-based*" management. Enterprise networks deploy network probes on several network segments; these probes report back to the RMON console. RMON allows network statistics to be collected even if a failure occurs between the probe and the RMON console. The RMON MIB is located at *iso.org.dod.internet.mgt.mib.rmon*.

RMON1 looks at the data link and physical layers of the OSI model and provides aggregate information on the statistics and LAN traffic.

ID	Name	Description
1	Statistics	Contains real-time statistics for interfaces: packets sent, bytes, cyclic redundancy check (CRC) errors, fragments.
2	History	Stores periodic statistic samples for later retrieval.
3	Alarm	An alarm event is generated if a statistic sample crosses a threshold.
4	Host	Host-specific statistics.
5	HostTopN	Most active hosts.
6	Matrix	Stores statistics for conversations between two hosts.
7	Filters	Allows packets to be filtered.
8	Packet Capture	Allows packets to be captured for subsequent analysis.
9	Events	Generates notification of events.

RMON2 provides an extension for monitoring upper-layer protocols (3 <-> 7).

ID	Name	Description
11	Protocol Directory	Lists the protocols the device supports
12	Protocol Distribution	Traffic statistics for each protocol
13	Address Mapping	Contains network-to-MAC layer address mapping (IP to MAC)
14	Network Layer Host	Contains statistics for traffic sent to or from network layer hosts
15	Network Layer Matrix	Contains statistics for conversations between two network layer hosts
16	Application Layer Host	Contains application layer statistics for traffic sent to or from each host
17	Application Layer Matrix	Contains application layer statistics for conversations between pairs of hosts
18	User History	Contains periodic samples of specified variables
19	Probe Configuration	Probes parameter configuration

SNMP network management applications periodically uses UDP to poll the agent residing on a managed device for useful, predetermined information from the MIB(s). The problem is it polls the device on a set schedule, so there will be a lag between when an event occurs and when the application learns of it. SNMP traps are not so passive. When certain criteria are met, the agent sends the manager a notification instantly, so it no longer has to wait around to find out. This can introduce bandwidth savings. Think of it like push notification in the cellular world. The data that the agent collects is stored in its MIB. Community strings are used to provide a level of authorization for the MIB contents (read or write) - kind of like weak SNMP passwords. They are transmitted in clear text across the network, so be careful.

SNMP Versions

Version	Authentication	Data Protection	Unique Features	
SNMPv1	Community string	None	32-bit counters	- Get - GetNext - Response - Set - Trap
SNMPv2c	Community string	None	Added Bulk Request and Inform Request message types, 64-bit counters	
SNMPv3	Username authentication	Hash-based MAC (SHA or MD5) Following encryption types: * CBC-DES * CBC-3DES * CFB-AES-128/192/256	Added user authentication, data integrity, and encryption } Cipher Block Chaining Mode } Cipher Feedback Mode	

- **GetBulk**
 - **Inform**: allows one NMS manager to send trap information to other NMS managers and to receive information.

Three security levels:

- **noAuthNoPriv (noAuth)**: Neither authenticates nor encrypts
- **authNoPriv (auth)**: Authenticates the sender but does not encrypt the message
- **authPriv (priv)**: Both authenticates the sender and encrypts the message

Configuring SNMP

1. Configure SNMP **ACLs** (optional, but recommended): range of IP addresses that are allowed to use the community string to gain access to the SNMP agent
2. Configure **Community Strings** (only for version 1/2c)
3. Configure SNMP **group/user** (only for version 3)
4. Configure SNMP **trap destination/type**

Configuring SNMPv1/2c on local switch for Securing SNMP Manager → Switch Agent messages: GET/SET

The same RO/RW community strings must also be defined on the SNMP manager so it can successfully communicate with the switch.

```
Switch(config)# access-list acl-number permit ip {any | host src-ip-addr | src-ip-addr wildcard-mask}
                                     {any | host dst-ip-addr | dst-ip-addr wildcard-mask} [log]
Switch(config)# snmp-server community password {RO | RW} acl-number
```

Configuring SNMPv3 on local switch for Securing SNMP Manager → Switch Agent messages: GET/SET

The same group name, username, authentication method and password, and encryption method and password must also be defined on the SNMP manager so it can successfully communicate with the switch.

```
Switch(config)# access-list acl-number permit ip {any | host src-ip-addr | src-ip-addr wildcard-mask}
                                     {any | host dst-ip-addr | dst-ip-addr wildcard-mask} [log]
Switch(config)# snmp-server group group-name v3 {noauth | auth | priv}
Switch(config)# snmp-server user username group-name v3 [encrypted]
                                     [auth {md5| sha} auth-password] [priv {des| 3des| aes {128 | 192 | 256} priv-password}]
                                     [acl-for-authorized-snmp-servers]
```

Configuring SNMPv1/2c/3 on local switch for the destination of Switch Agent → SNMP Manager messages: TRAP

By default, all trap types are sent.

```
Switch(config)# snmp-server host server-ip-addr [traps | informs] version {1|2c|3} {noauth | auth | priv}
                                     {v1/v2c-community-string | v3-username} [trap-type]
Switch(config)# snmp-server enable traps trap-type
```

Verifying SNMP

```
Switch# show snmp
```

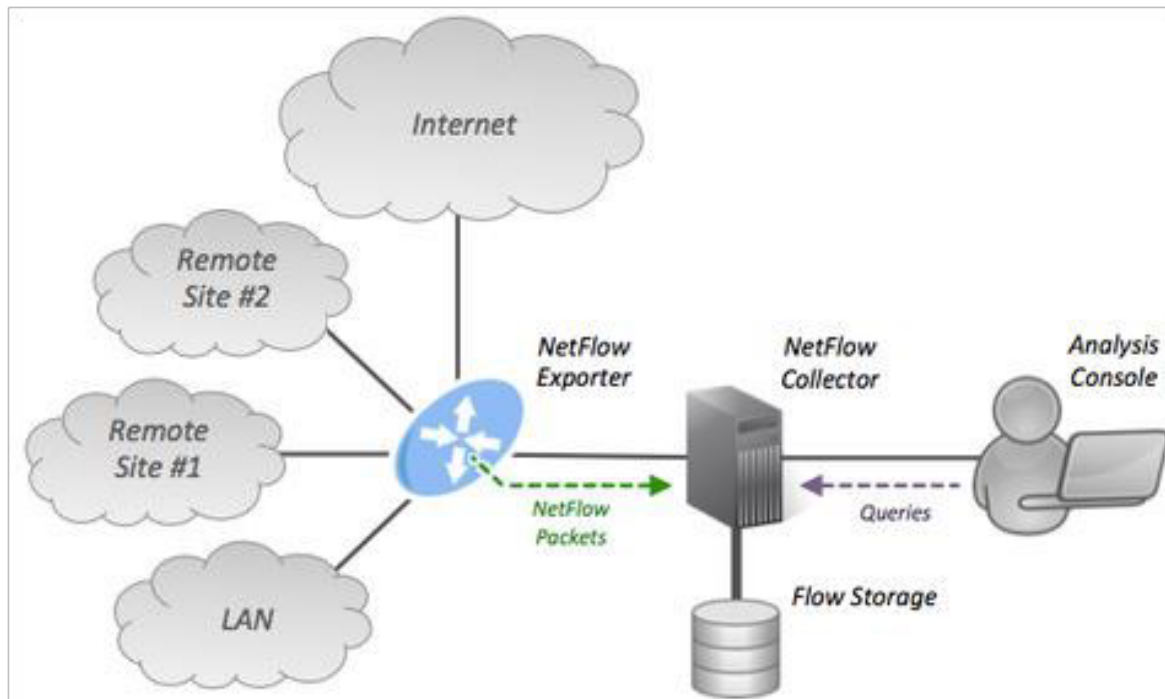
Netflow

2/26/15

NetFlow is a network protocol developed by Cisco Systems for collecting IP flows as they are passed through routers and multilayer switches. Routers and switches ('**NetFlow accounting**' or '**Netflow Exporter**') that support NetFlow can collect **IP traffic statistics on all ingress interfaces where NetFlow is enabled**, and later export those statistics as NetFlow records, toward at least one **NetFlow collector** - typically a server that does the actual traffic analysis. Cisco standard NetFlow version 5 defines **a flow as a unidirectional sequence of packets that all share the following 7 values: Ingress interface, IP ToS, Source IP address, Destination IP address, IP protocol, Source port for UDP or TCP, 0 for other protocols, Destination port for UDP or TCP, type and code for ICMP, or 0 for other protocols.**

The router will **output a flow record when it determines that the flow is finished**. NetFlow records are traditionally **exported using UDP**. For efficiency reasons, the router traditionally does not keep track of flow records already exported, so if a NetFlow packet is dropped due to network congestion or packet corruption, all contained records are lost forever. The UDP protocol does not inform the router of the loss so it cannot send the packets again. This can be a real problem, especially with NetFlow v8 or v9 that can aggregate a lot of packets or flows into a single record. A single UDP packet loss can cause a huge impact on the statistics of some flows. That is why some **modern implementations of NetFlow use SCTP** (Stream Control Transmission Protocol) to export packets so as to provide some protection against packet loss. Note that TCP would not be suitable for NetFlow because a strict ordering of packets would cause excessive buffering and delays.

The problem with SCTP is that it requires interaction between each NetFlow collector and each router exporting NetFlow. There may be **performance limitations** if a router has to deal with many NetFlow collectors, and if a NetFlow collector has to deal with lots of routers, especially when some of them are unavailable due to failure or maintenance.



NetFlow data records contains the following information (*NetFlow v9 and v10 exports additional fields*):

- Routing information (next-hop address, source and destination AS number, destination prefix mask)
- Source and destination IP addresses
- Type of service (ToS)
- Source and destination TCP/UDP ports
- TCP flags and encapsulated protocol (TCP/UDP)
- Packet and byte counts
- Input and output interface numbers
- Start and end timestamps

Network data analyzers display a graphical user interface (GUI) and analyze NetFlow data collected from flow collector files. This allows users to complete **near real-time visualization** or trending analysis of recorded and aggregated flow data. Users can specify the router and aggregation scheme and desired time interval.

NetFlow data can be analyzed for the following key applications:

- **Accounting and billing:** Used by service providers for charging based on bandwidth and application usage and quality of service (QoS).
- **Network planning and analysis:** Link and router capacity.
- **Network and security monitoring:** Visualize real-time traffic patterns with denial-of-service (DoS) monitoring capabilities.
- **Application monitoring and profiling:** Time-based view of application usage.
- **User monitoring and profiling:** Identifies customer and user network utilization and resource application.
- **NetFlow data warehousing and mining:** NetFlow data can be warehoused for later retrieval and analysis.

Netflow vs SNMP RMON

NetFlow enables you to gather more statistical information than RMON with fewer resources. It provides greater detail of the collected data, with ToS, routing information, TCP flags, date and time stamping. NetFlow has greater scalability and **does not require network probes**. As compared with SNMP, **NetFlow is push-based, whereas RMON is poll-based**.

	Technology	Description
Devices-based Management	Syslog	Reports state information based on facility and severity levels
	CDP	Proprietary protocol for network discovery that provides information on neighboring devices
	SNMP	Used in NMS to monitor network devices implementing agents that gather configuration and statistics information into MIBs and generate trap events
Flow-based Management	RMON	Remote Monitoring. Provides aggregate information of network statistics and LAN traffic
	NetFlow	Collects network flow data for network planning, performance, accounting, and billing applications