CCNPv7.1 SWITCH

Chapter 4 Lab 4-1– Implement Spanning Tree Protocols

1. Topology



1. Objectives
* Observe default Spanning Tree behavior
* Implement Rapid Spanning Tree
* Implement STP tool kit components
1. Background

The potential effect of a loop in the layer 2 network is significant. Layer 2 loops could impact connected hosts as well as the network equipment. Layer 2 loops can be prevented by following good design practices and careful implementation of the Spanning Tree Protocol. In this lab you will observe and manipulate the operation of spanning tree protocols to help secure the layer 2 network from loops and topology disruptions. The terms "switch" and "bridge" will be used interchangeably throughout the lab.

**Note:** This lab uses Cisco Catalyst 3560 and 2960 switches running Cisco IOS 15.0(2)SE6 IP Services and LAN Base images, respectively. The 3560 and 2960 switches are configured with the SDM templates “dual-ipv4-and-ipv6 routing” and “lanbase-routing”, respectively. Depending on the switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab. Catalyst 3650 switches (running any Cisco IOS XE release) and Catalyst 2960-Plus switches (running any comparable Cisco IOS image) can be used in place of the Catalyst 3560 switches and the Catalyst 2960 switches.

Required Resources

* 2 Cisco 2960 with the Cisco IOS Release 15.0(2)SE6 C2960-LANBASEK9-M or comparable
* 2 Cisco 3560v2 with the Cisco IOS Release 15.0(2)SE6 C3560-ipservicesK9-M or comparable
* Computer with terminal emulation software
* Ethernet and console cables
* 1 Windows 7 PC with Wireshark, TCPDump, or another comparable packet capture utility installed
1. Observe default Spanning Tree behavior
	1. Load base config and configure trunks

Use the **reset.tcl** script you created in Lab 1 “Preparing the Switch” to set your switches up for this lab. Then load the file BASE.CFG into the running-config with the command **copy flash:BASE.CFG running-config**.

Perform this step on all four switches. An example from DLS1:

DLS1# **tclsh reset.tcl**

Erasing the nvram filesystem will remove all configuration files! Continue? [confirm]

[OK]

Erase of nvram: complete

Reloading the switch in 1 minute, type reload cancel to halt

Proceed with reload? [confirm]

\*Mar 7 18:41:40.403: %SYS-7-NV\_BLOCK\_INIT: Initialized the geometry of nvram

\*Mar 7 18:41:41.141: %SYS-5-RELOAD: Reload requested by console. Reload Reason: Reload command.

*<switch reloads - output omitted>*

Would you like to enter the initial configuration dialog? [yes/no]: **n**

Switch> **en**

\*Mar 1 00:01:30.915: %LINK-5-CHANGED: Interface Vlan1, changed state to administratively down

Switch# **copy BASE.CFG running-config**

Destination filename [running-config]?

184 bytes copied in 0.310 secs (594 bytes/sec)

DLS1#

Next, enable interfaces F0/7 through F0/12 as 802.1Q trunk ports. Perform this step on all four switches. An example from DLS1:

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **int ran f0/7-12**

DLS1(config-if-range)# **switchport trunk encap dot1q**

DLS1(config-if-range)# **switchport trunk native vlan 666**

DLS1(config-if-range)# **switchport trunk allowed vlan except 1,999**

DLS1(config-if-range)# **switchport mode trunk**

DLS1(config-if-range)# **switchport nonegotiate**

DLS1(config-if-range)# **no shut**

DLS1(config-if-range)# **exit**

DLS1(config)#

Finally, configure all four switches as VTP version 3 servers in domain SWLAB with no password. An example from DLS1:

DLS1(config)# **vtp mode server**

Setting device to VTP Server mode for VLANS.

DLS1(config)# **vtp domain SWLAB**

Changing VTP domain name from NULL to SWLAB

DLS1(config)# **vtp version 3**

DLS1(config)#

* 1. Configure VLANs

Configure DLS1 as the VTP primary server for VLANs, and then create VLANs. The VLAN database will propagate to the other switches in the network.

DLS1# **vtp primary vlan**

This system is becoming primary server for feature vlan

No conflicting VTP3 devices found.

Do you want to continue? [confirm]

DLS1#

\*Mar 1 01:35:22.917: %SW\_VLAN-4-VTP\_PRIMARY\_SERVER\_CHG: e840.406f.7280 has become the primary server for the VLAN VTP feature

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **vlan 99**

DLS1(config-vlan)# **name MANAGEMENT**

DLS1(config-vlan)# **vlan 100**

DLS1(config-vlan)# **name SERVERS**

DLS1(config-vlan)# **vlan 110**

DLS1(config-vlan)# **name GUEST**

DLS1(config-vlan)# **vlan 120**

DLS1(config-vlan)# **name OFFICE**

DLS1(config-vlan)# **vlan 999**

DLS1(config-vlan)# **name PARKING\_LOT**

DLS1(config-vlan)# **state suspend**

DLS1(config-vlan)# **vlan 666**

DLS1(config-vlan)# **name NATIVE\_DO\_NOT\_USE**

DLS1(config-vlan)# **exit**

DLS1(config)#

ALS2# **show vtp status**

VTP Version capable : 1 to 3

VTP version running : 3

VTP Domain Name : SWLAB

VTP Pruning Mode : Disabled

VTP Traps Generation : Disabled

Device ID : 5017.ff84.0a80

Feature VLAN:

--------------

VTP Operating Mode : Server

Number of existing VLANs : 11

Number of existing extended VLANs : 0

Maximum VLANs supported locally : 255

Configuration Revision : 7

Primary ID : e840.406f.7280

Primary Description : DLS1

MD5 digest : 0xF3 0xD5 0xF7 0x62 0x3F 0x7C 0x84 0x86

 0x41 0xC0 0x4E 0xCA 0x36 0xB8 0x15 0x47

<output omitted>

ALS2# **show vlan brief | i active**

1 default active Fa0/1, Fa0/2, Fa0/3, Fa0/4

99 MANAGEMENT active

100 SERVERS active

110 GUEST active

120 OFFICE active

666 NATIVE\_DO\_NOT\_USE active

ALS2#

* 1. Identify and modify the root bridge

Use the **show span root** command on all of the switches to find the root switch for all of the VLANs. Note: Your results may vary from the examples.

DLS1# **show span root**

 Root Hello Max Fwd

Vlan Root ID Cost Time Age Dly Root Port

---------------- -------------------- --------- ----- --- --- ------------

VLAN0099 32867 5017.ff84.0a80 19 2 20 15 Fa0/9

VLAN0100 32868 5017.ff84.0a80 19 2 20 15 Fa0/9

VLAN0110 32878 5017.ff84.0a80 19 2 20 15 Fa0/9

VLAN0120 32888 5017.ff84.0a80 19 2 20 15 Fa0/9

VLAN0666 33434 5017.ff84.0a80 19 2 20 15 Fa0/9

DLS1#

ALS2# **show span root**

 Root Hello Max Fwd

Vlan Root ID Cost Time Age Dly Root Port

---------------- -------------------- --------- ----- --- --- ------------

VLAN0099 32867 5017.ff84.0a80 0 2 20 15

VLAN0100 32868 5017.ff84.0a80 0 2 20 15

VLAN0110 32878 5017.ff84.0a80 0 2 20 15

VLAN0120 32888 5017.ff84.0a80 0 2 20 15

VLAN0666 33434 5017.ff84.0a80 0 2 20 15

ALS2#

Compare the output of the **show span** command on all of the switches; why did the current root get elected?

DLS2#**show span vlan 99**

VLAN0099

 Spanning tree enabled protocol ieee

 Root ID Priority 32867

 Address 5017.ff84.0a80

 Cost 19

 Port 11 (FastEthernet0/9)

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 32867 (priority 32768 sys-id-ext 99)

 Address e840.406f.7280

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Aging Time 300 sec

Interface Role Sts Cost Prio.Nbr Type

------------------- ---- --- --------- -------- --------------------------------

Fa0/7 Altn BLK 19 128.9 P2p

Fa0/8 Altn BLK 19 128.10 P2p

Fa0/9 Root FWD 19 128.11 P2p

Fa0/10 Altn BLK 19 128.12 P2p

Fa0/11 Altn BLK 19 128.13 P2p

Fa0/12 Altn BLK 19 128.14 P2p

The current root bridge was elected based on the lowest Bridge ID (consisting of the Priority, extended system ID equal to the VLAN ID, and base MAC address values). In the output above, the root’s MAC is 5017.ff84.0a80; the local bridge MAC is e840.406f.7280.

With the priority and extended system IDs being identical, the root bridge's MAC is numerically smaller than the local bridge’s MAC. The end result is that in a completely un-configured network, one single switch will be elected as the root bridge. The resulting choice of switch may or may not be desirable.

There are two basic ways to manipulate the configuration to control the location of the root bridge.

* The **spanning-tree vlan** *vlan-id* **priority** *value* command can be used to manually set a priority value
* The **spanning-tree vlan** *vlan-id* **root { primary | secondary }** command can be used to automatically set a priority value.

The difference between the two is the **priority** command will set a specific number (multiple of 4096) as the priority, while the **root primary** command will set the local bridge's priority to 24,576 (if the local bridge MAC is lower than the current root bridge's MAC) or 4096 lower than the current root's priority (if the local bridge MAC is higher than the current root bridge's MAC).

The logic behind this operation is straight-forward. The **root primary** command tries to lower the priority only as much as is needed to win the root election, while leaving priorities between 24576 and the default 32768 for use by secondary bridges. The command always takes the entire Bridge ID into account when computing the resulting priority value.

The **spanning-tree vlan** *vlan-id* **root secondary** command will statically set the local bridge’s priority to 28,672. In an otherwise unconfigured network where all switch priorities default to 32,768, the **root primary** command will set the priority on the switch to 24,576 (two “steps” lower than the default priority) while the **root secondary** command will set the priority on the secondary root to the 28,672 (one “step” lower than the default priority).

Modify DLS1 and DLS2 so that DLS 1 is elected the primary root bridge for VLANs 99 and 100 and DLS2 is elected the primary root bridge for VLAN 110 and 120. DLS1 should be elected as the secondary root bridge for VLAN 110 and 120, and DLS2 should be elected as the secondary root bridge for VLANs 99, and 100.

You will need to make configuration changes on both DLS1 and DLS2.

An example from DLS1:

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **spanning-tree vlan 99,100 root primary**

DLS1(config)# **spanning-tree vlan 110,120 root secondary**

DLS1(config)# **exit**

DLS1#

Verification from DLS1:

DLS1# **show spanning-tree root**

 Root Hello Max Fwd

Vlan Root ID Cost Time Age Dly Root Port

---------------- -------------------- --------- ----- --- --- ------------

VLAN0099 24675 e840.406f.7280 0 2 20 15

VLAN0100 24676 e840.406f.7280 0 2 20 15

VLAN0110 24686 e840.406f.6e00 19 2 20 15 Fa0/11

VLAN0120 24696 e840.406f.6e00 19 2 20 15 Fa0/11

VLAN0666 33434 5017.ff84.0a80 19 2 20 15 Fa0/9

DLS1#

The **show spanning-tree bridge** command also provides detailed information about the current configuration of the local bridge:

DLS1# **show spanning-tree bridge ?**

 address Mac address of this bridge

 detail Detailed of the status and configuration

 forward-time Forward delay interval

 hello-time Hello time

 id Spanning tree bridge identifier

 max-age Max age

 priority Bridge priority of this bridge

 protocol Spanning tree protocol

 | Output modifiers

 <cr>

DLS1# **show spanning-tree bridge**

 Hello Max Fwd

Vlan Bridge ID Time Age Dly Protocol

---------------- --------------------------------- ----- --- --- --------

VLAN0099 24675 (24576, 99) e840.406f.7280 2 20 15 ieee

VLAN0100 24676 (24576, 100) e840.406f.7280 2 20 15 ieee

VLAN0110 28782 (28672, 110) e840.406f.7280 2 20 15 ieee

VLAN0120 28792 (28672, 120) e840.406f.7280 2 20 15 ieee

VLAN0666 33434 (32768, 666) e840.406f.7280 2 20 15 ieee

DLS1#

* 1. Manipulate port and path costs

As the network is implemented right now, there are two paths between each directly connected switch. As the Root Port is elected, path and port costs are evaluated to determine the shortest path to the root bridge.

In the case where there are multiple equal cost paths to the root bridge, additional attributes must be evaluated. In our case, the lower interface number (for example, F0/11) is chosen as the Root Port, and the higher interface number (for example, F0/12) is put into a spanning tree Blocking state.

You can see which ports are blocked with the **show spanning-tree** *vlan-id*command or the **show spanning-tree blockedports** command. For now examine VLAN 110 on DLS1.

DLS1# **show spanning-tree vlan 110**

VLAN0110

 Spanning tree enabled protocol ieee

 Root ID Priority 24686

 Address e840.406f.6e00

 Cost 19

 Port 13 (FastEthernet0/11)

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 28782 (priority 28672 sys-id-ext 110)

 Address e840.406f.7280

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Aging Time 300 sec

Interface Role Sts Cost Prio.Nbr Type

------------------- ---- --- --------- -------- --------------------------------

Fa0/7 Desg FWD 19 128.9 P2p

Fa0/8 Desg FWD 19 128.10 P2p

Fa0/9 Desg FWD 19 128.11 P2p

Fa0/10 Desg FWD 19 128.12 P2p

Fa0/11 Root FWD 19 128.13 P2p

Fa0/12 Altn BLK 19 128.14 P2p

DLS1# **show spanning-tree blockedports**

Name Blocked Interfaces List

-------------------- ------------------------------------

VLAN0110 Fa0/12

VLAN0120 Fa0/12

VLAN0666 Fa0/7, Fa0/8, Fa0/10, Fa0/11, Fa0/12

Number of blocked ports (segments) in the system : 7

As you can see, VLAN 110 has its Root Port on Fa0/11 and Fa0/12 is an Alternate Blocking Port. Note that despite the switch not yet running Rapid STP, it recognizes the port roles as known by RSTP.

It is possible to manipulate which port becomes the Root Port on non-root bridges by manipulating the port cost value, or by changing the port priority value. Remember that this change could have an impact on downstream switches as well. For this example, we will examine both options.

**Note**: The changes you are about to implement are considered topology changes and *could* have a significant impact on the overall structure of the spanning tree in your switch network. Do not make these changes in a production network without careful planning and prior coordination.

The first change you will make will influence the Root Port election based on a change to the port cost. We will further examine the impact of the changes to downstream switches.

To do this, issue the **shutdown** command on interfaces Fa0/9 and Fa0/10 on DLS1 and DLS2. Example from DLS1:

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **int ran f0/9-10**

DLS1(config-if-range)# **shut**

DLS1(config-if-range)# **exit**

DLS1(config)#

Then, examine the VLAN 110 root values on ALS1:

ALS1# **show span root**

 Root Hello Max Fwd

Vlan Root ID Cost Time Age Dly Root Port

---------------- -------------------- --------- ----- --- --- ------------

VLAN0099 24675 e840.406f.7280 19 2 20 15 Fa0/7

VLAN0100 24676 e840.406f.7280 19 2 20 15 Fa0/7

VLAN0110 24686 e840.406f.6e00 38 2 20 15 Fa0/7

VLAN0120 24696 e840.406f.6e00 38 2 20 15 Fa0/7

VLAN0666 33434 5017.ff84.0a80 19 2 20 15 Fa0/11

ALS1#

The election of the Root Port is based on the lowest total path cost to the root bridge. The root path cost is a sum of all of the Root Port costs between the local bridge and the root bridge. If the total path cost to the root bridge is the same over multiple ports, then the port towards the neighbor switch that has the lowest Bridge ID is chosen as the Root Port.

If the local bridge has multiple connections to a neighbor bridge that is in the lowest-cost path, BDPUs sent from that neighbor are examined and the BPDU containing the lowest sending Port-ID is chosen as the Root Port. In this case the term "sending" refers to the switch and its port that *forwarded* the BPDU.

Notice in the output above that the root bridge for VLAN110 is reachable from ALS1 via Fa0/7 with a total root path cost of 38 (19 for the Fa0/7 trunk between ALS1 and DLS1, and 19 for the trunk between DLS1 and DLS2).

On ALS2, change the spanning tree cost on interface Fa0/7 to 12.

ALS2(config)# **int f0/7**

ALS2(config-if)# **spanning-tree cost 12**

ALS2(config-if)# **exit**

ALS2(config)#

Now go back to ALS1 and see the impact:

ALS1# **show span root**

 Root Hello Max Fwd

Vlan Root ID Cost Time Age Dly Root Port

---------------- -------------------- --------- ----- --- --- ------------

VLAN0099 24675 e840.406f.7280 19 2 20 15 Fa0/7

VLAN0100 24676 e840.406f.7280 19 2 20 15 Fa0/7

VLAN0110 24686 e840.406f.6e00 31 2 20 15 Fa0/11

VLAN0120 24696 e840.406f.6e00 31 2 20 15 Fa0/11

VLAN0666 33434 5017.ff84.0a80 19 2 20 15 Fa0/11

ALS1#

ALS1's Root Port changed to F0/11, and the path cost to the Root Bridge changed to 31 (19 + 12).

The change you just made on ALS2 did not impact the Root Port from its perspective; it is still Fa0/7.

Next you will use port priority to modify which port is selected as the Root Port. For this exercise, we will focus on VLAN 100.

On DLS1, use **show span vlan 100** to see what the priorities are (default to 128)

DLS1# **show span vlan 100**

VLAN0100

 Spanning tree enabled protocol ieee

 Root ID Priority 24676

 Address e840.406f.7280

 This bridge is the root

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 24676 (priority 24576 sys-id-ext 100)

 Address e840.406f.7280

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Aging Time 300 sec

Interface Role Sts Cost Prio.Nbr Type

------------------- ---- --- --------- -------- --------------------------------

Fa0/7 Desg FWD 19 128.9 P2p

Fa0/8 Desg FWD 19 128.10 P2p

Fa0/11 Desg FWD 19 128.13 P2p

Fa0/12 Desg FWD 19 128.14 P2p

In the output above, focus on interface Fa0/7. Notice that its Port ID is made up of two values, labeled as Prio (Priority) and Nbr (Number): The priority number (128) and the port number (9).

The port number is not necessarily equal to the interface ID. On the 3560s used for creating this lab, port numbers 1 and 2 are assigned to G0/1 and G0/2 respectively, whereas on the 2960s G0/1 and G0/2 area assigned the port numbers 25 and 26. A switch may use any port number for STP purposed as long as they are unique for each port on the switch.

The port priority can be any value between 0 and 240, in increments of 16 (older switches may allow setting the priority in different increments).

Next, examine ALS1 to find the root port for VLAN 100:

ALS1# **show span root | i VLAN0100**

VLAN0100 24676 e840.406f.7280 19 2 20 15 Fa0/7

ALS1#

On DLS1, change the port priority value of Fa0/8 to 112:

DLS1(config)# **int f0/8**

DLS1(config-if)# **spanning-tree port-priority 112**

DLS1(config-if)# **exit**

And then examine the impact on ALS1:

ALS1# **show span root | i VLAN0100**

VLAN0100 24676 e840.406f.7280 19 2 20 15 Fa0/8

ALS1# **show span vlan 100**

VLAN0100

 Spanning tree enabled protocol ieee

 Root ID Priority 24676

 Address e840.406f.7280

 Cost 19

 Port 8 (FastEthernet0/8)

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 32868 (priority 32768 sys-id-ext 100)

 Address 64a0.e72a.2200

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Aging Time 15 sec

Interface Role Sts Cost Prio.Nbr Type

------------------- ---- --- --------- -------- --------------------------------

Fa0/7 Altn BLK 19 128.7 P2p

Fa0/8 Root FWD 19 128.8 P2p

Fa0/11 Desg FWD 19 128.11 P2p

Fa0/12 Desg FWD 19 128.12 P2p

Notice that the priority value at ALS1 doesn't change, but the Root Port did, based on DLS1's advertised port priorities.

* 1. Examine Re-convergence Time

Use the **debug spanning-tree events command** on DLS1 and watch how long re-convergence takes when interface Fa0/11 on DLS1 is shut down (Fa0/11 is DLS1’s Root Port for VLAN 110). The output below has been manually filtered for VLAN 110 related messages only:

DLS1# **debug span eve**

Spanning Tree event debugging is on

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **int f0/11**

DLS1(config-if)# **shut**

DLS1(config-if)#

\*Mar 1 02:08:50.781: STP: VLAN0110 new root port Fa0/12, cost 19

**\*Mar 1 02:08:50.789: STP: VLAN0110 Fa0/12 -> listening**

\*Mar 1 02:08:50.789: STP[110]: Generating TC trap for port FastEthernet0/11

\*Mar 1 02:08:50.7

\*Mar 1 02:08:52.769: %LINK-5-CHANGED: Interface FastEthernet0/11, changed state to administratively down

\*Mar 1 02:08:52.786: STP: VLAN0110 sent Topology Change Notice on Fa0/12

**\*Mar 1 02:09:05.797: STP: VLAN0110 Fa0/12 -> learning**

\*Mar 1 02:09:20.804: STP[110]: Generating TC trap for port FastEthernet0/12

\*Mar 1 02:09:20.804: STP: VLAN0110 sent Topology Change Notice on Fa0/12

**\*Mar 1 02:09:20.804: STP: VLAN0110 Fa0/12 -> forwarding**

DLS1(config-if)# **do sho span vlan 110**

VLAN0110

 Spanning tree enabled protocol ieee

 Root ID Priority 24686

 Address e840.406f.6e00

 Cost 19

 Port 14 (FastEthernet0/12)

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 28782 (priority 28672 sys-id-ext 110)

 Address e840.406f.7280

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Aging Time 15 sec

Interface Role Sts Cost Prio.Nbr Type

------------------- ---- --- --------- -------- --------------------------------

Fa0/7 Desg FWD 19 128.9 P2p

Fa0/8 Desg FWD 19 112.10 P2p

Fa0/12 Root FWD 19 128.14 P2p

As you can see from the timestamps, it took a full 30 seconds for PVST to settle on Fa0/12 as the root port and move to the Forwarding state on the Designated Ports. When Fa0/11 is reactivated:

DLS1(config-if)#no shut

\*Mar 1 02:12:28.902: set portid: VLAN0110 Fa0/11: new port id 800D

**\*Mar 1 02:12:28.902: STP: VLAN0110 Fa0/11 -> listening**

\*Mar 1 02:12:29.900: STP: VLAN0110 new root port Fa0/11, cost 19

\*Mar 1 02:12:29.900: STP: VLAN0110 sent Topology Change Notice on Fa0/11

\*Mar 1 02:12:29.900: STP [110]: Generating TC trap for port FastEthernet0/12

**\*Mar 1 02:12:29.900: STP: VLAN0110 Fa0/12 -> blocking**

**\*Mar 1 02:12:43.909: STP: VLAN0110 Fa0/11 -> learning**

\*Mar 1 02:12:58.916: STP[110]: Generating TC trap for port FastEthernet0/11

\*Mar 1 02:12:58.916: STP: VLAN0110 sent Topology Change Notice on Fa0/11

**\*Mar 1 02:12:58.916: STP: VLAN0110 Fa0/11 -> forwarding**

The re-convergence process took another full 30 seconds.

1. Implement Rapid Spanning Tree

By default Cisco switches are running Per-VLAN Spanning Tree, which is a Cisco-proprietary protocol derived from the IEEE 802.1D standard.

DLS1# **show span detail**

VLAN0099 is executing the **ieee compatible Spanning Tree protocol**

<output omitted>

The running configuration shows you the protocol being used

DLS1# **show run | inc spanning-tree mode**

**spanning-tree mode pvst**

DLS1#

The issue with PVST is that its convergence is quite slow. The time for the transition between port states is called *forward-delay* and by default, it is 15 seconds. In addition, the time until a BPDU stored on a port expires is called *max-age* and is 20 seconds by default. Depending on the nature of a topological change, STP requires between 30 and 50 seconds to converge on a new loop-free topology.

Rapid Spanning Tree significantly reduces the time it takes to go from the Discarding (PVST: Blocking) to the Forwarding state.

Configure Rapid Spanning Tree Protocol on DLS1. Use the **clear spanning-tree detected-protocols** privileged EXEC command to flush any stored PVST information.

DLS1# **conf t**

DLS1(config)# **spanning-tree mode rapid-pvst**

DLS1(config)# **end**

DLS1#**clear spanning-tree detected-protocols**

DLS1#

Then verify the protocol. Use the **show span vlan 110** command:

DLS1# **show span vlan 110**

VLAN0110

 Spanning tree enabled protocol rstp

 Root ID Priority 24686

 Address e840.406f.6e00

 Cost 19

 Port 13 (FastEthernet0/11)

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 28782 (priority 28672 sys-id-ext 110)

 Address e840.406f.7280

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Aging Time 300 sec

Interface Role Sts Cost Prio.Nbr Type

------------------- ---- --- --------- -------- --------------------------------

Fa0/7 Desg LRN 19 128.9 P2p Peer(STP)

Fa0/8 Desg LRN 19 112.10 P2p Peer(STP)

Fa0/11 Root FWD 19 128.13 P2p Peer(STP)

Fa0/12 Altn BLK 19 128.14 P2p Peer(STP)

Take note of the Type field in the output. All of the other switches are still running PVST, which is noted here by the entry Peer(STP).

Configure the rest of the switches to use Rapid Spanning Tree Protocol, then verify the protocol is running. An example from DLS2:

DLS2# **show span vlan 99**

VLAN0099

 Spanning tree enabled protocol rstp

To examine the impact of Rapid Spanning Tree on convergence time, use the **debug spanning-tree events command** on DLS1 and watch how long re-convergence takes when interface Fa0/11 on DLS1 is shut down (Fa0/11 is DLS1’s Root Port for VLAN 110). The output below has been manually filtered for VLAN 110 related messages only:

DLS1# **debug spanning-tree events**

Spanning Tree event debugging is on

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **int f0/11**

DLS1(config-if)# **shut**

DLS1(config-if)#

\*Mar 1 02:18:53.201: RSTP(110): updt roles, root port Fa0/11 going down

\*Mar 1 02:18:53.201: RSTP(110): Fa0/12 is now root port

\*Mar 1 02:18:53.201: RSTP(110): syncing port Fa0/7

\*Mar 1 02:18:53.201: RSTP(110): syncing port Fa0/8

\*Mar 1 02:18:53.226: STP[110]: Generating TC trap for port FastEthernet0/12

\*Mar 1 02:18:53.242: RSTP(110): transmitting a proposal on Fa0/7

\*Mar 1 02:18:53.242: RSTP(110): transmitting a proposal on Fa0/8

\*Mar 1 02:18:55.189: %LINK-5-CHANGED: Interface FastEthernet0/11, changed state to administratively down

\*Mar 1 02:18:56.195: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/11, changed state to down

In the output above, the change of Root Port and synchronization to interfaces Fa0/7 and Fa0/8 took less than six-tenths of a second

DLS1(config-if)# **no shut**

DLS1(config-if)#

\*Mar 1 02:22:38.368: RSTP(110): initializing port Fa0/11

\*Mar 1 02:22:38.368: RSTP(110): Fa0/11 is now designated

\*Mar 1 02:22:38.393: RSTP(110): transmitting a proposal on Fa0/11

\*Mar 1 02:22:38.401: RSTP(110): updt roles, received superior bpdu on Fa0/11

\*Mar 1 02:22:38.401: RSTP(110): Fa0/11 is now root port

\*Mar 1 02:22:38.401: RSTP(110): Fa0/12 blocked by re-root

\*Mar 1 02:22:38.409: RSTP(110): syncing port Fa0/7

\*Mar 1 02:22:38.409: RSTP(110): syncing port Fa0/8

\*Mar 1 02:22:38.409: RSTP(110): synced Fa0/11

\*Mar 1 02:22:38.409: RSTP(110): Fa0/12 is now alternate

\*Mar 1 02:22:38.418: STP[110]: Generating TC trap for port FastEthernet0/11

\*Mar 1 02:22:38.435: RSTP(110): transmitting an agreement on Fa0/11 as a response to a proposal

\*Mar 1 02:22:38.435: RSTP(110): transmitting a proposal on Fa0/7

\*Mar 1 02:22:38.435: RSTP(110): transmitting a proposal on Fa0/8

\*Mar 1 02:22:38.653: %LINK-3-UPDOWN: Interface FastEthernet0/11, changed state to up

\*Mar 1 02:22:39.659: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/11, changed state to up

In the output above, Fa0/11 is brought back up, and the change of root port and synchronization to interfaces Fa0/7 and Fa0/8 took one half second. This is a significant improvement from standard spanning tree.

1. Part 3: Implement STP tool kit components
	1. Implement and observe PortFast

In both STP and RSTP, a newly connected port must be guaranteed not to create a switching loop before it can become a Forwarding port. This may take up to 30 seconds. However, such a check is not necessary for ports connected to end devices that do not perform switching or bridging, such as workstations, network printers, servers, etc. In RSTP, these ports are called *edge* ports (ports that connect to other switches in the topology are called *non-edge* ports). Edge ports can safely enter the Forwarding state right after they come up, because by definition they do not connect to any device capable of forwarding frames.

Cisco developed a feature called PortFast that essentially allows defining a port as an edge port. Any PortFast enabled port will enter the Forwarding state immediately after coming up, without going through the intermediary non-forwarding states, saving 30 seconds each time a new connection is made to the port. PortFast can be used with all STP versions.

Apart from allowing a port to jump into the Forwarding state as soon as it is connected, the concept of an edge port is extremely important in RSTP and MSTP. Recall that as part of its improvements over legacy STP, RSTP uses a so-called Proposal/Agreement mechanism to rapidly, yet safely enable a link between switches if one of the switches has its Root port on that link.

Upon receiving a Proposal on its Root port, a switch puts all its non-edge Designated ports into the Discarding state, effectively cutting itself off the network and preventing a possible switching loop (this is called the Sync operation). When this is accomplished, the switch sends an Agreement back out its Root port so that the upstream Designated port receiving this Agreement can be immediately put into the Forwarding state. The switch will then start sending its own Proposals on all its non-edge Designated ports that have been just made Discarding, waiting for Agreements to arrive from downstream switches that would allow these ports to instantaneously become Forwarding again.

If end devices are connected to ports not configured as edge (that is, PortFast) ports, these ports will become Discarding during the Sync operation. Because end hosts do not support RSTP and cannot send back an Agreement, they will be cut off from the network for 30 seconds until the ports reach the Forwarding state using ordinary timers. As a result, users will experience significant connectivity outages.

Ports configured as edge ports are not affected by the Sync operation and will remain in the Forwarding state even during the Proposal/Agreement handling. Activating RSTP in a network without properly configuring ports toward end hosts as edge ports will cause the network to perform possibly even more poorly than with legacy STP. With RSTP, proper configuration of ports toward end hosts as edge ports is an absolute necessity. Cisco switches default to all their ports being non-edge ports.

For this lab step to work, Host A must be connected to ALS1’s Fa0/6.

On ALS1, issue the command **debug spanning-tree events** , then configure F0/6 to be in VLAN 120.

Finally, **no shut** the interface, **shut** the interface, and observe the syslog

ALS1# **debug spanning-tree events**

Spanning Tree event debugging is on

ALS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

ALS1(config)# **int f0/6**

ALS1(config-if)# **swi mo ac**

ALS1(config-if)# **swi ac vl 120**

ALS1(config-if)# **no shut**

ALS1(config-if)#

\*Mar 1 02:26:47.778: RSTP(120): initializing port Fa0/6

\*Mar 1 02:26:47.778: RSTP(120): Fa0/6 is now designated

\*Mar 1 02:26:47.786: RSTP(120): transmitting a proposal on Fa0/6

\*Mar 1 02:26:48.197: %LINK-3-UPDOWN: Interface FastEthernet0/6, changed state to up

\*Mar 1 02:26:48.759: RSTP(120): transmitting a proposal on Fa0/6

\*Mar 1 02:26:49.204: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/6, changed state to up

\*Mar 1 02:26:50.772: RSTP(120): transmitting a proposal on Fa0/6

\*Mar 1 02:26:52.786: RSTP(120): transmitting a proposal on Fa0/6

\*Mar 1 02:26:54.799: RSTP(120): transmitting a proposal on Fa0/6

\*Mar 1 02:26:56.812: RSTP(120): transmitting a proposal on Fa0/6

ALS1(config-if)# **shut**

\*Mar 1 02:26:58.825: RSTP(120): transmitting a proposal on Fa0/6

\*Mar 1 02:27:02.441: %LINK-5-CHANGED: Interface FastEthernet0/6, changed state to administratively down

\*Mar 1 02:27:03.448: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/6, changed state to down

ALS1(config-if)#

As you can see in the output above, RSTP sees the interface come up, recognizes it as a Designated port, and starts sending proposals. Now we will add the **spanning-tree portfast** command to the interface (the debug is still running):

ALS1(config-if)# **spanning-tree portfast**

%Warning: portfast should only be enabled on ports connected to a single

 host. Connecting hubs, concentrators, switches, bridges, etc... to this

 interface when portfast is enabled, can cause temporary bridging loops.

 Use with CAUTION

%Portfast has been configured on FastEthernet0/6 but will only

 have effect when the interface is in a non-trunking mode.

ALS1(config-if)#

ALS1(config-if)# **no shut**

ALS1(config-if)#

\*Mar 1 02:28:13.534: RSTP(120): initializing port Fa0/6

\*Mar 1 02:28:13.534: RSTP(120): Fa0/6 is now designated

\*Mar 1 02:28:13.945: %LINK-3-UPDOWN: Interface FastEthernet0/6, changed state to up

\*Mar 1 02:28:14.952: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/6, changed state to up

ALS1(config-if)# **shut**

\*Mar 1 02:28:35.999: %LINK-5-CHANGED: Interface FastEthernet0/6, changed state to administratively down

\*Mar 1 02:28:37.006: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/6, changed state to down

ALS1(config-if)# **no shut**

ALS1(config-if)#

\*Mar 1 02:28:51.434: RSTP(120): initializing port Fa0/6

\*Mar 1 02:28:51.434: RSTP(120): Fa0/6 is now designated

ALS1(config-if)#

\*Mar 1 02:28:51.761: %LINK-3-UPDOWN: Interface FastEthernet0/6, changed state to up

\*Mar 1 02:28:52.768: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/6, changed state to up

Disable the debug using the **undebug all** command.

Notice in output above that with PortFast configured, no proposals are sent out of interface Fa0/6; the port goes into Forwarding state immediately.

ALS1# **show span detail | beg VLAN0120**

 VLAN0120 is executing the rstp compatible Spanning Tree protocol

<output omitted>

 Port 6 (FastEthernet0/6) of VLAN0120 is designated forwarding

 Port path cost 19, Port priority 128, Port Identifier 128.6.

 Designated root has priority 24696, address e840.406f.6e00

 Designated bridge has priority 32888, address 64a0.e72a.2200

 Designated port id is 128.6, designated path cost 31

 Timers: message age 0, forward delay 0, hold 0

 Number of transitions to forwarding state: 1

 The port is in the portfast mode

 Link type is point-to-point by default

 BPDU: sent 39, received 0

**Note(1)**: PortFast should never be enabled on ports connected to another switches. Doing so could cause a switching loop. RSTP and MSTP have their own mechanisms to put inter-switch links into Forwarding state rapidly.

**Note(2)**: On trunk interfaces, configuring the **spanning-tree portfast** command will have no effect. This is a safety precaution, as trunks are usually connected to other switches. However, in situations like inter-VLAN routing using a router-on-stick, or when a trunk is being connected to a server that operates on multiple VLANs simultaneously, it may still be advantageous, and safe, to allow this trunk to be treated as an edge port and become Forwarding as soon as it is connected. In these cases, you can use the **spanning-tree portfast trunk** command on a trunk port to force a switch to treat it as an edge port regardless of its operating mode. Be absolutely sure that the device connected to such port is not performing Layer2 switching before using this command.

**Note(3)**: Because the proper configuration of edge ports in RSTP and MSTP is of such great importance for proper network performance, Cisco also provides the way of globally configuring the PortFast on all access ports using the **spanning-tree portfast default** global configuration command. With this command configured, each port that operates in the access mode will automatically have PortFast enabled. Trunk ports will not be affected. The logic of this behavior is simple: Usually, trunk ports connect to other switch where PortFast should never be enabled, while access ports usually connect to end devices.

* 1. Implement and Observe BPDU Guard

PortFast causes an interface to go into Forwarding state immediately. There is a risk that if two PortFast-enabled ports are inadvertently or maliciously connected together, they will both come up as Forwarding ports, immediately creating a switching loop.

The default, expected behavior of a PortFast port that receives a BPDU is for that port to revert to a normal spanning-tree non-edge port. There is the potential that the load on a given switch might be too great to handle the received BPDU properly, prolonging the loop condition.

BPDU Guard adds another layer of protection. Whenever a port protected by BPDU Guard unexpectedly receives a BPDU, it is immediately put into err-disabled state. Any interfaces can be protected with BPDU Guard, but its typical use in on PortFast-enabled ports.

BPDU Guard can be configured globally, or on a per-interface basis. If the BPDU Guard is configured on the global level using the **spanning-tree portfast bpduguard default** command, the BPDU Guard will be automatically enabled on all PortFast-enabled ports of the switch. If the BPDU Guard is configured on a particular interface using the **spanning-tree bpduguard enable** command, it will apply to this port unconditionally, regardless of whether it is a PortFast-enabled port.

For this example, we will configure BPDU guard on a trunking interface that is a non-root port on ALS2. Configuring BPDU Guard on an interface that is intended to be a trunk is *not a recommended practice*; we are doing this just to demonstrate the functionality of the tool.

ALS2# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

ALS2(config)# **int f0/11**

ALS2(config-if)# **spanning-tree bpduguard enable**

ALS2(config-if)# **exit**

ALS2(config)#

\*Mar 1 02:30:57.792: %SPANTREE-2-BLOCK\_BPDUGUARD: Received BPDU on port Fa0/11 with BPDU Guard enabled. Disabling port.

\*Mar 1 02:30:57.792: %PM-4-ERR\_DISABLE: bpduguard error detected on Fa0/11, putting Fa0/11 in err-disable state

\*Mar 1 02:30:58.798: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/11, changed state to down

\*Mar 1 02:30:59.813: %LINK-3-UPDOWN: Interface FastEthernet0/11, changed state to down

As you can see, the interface is almost immediately err-disabled. Revert the configuration settings and issue the **shutdown** and **no shutdown** commands on Fa0/11 to bring it back up.

ALS2(config)# **int f0/11**

ALS2(config-if)# **shut**

ALS2(config-if)# **spanning-tree bpduguard disable**

ALS2(config-if)# **no shut**

ALS2(config-if)# **exit**

ALS2(config)#

\*Mar 1 02:32:06.092: %LINK-3-UPDOWN: Interface FastEthernet0/11, changed state to up

\*Mar 1 02:32:07.107: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/11, changed state to up

* 1. Implement and Observe BPDU Filter

Neither PortFast nor BPDU Guard prevents the switch from sending BPDUs on an interface; if such a behavior is required, BPDU Filter can be used. It can be configured either globally or at a specific interface.

If BPDU Filter is configured on the global level using the **spanning-tree portfast bpdufilter default** global configuration command, the BPDU Filter applies only to PortFast-enabled ports. When these ports come up, they will *send* up to 11 BPDUs and then stop sending further BPDUs. If the BPDU Filter-configured interface *receives* a BPDU *at any time*, the BPDU Filter and PortFast will be deactivated on that port and it will become a normal spanning tree interface. As a result, a globally configured BPDU Filter does not prevent ports from receiving and processing BPDUs; it only attempts to stop sending BPDUs on ports where most probably, there is no device attached that would be interested in processing them.

Configured at an interface, BPDU Filter causes the port to stop sending and processing received BPDUs altogether. This can be used, for example, to split a network into two or more independent STP domains, each having its own root bridge and resulting topology. However, because these domains are no longer protected against mutual loops by STP, it is the task of the network administrator to make sure that these two domains are never connected by more than just a single link.

To test this feature, we will run a packet capture utility on Host A connected to ALS1, configure BPDU Filter on interface Fa0/6, and see that BPDUs stop being transmitted.

Login to Host A, run your packet capture utility and filter the output to show only STP packets. In Wireshark, the filter bar should look something like this:



You should see a BPDU being received at your host every 2 seconds.



Now configure BPDU Filter on ALS1 interface Fa0/6

ALS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

ALS1(config)# **int f0/6**

ALS1(config-if)# **spanning-tree bpdufilter enable**

ALS1(config-if)# **exit**

ALS1(config)#

Clear the Wireshark display filter and observe your packet capture window; you will have stopped receiving BPDUs.



* 1. Implement and Observe Root Guard

Root Guard helps prevent a root switch or Root Port takeover. It is configured on the port to be protected. If a port protected by Root Guard receives a superior BPDU that would normally cause the port to become a Root port, the BPDU will be discarded and the port will be moved to the Root-Inconsistent state. An STP inconsistent state differs from the error disabled state that the port is not disabled entirely; instead, it is only put into the Blocking (Discarding) state and will be put back into its proper role and state once the cause for its inconsistent state disappears. With Root Guard, a port will be reinstated into its appropriate role and state automatically when it stops receiving superior BPDUs.

**Note**: BPDU Root Guard is a protective mechanism in situations when your network and the network of your customer need to form a single STP domain, yet you want to have the STP root bridge in your network part and you do not want your customer to take over this root switch selection, or back up the connectivity in your network through the customer. In these cases, you would put the Root Guard on ports toward the customer. However, inside your own network, using Root Guard would actually be harmful. Your network can be considered trustworthy and there is no rogue root switch to protect against. Using Root Guard in your own network would cause it to be unable to converge on a new workable spanning tree if any of the primary links failed, and it would also prevent your network from converging to a secondary root switch if the primary root switch failed entirely.

To illustrate the behavior of Root Guard, we will configure it on a designated port on DLS1 for VLAN 100. DLS1 is the root bridge for VLAN 100, so all trunk ports are designated.

DLS1# **show span root**

 Root Hello Max Fwd

Vlan Root ID Cost Time Age Dly Root Port

---------------- -------------------- --------- ----- --- --- ------------

VLAN0099 24675 e840.406f.7280 0 2 20 15

VLAN0100 24676 e840.406f.7280 0 2 20 15

VLAN0110 24686 e840.406f.6e00 19 2 20 15 Fa0/11

VLAN0120 24696 e840.406f.6e00 19 2 20 15 Fa0/11

VLAN0666 33434 5017.ff84.0a80 38 2 20 15 Fa0/7

From the ALS1 side of things, the root port is interface F0/8. Normally it would be F0/7, but we changed the port priority of F0/8 to 112, and this impacts root port selection at ALS1 when all interfaces are operational:

ALS1# **show span root | inc VLAN0100**

VLAN0100 24676 e840.406f.7280 19 2 20 15 Fa0/8

Configure Root Guard on DLS1 interface Fa0/8 (you may immediately see errors with another VLAN, like 666. Ignore these as we are focusing on VLAN 100):

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **int f0/8**

DLS1(config-if)# **spanning-tree guard root**

DLS1(config-if)# **exit**

DLS1(config)#

Then go to ALS1 and configure it to be the root for VLAN 100 using the priority 16384

ALS1(config)# **spanning-tree vlan 100 priority 16384**

Then back at DLS1, check the spanning tree interface status for Fa0/8:

DLS1# **show spanning-tree interface f0/8 | inc VLAN0100**

VLAN0100 Desg BKN\*19 112.10 P2p \*ROOT\_Inc

DLS1#

This output has two indicators of the issue. First BKN\* is short for "BROKEN", and \*ROOT\_Inc represents the Root Inconsistent message. A list of all STP inconsistent ports including the reason for their inconsistency can also be requested with the command **show spanning-tree inconsistentports**.

DLS1# **show spanning-tree inconsistentports**

Name Interface Inconsistency

-------------------- ------------------------ ------------------

VLAN0100 FastEthernet0/8 Root Inconsistent

VLAN0666 FastEthernet0/8 Root Inconsistent

Number of inconsistent ports (segments) in the system : 2

DLS1#

To clear this, go back to ALS1 and issue the command **no spanning-tree vlan 100 priority 16384**. Once you do this, you will see the following SYSLOG message at DLS1, and the interface will become consistent again.

DLS1#

\*Mar 1 02:54:06.761: %SPANTREE-2-ROOTGUARD\_UNBLOCK: Root guard unblocking port FastEthernet0/8 on VLAN0100.

DLS1# **show spanning-tree interface f0/7 | inc VLAN0100**

VLAN0100 Desg FWD 19 128.9 P2p

DLS1#

For completeness, remove Root Guard from Fa0/7 on DLS1

DLS1# **conf t**

Enter configuration commands, one per line. End with CNTL/Z.

DLS1(config)# **int f0/8**

DLS1(config-if)# **no spanning-tree guard root**

DLS1(config-if)# **exit**

DLS1(config)#

* 1. Implement and Observe Loop Guard

The last tool we will demonstrate is Loop Guard. Its job is to prevent Root and Alternate ports from becoming Designated ports if BPDUs suddenly cease to be received on them

In a normal STP network, all ports receive and process BPDUs, even Blocking (Discarding) ports. This is how they know that the device at the other end of the link is alive and still superior to them. If a Blocked port stops receiving these BPDUs, it can only assume that the device on the other side is no longer there and they are now superior, and should be in Forwarding state for the given segment. An example of when this could occur is the instance where the Rx fiber in an optical cable becomes disconnected, cut, or connected to a different port or device than the corresponding Tx fiber, in essence creating an uni-directional link.

This could cause permanent switching loops in the network, so Loop Guard helps to prevent them.

For this example, we will configure Loop Guard on an Alternate port on ALS2, and then stop sending out BPDUs from the corresponding Designated port on the other end of the link, and observe the behavior.

ALS2# **show span vlan 100**

VLAN0100

 Spanning tree enabled protocol rstp

 Root ID Priority 24676

 Address e840.406f.7280

 Cost 31

 Port 7 (FastEthernet0/7)

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Bridge ID Priority 32868 (priority 32768 sys-id-ext 100)

 Address 5017.ff84.0a80

 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

 Aging Time 300 sec

Interface Role Sts Cost Prio.Nbr Type

------------------- ---- --- --------- -------- --------------------------------

Fa0/7 Root FWD 12 128.7 P2p

Fa0/8 Altn BLK 19 128.8 P2p

Fa0/11 Altn BLK 19 128.11 P2p

Fa0/12 Altn BLK 19 128.12 P2p

Here ALS2 tells us that its path to the root is via Fa0/7 (connected to DLS2) with a total cost of 31, while Fa0/11 and 12 (connected to ALS1) are Alternate ports. Fa0/11 and Fa0/12 are alternate ports because the interface cost plus the cost advertised by ALS1 equals 39, which is greater than the local interface cost plus the cost advertised by DLS2. Fa0/7 has a locally configured cost of 12. That plus the 19 advertised by DLS2 equals 31. You can see these details in the output of **show spanning-tree detail**

Configure Loop Guard on ALS2s Fa0/11 interface:

ALS2(config)# **int f0/11**

ALS2(config-if)# **spanning-tree guard loop**

ALS2(config-if)# **exit**

Modify the corresponding interface (Fa0/11) on ALS1 to stop sending BPDUs:

ALS1(config)# **int f0/11**

ALS1(config-if)# **spanning-tree bpdufilter enable**

ALS1(config-if)# **exit**

Shortly after doing this, you should receive the following SYSLOG message on ALS2 for every VLAN that had Fa0/11 as an Alternate port:

ALS2#

\*Mar 1 03:22:36.795: %SPANTREE-2-LOOPGUARD\_BLOCK: Loop guard blocking port FastEthernet0/11 on VLAN0100.

\*Mar 1 03:22:37.802: %SPANTREE-2-LOOPGUARD\_BLOCK: Loop guard blocking port FastEthernet0/11 on VLAN0099.

ALS2# **show spanning-tree inconsistentports**

Name Interface Inconsistency

-------------------- ------------------------ ------------------

VLAN0099 FastEthernet0/11 Loop Inconsistent

VLAN0100 FastEthernet0/11 Loop Inconsistent

Number of inconsistent ports (segments) in the system : 2

ALS2#

Fix this by reversing the configuration at ALS1 and verifying at ALS2:

ALS1(config)# **int f0/11**

ALS1(config-if)# **spanning-tree bpdufilter disable**

ALS1(config-if)# **exit**

\*Mar 1 03:23:37.227: %SPANTREE-2-LOOPGUARD\_UNBLOCK: Loop guard unblocking port FastEthernet0/11 on VLAN0099.

ALS2# **show spanning-tree inconsistentports**

Name Interface Inconsistency

-------------------- ------------------------ ------------------

Number of inconsistent ports (segments) in the system : 0

* 1. End of Lab

Do not save your configurations. The equipment will be reset for the next lab.