Part II of this book introduces the basics of Ethernet LANs, both in concept and in how to implement the features. However, the two primary features discussed in Part III of this book—Virtual LANs (VLANs) and Spanning Tree Protocol (STP)—impact almost everything you have learned about Ethernet so far.

VLANs allow a network engineer to create separate Ethernet LANs through simple configuration choices. The ability to separate some switch ports into one VLAN and other switch ports into another VLAN gives network designers a powerful tool for creating networks. Once created, VLANs also have a huge impact on how a switch works, which then impacts how you verify and troubleshoot the operation of a campus LAN.

The two VLAN-related exam topics (2.1 and 2.2) use the verbs *configure* and *verify*. To support that depth, Chapter 8 opens this part with details about the concepts related to VLANs, VLAN trunks, and EtherChannels. It then shows a variety of configuration options, sprinkled with some troubleshooting topics.

STP acts to prevent frames from looping repeatedly around a LAN that has redundant links. Without STP, switches would forward broadcasts and some other frames around and around the LAN, eventually clogging the LAN so much as to make it unusable.

The CCNA 200-301 version 1.1 exam blueprint refers to STP using only topic 2.5. That topic uses the plural phrase *Spanning Tree Protocols*, in reference to the original Spanning Tree Protocol (STP) and its replacement Rapid STP (RSTP). That exam topic uses the verb *identify*, which places more importance on interpreting STP operations using **show** commands rather than every configuration option.

As for this part of the book, Chapter 9 focuses on STP concepts, with Chapter 10 focusing on configuration and verification—but with emphasis on interpreting STP behavior. It also discusses Layer 2 EtherChannel configuration.

Part III

Implementing VLANs and STP

CHAPTER 8

Implementing Ethernet Virtual LANs

This chapter covers the following exam topics:

- **1.0 Network Fundamentals**
  - **1.13 Describe switching concepts**
    - **1.13.a MAC learning and aging**
    - **1.13.b Frame switching**
    - **1.13.c Frame flooding**
    - **1.13.d MAC address table**
- **2.0 Network Access**
  - **2.1 Configure and verify VLANs (normal range) spanning multiple switches**
    - **2.1.a Access ports (data and voice)**
    - **2.1.b Default VLAN**
    - **2.1.c InterVLAN connectivity**
  - **2.2 Configure and verify interswitch connectivity**
    - **2.2.a Trunk ports**
    - **2.2.b 802.1Q**
    - **2.2.c Native VLAN**

So far in this book, you have learned that Ethernet switches receive Ethernet frames, make decisions, and then forward (switch) those Ethernet frames. That core logic revolves around MAC addresses, the interface in which the frame arrives, and the interfaces out which the switch forwards the frame.

While true, that logic omits any consideration of virtual LANs (VLANs). VLANs impact the switching logic for each frame because each VLAN acts as a subset of the switch ports in an Ethernet LAN. Switches believe each Ethernet frame to be received in an identifiable VLAN, forwarded based on MAC table entries for that VLAN, and forwarded out ports in that VLAN. This chapter explores those concepts and others related to VLANs.

As for the organization of the chapter, the first major section of the chapter explains the core concepts. These concepts include how VLANs work on a single switch, how to use VLAN trunking to create VLANs that span across multiple switches, and how to forward traffic between VLANs using a router. The second major section shows how to configure VLANs and VLAN trunks: how to statically assign interfaces to a VLAN. The final major section discusses some issues that can arise when using VLANs and trunks and how to avoid those issues.

Also, this chapter goes beyond the normal chapter page length. If you want to break up your study into two reading sections, stop the first session when you reach the heading "Implementing Interfaces Connected to Phones."

"Do I Know This Already?" Quiz

Take the quiz (either here or use the PTP software) if you want to use the score to help you decide how much time to spend on this chapter. The letter answers are listed at the bottom of the page following the quiz. Appendix C, found both at the end of the book as well as on the companion website, includes both the answers and explanations. You can also find both answers and explanations in the PTP testing software.

Table 8-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
Virtual LAN Concepts	1–3
VLAN and VLAN Trunking Configuration and Verification	4–6
Troubleshooting VLANs and VLAN Trunks	7–8

1. In a LAN, which of the following terms best equates to the term *VLAN*?

- **a.** Collision domain
- **b.** Broadcast domain
- **c.** Subnet
- **d.** Single switch
- **e.** Trunk

2. Imagine a small lab network with one LAN switch, with the ports configured to be in three different VLANs. How many IP subnets do you need for the hosts connected to the switch ports, assuming that all hosts in all VLANs want to use TCP/IP?

- **a.** 0
- **b.** 1
- **c.** 2
- **d.** 3
- **e.** You cannot tell from the information provided.

3. Switch SW1 sends a frame to switch SW2 using 802.1Q trunking. Which of the answers describes how SW1 changes or adds to the Ethernet frame before forwarding the frame to SW2?

- **a.** Inserts a 4-byte header and does change the MAC addresses.
- **b.** Inserts a 4-byte header and does not change the MAC addresses.
- **c.** Encapsulates the original frame behind an entirely new Ethernet header.
- **d.** None of the other answers are correct.

4. You are told that switch 1 is configured with the **switchport mode dynamic auto** command on its Fa0/5 interface, which is connected to switch 2. Which of the following settings on the **switchport mode** command on switch 2 would allow trunking to work? (Choose two answers.)
   a. **trunk**
   b. **dynamic auto**
   c. **dynamic desirable**
   d. **access**
   e. None of the other answers are correct.
5. A switch has just arrived from Cisco. The switch has never been configured with any VLANs, but VTP has been disabled. An engineer configures the **vlan 22** and **name Hannahs-VLAN** commands and then exits configuration mode. Which of the following are true? (Choose two answers.)
   a. VLAN 22 is listed in the output of the **show vlan brief** command.
   b. VLAN 22 is listed in the output of the **show running-config** command.
   c. VLAN 22 is not created by this process.
   d. VLAN 22 does not exist in that switch until at least one interface is assigned to that VLAN.
6. Which of the following commands identify switch interfaces as being trunking interfaces—interfaces that currently operate as VLAN trunks? (Choose two answers.)
   a. **show interfaces**
   b. **show interfaces switchport**
   c. **show interfaces trunk**
   d. **show trunks**
7. In a switch that disables VTP, an engineer configures the commands **vlan 30** and **shutdown vlan 30**. Which answers should be true about this switch? (Choose two answers.)
   a. The **show vlan brief** command should list VLAN 30.
   b. The **show running-config** command should list VLAN 30.
   c. The switch should forward frames that arrive in access ports in VLAN 30.
   d. The switch should forward frames that arrive in trunk ports tagged with VLAN 30.
8. The **show interfaces g0/1 trunk** command provides three lists of VLAN IDs. Which items would limit the VLANs that appear in the first of the three lists of VLANs?
   a. A **shutdown vlan 30** global command
   b. A **switchport trunk allowed vlan** interface subcommand
   c. An STP choice to block on G0/1
   d. A **no vlan 30** global command

Foundation Topics

Virtual LAN Concepts

Before understanding **VLANs**, you must first have a specific understanding of the definition of a LAN. For example, from one perspective, a LAN includes all the user devices, servers, switches, routers, cables, and wireless access points in one location. However, an alternative narrower definition of a LAN can help in understanding the concept of a virtual LAN:

A LAN includes all devices in the same broadcast domain.

A broadcast domain includes the set of all LAN-connected devices, so that when any of the devices sends a broadcast frame, all the other devices get a copy of the frame. So, from one perspective, you can think of a LAN and a broadcast domain as being basically the same thing.

Using only default settings, a switch considers all its interfaces to be in the same broadcast domain. That is, for one switch, when a broadcast frame entered one switch port, the switch forwarded that broadcast frame out all other ports. With that logic, to create two different LAN broadcast domains, you had to buy two different Ethernet LAN switches, as shown in Figure 8-1.

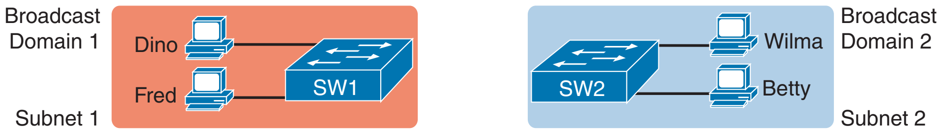


Figure 8-1 *Creating Two Broadcast Domains with Two Physical Switches and No VLANs*

By using two VLANs, a single switch can accomplish the same goals of the design in Figure 8-1—to create two broadcast domains—with a single switch. With VLANs, a switch can configure some interfaces into one broadcast domain and some into another, creating multiple broadcast domains. These individual broadcast domains created by the switch are called *virtual LANs (VLANs)*.

For example, in Figure 8-2, the single switch creates two VLANs, treating the ports in each VLAN as being completely separate. The switch would never forward a frame sent by Dino (in VLAN 1) over to either Wilma or Betty (in VLAN 2).

Key Topic

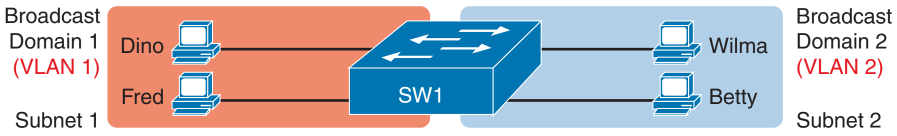


Figure 8-2 *Creating Two Broadcast Domains Using One Switch and VLANs*

Designing campus LANs to use more VLANs, each with a smaller number of devices, often helps improve the LAN in many ways. For example, a broadcast sent by one host in a VLAN will be received and processed by all the other hosts in the VLAN—but not by hosts in a different VLAN. Limiting the number of hosts that receive a single broadcast frame reduces the number of hosts that waste effort processing unneeded broadcasts. It also reduces security risks because fewer hosts see frames sent by any one host. These are just a few reasons for separating hosts into different VLANs. The following list summarizes the most common reasons for choosing to create smaller broadcast domains (VLANs):

Key Topic

- To reduce CPU overhead on each device, improving host performance, by reducing the number of devices that receive each broadcast frame
- To reduce security risks by reducing the number of hosts that receive copies of frames that the switches flood (broadcasts, multicasts, and unknown unicasts)
- To improve security for hosts through the application of different security policies per VLAN
- To create more flexible designs that group users by department, or by groups that work together, instead of by physical location
- To solve problems more quickly, because the failure domain for many problems is the same set of devices as those in the same broadcast domain
- To reduce the workload for the Spanning Tree Protocol (STP) by limiting a VLAN to a single access switch

The rest of this chapter looks closely at the mechanics of how VLANs work across multiple Cisco switches, including the required configuration. To that end, the next section examines VLAN trunking, a feature required when installing a VLAN that exists on more than one LAN switch.

Creating Multiswitch VLANs Using Trunking

Configuring VLANs on a single switch requires only a little effort: you simply configure each port to tell it the VLAN number to which the port belongs. With multiple switches, you have to consider additional concepts about how to forward traffic between the switches.

When you are using VLANs in networks that have multiple interconnected switches, the switches need to use *VLAN trunking* on the links between the switches. VLAN trunking causes the switches to use a process called *VLAN tagging*, by which the sending switch adds another header to the frame before sending it over the **trunk**. This extra trunking header includes a *VLAN identifier* (VLAN ID) field so that the sending switch can associate the frame with a particular VLAN ID, and the receiving switch can then know in what VLAN each frame belongs.

Figure 8-3 shows an example that demonstrates VLANs that exist on multiple switches, but it does not use trunking. First, the design uses two VLANs: VLAN 10 and VLAN 20. Each switch has two ports assigned to each VLAN, so each VLAN exists in both switches. To forward traffic in VLAN 10 between the two switches, the design includes a link between switches, with that link fully inside VLAN 10. Likewise, to support VLAN 20 traffic between switches, the design uses a second link between switches, with that link inside VLAN 20.

Answers to the "Do I Know This Already?" quiz:

1 B **2** D **3** B **4** A, C **5** A, B **6** B, C **7** A, B **8** B

The design in Figure 8-3 functions perfectly. For example, PC11 (in VLAN 10) can send a frame to PC14. The frame flows into SW1, over the top link (the one that is in VLAN 10) and over to SW2.

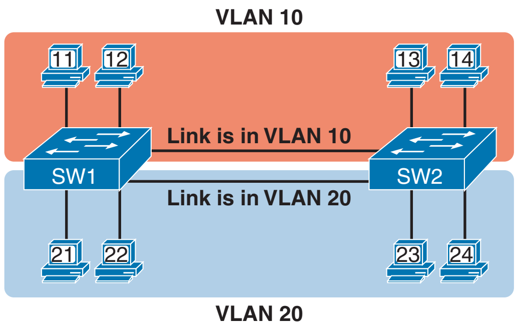


Figure 8-3 *Multiswitch VLAN Without VLAN Trunking*

The design shown in Figure 8-3 works, but it simply does not scale very well. It requires one physical link between switches to support every VLAN. If a design needed 10 or 20 VLANs, you would need 10 or 20 links between switches, and you would use 10 or 20 switch ports (on each switch) for those links.

VLAN Tagging Concepts

VLAN trunking creates one link between switches that supports as many VLANs as you need. As a VLAN trunk, the switches treat the link as if it were a part of all the VLANs. At the same time, the trunk keeps the VLAN traffic separate, so frames in VLAN 10 would not go to devices in VLAN 20, and vice versa, because each frame is identified by VLAN number as it crosses the trunk. Figure 8-4 shows the idea, with a single physical link between the two switches.

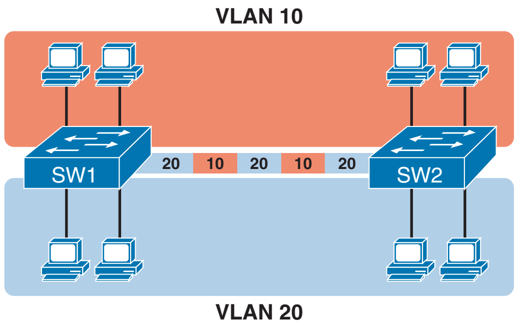


Figure 8-4 *Multiswitch VLAN with Trunking*

The use of trunking allows switches to forward frames from multiple VLANs over a single physical connection by adding a small header to the Ethernet frame. For example, Figure 8-5 shows PC11 sending a broadcast frame on interface Fa0/1 at Step 1. To flood the frame, switch SW1 needs to forward the broadcast frame to switch SW2. However, SW1 needs to let SW2 know that the frame is part of VLAN 10, so that after the frame is received, SW2 will flood the frame only into VLAN 10, and not into VLAN 20. So, as shown at Step 2, before sending the frame, SW1 adds a VLAN header to the original Ethernet frame, with the VLAN header listing a VLAN ID of 10 in this case.

Key Topic

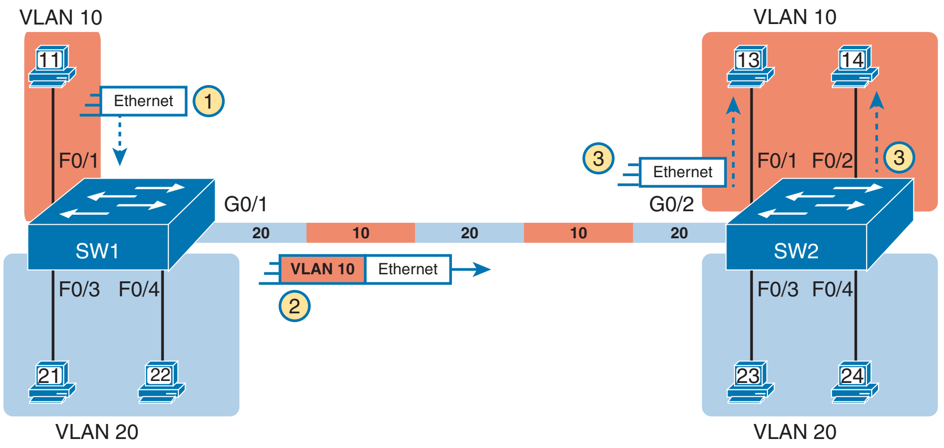


Figure 8-5 *VLAN Trunking Between Two Switches*

When SW2 receives the frame, it understands that the frame is in VLAN 10. SW2 then removes the VLAN header, forwarding the original frame out its interfaces in VLAN 10 (Step 3).

For another example, consider the case when PC21 (in VLAN 20) sends a broadcast. SW1 sends the broadcast out port Fa0/4 (because that port is in VLAN 20) and out Gi0/1 (because it is a trunk, meaning that it supports multiple different VLANs). SW1 adds a trunking header to the frame, listing a VLAN ID of 20. SW2 strips off the trunking header after determining that the frame is part of VLAN 20, so SW2 knows to forward the frame out only ports Fa0/3 and Fa0/4, because they are in VLAN 20, and not out ports Fa0/1 and Fa0/2, because they are in VLAN 10.

The 802.1Q and ISL VLAN Trunking Protocols

Cisco has supported two different trunking protocols over the years: Inter-Switch Link (ISL) and IEEE **802.1Q**. Cisco created the ISL years before 802.1Q, in part because the IEEE had not yet defined a VLAN trunking standard. Today, 802.1Q has become the more popular trunking protocol, with Cisco not even bothering to support ISL in many of its switch models today.

While both ISL and 802.1Q tag each frame with the VLAN ID, the details differ. 802.1Q inserts an extra 4-byte 802.1Q VLAN header into the original frame's Ethernet header, as shown at the top of Figure 8-6. As for the fields in the 802.1Q header, only the 12-bit VLAN ID field inside the 802.1Q header matters for topics discussed in this book. This 12-bit field supports a theoretical maximum of 212 (4096) VLANs, but in practice it supports a maximum of 4094. (Both 802.1Q and ISL use 12 bits to tag the VLAN ID, with two reserved values [0 and 4095].)

Cisco switches break the range of VLAN IDs (1–4094) into two ranges: the normal range and the extended range. All switches can use normal-range VLANs with values from 1 to 1005. Only some switches can use extended-range VLANs with VLAN IDs from 1006 to 4094. The rules for which switches can use extended-range VLANs depend on the configuration of the VLAN Trunking Protocol (**VTP**), which is discussed briefly in the section "VLAN Trunking Configuration," later in this chapter.

Key Topic

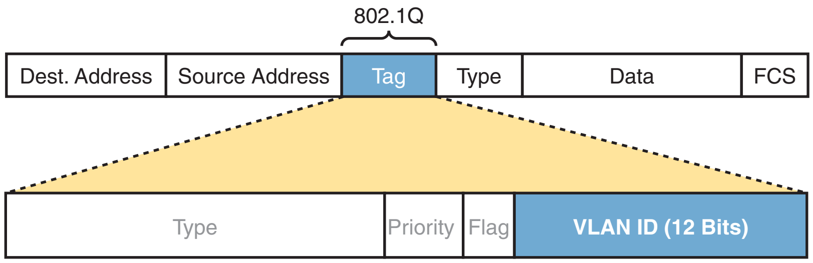


Figure 8-6 *802.1Q Trunking*

802.1Q also defines one special VLAN ID on each trunk as the **native VLAN** (defaulting to use VLAN 1). In a normally working 802.1Q trunk, both endpoints use trunking, and both use the same native VLAN. Neither end, when sending a frame assigned to this native VLAN, adds the 802.1Q header. Both switches, knowing that untagged frames mean that the frame is part of the native VLAN, treat untagged frames as being part of the native VLAN.

The IEEE included the native VLAN concept for cases in which one device operates as a trunk while the other side does not. The nontrunking endpoint may do so temporarily or permanently (typically because of misconfiguration or a lack of 802.1Q support). In such cases, the nontrunking device will be confused and discard any received frames that contain a trunking header. However, when the trunking side sends frames as part of the native VLAN—untagged—the nontrunking side will understand the frame and consider it to be part of the access VLAN assigned to the interface. So, the native VLAN gives engineers a tool to allow for cases of making one VLAN work over the link, even when some trunking issue might exist.

Forwarding Data Between VLANs

If you create a campus LAN that contains many VLANs, you typically still need all devices to be able to send data to all other devices. This next topic discusses some concepts about how to route data between those VLANs.

The Need for Routing Between VLANs

LAN switches that forward data based on Layer 2 logic, as discussed so far in this book, often go by the name *Layer 2 switch*. For example, Chapter 5, "Analyzing Ethernet LAN Switching," discussed how LAN switches receive Ethernet frames (a Layer 2 concept), look at the destination Ethernet MAC address (a Layer 2 address), and forward the Ethernet frame out some other interface. All those concepts are defined by Layer 2 protocols, hence the name Layer 2 switch.

Layer 2 switches perform their logic per VLAN. For example, in Figure 8-7, the two PCs on the left sit in VLAN 10, in subnet 10. The two PCs on the right sit in a different VLAN (20), with a different subnet (20). Note that the figure repeats earlier Figure 8-2, but with the switch broken into halves, to emphasize the point that Layer 2 switches will not forward data between two VLANs.

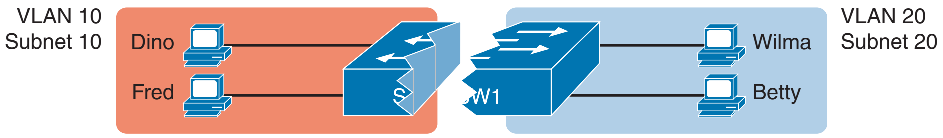


Figure 8-7 *Layer 2 Switch Does Not Route Between the VLANs*

As shown in the figure, when configured with some ports in VLAN 10 and others in VLAN 20, the switch acts like two separate switches in which it will forward traffic. In fact, one goal of VLANs is to separate traffic in one VLAN from another, preventing frames in one VLAN from leaking over to other VLANs. For example, when Dino (in VLAN 10) sends any Ethernet frame, if SW1 is a Layer 2 switch, that switch will not forward the frame to the PCs on the right in VLAN 20.

Routing Packets Between VLANs with a Router

When VLANs are included in a campus LAN design, the devices in a VLAN need to be in the same subnet. Following the same design logic, devices in different VLANs need to be in different subnets.

To forward packets between VLANs, the network must use a device that acts as a router. You can use an actual router or use a switch that can perform some functions like a router. These switches that also perform Layer 3 routing functions go by the name *multilayer switch* or **Layer 3 switch**. This section first discusses how to forward data between VLANs when using Layer 2 switches and ends with a brief discussion of how to use Layer 3 switches.

For example, Figure 8-8 shows a router that can route packets between subnets 10 and 20. The figure shows the same Layer 2 switch as shown in Figure 8-7, with the same perspective of the switch being split into parts with two different VLANs, and with the same PCs in the same VLANs and subnets. Now Router R1 has one LAN physical interface connected to the switch and assigned to VLAN 10, and a second physical interface connected to the switch and assigned to VLAN 20. With an interface connected to each subnet, the Layer 2 switch can keep doing its job—forwarding frames inside a VLAN, while the router can do its job—routing IP packets between the subnets.

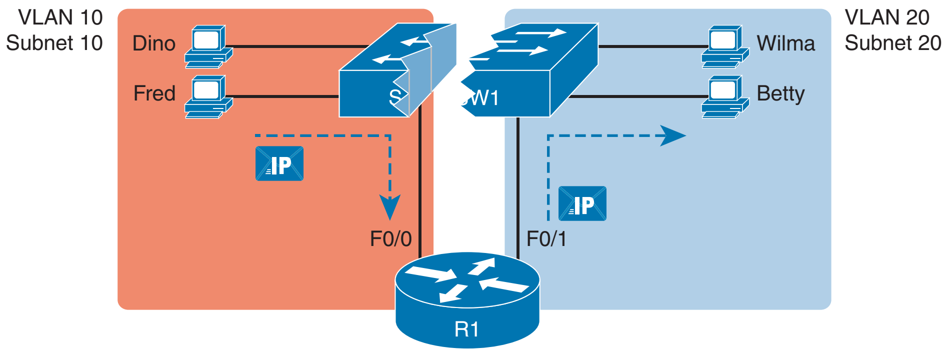


Figure 8-8 *Routing Between Two VLANs on Two Physical Interfaces*

The figure shows an IP packet being routed from Fred, which sits in one VLAN/subnet, to Betty, which sits in the other. The Layer 2 switch forwards two different Layer 2 Ethernet frames: one in VLAN 10, from Fred to R1's F0/0 interface, and the other in VLAN 20, from R1's F0/1 interface to Betty. From a Layer 3 perspective, Fred sends the IP packet to its default router (R1), and R1 routes the packet out another interface (F0/1) into another subnet where Betty resides.

The design in Figure 8-8 works, but there are several different solutions for routing packets between VLANs. This chapter shows the option of using a separate physical router, with a separate link per VLAN, because it can be the easiest of the options to understand and visualize. Chapter 18, "IP Routing in the LAN," works through those other features for routing packets between VLANs.

VLAN and VLAN Trunking Configuration and Verification

Cisco switches do not require any configuration to work. You can purchase Cisco switches, install devices with the correct cabling, turn on the switches, and they work. You would never need to configure the switch, and it would work fine, even if you interconnected switches, until you needed more than one VLAN. But if you want to use VLANs—and most enterprise networks do—you need to add some configuration.

This chapter separates the VLAN configuration details into two major sections. The first section looks at how to configure *static access interfaces*—switch interfaces configured to be in one VLAN only, therefore not using VLAN trunking. The second part shows how to configure interfaces that do use VLAN trunking.

Creating VLANs and Assigning Access VLANs to an Interface

This section shows how to create a VLAN, give the VLAN a name, and assign interfaces to a VLAN. To focus on these basic details, this section shows examples using a single switch, so VLAN trunking is not needed.

For a Cisco switch to forward frames in a particular VLAN, the switch must be configured to believe that the VLAN exists. In addition, the switch must have nontrunking interfaces (called **access interfaces** or **static access interfaces**) assigned to the VLAN and/or trunks that support the VLAN. The configuration steps for access interfaces are as follows:

Config Checklist

Step 1. To configure a new VLAN, follow these steps:

- **a.** From configuration mode, use the **vlan** *vlan-id* command in global configuration mode to create the VLAN and to move the user into VLAN configuration mode.
- **b.** (Optional) Use the **name** *name* command in VLAN configuration mode to list a name for the VLAN. If not configured, the VLAN name is VLAN*ZZZZ*, where *ZZZZ* is the four-digit decimal VLAN ID.

Step 2. For each access interface, follow these steps:

- **a.** Use the **interface** *type number* command in global configuration mode to move into interface configuration mode for each desired interface.
- **b.** Use the **switchport access vlan** *id-number* command in interface configuration mode to specify the VLAN number associated with that interface.
- **c.** (Optional) Use the **switchport mode access** command in interface configuration mode to make this port always operate in access mode (that is, to not trunk).

While the list might look a little daunting, the process on a single switch is actually pretty simple. For example, if you want to put the switch's ports in three VLANs—11, 12, and 13—you first add three **vlan** commands: **vlan 11**, **vlan 12**, and **vlan 13**. Then, for each interface, add a **switchport access vlan 11** (or **12** or **13**) command to assign that interface to the proper VLAN.

NOTE The term **default VLAN** (as shown in the exam topics) refers to the default setting on the **switchport access vlan** *vlan-id* command, and that default is VLAN ID 1. In other words, by default, each port is assigned to access VLAN 1.

VLAN Configuration Example 1: Full VLAN Configuration

Examples 8-1, 8-2, and 8-3 work through one scenario with VLAN configuration and verification. To begin, Example 8-1 begins by showing the VLANs in switch SW1 in Figure 8-9, with all default settings related to VLANs.

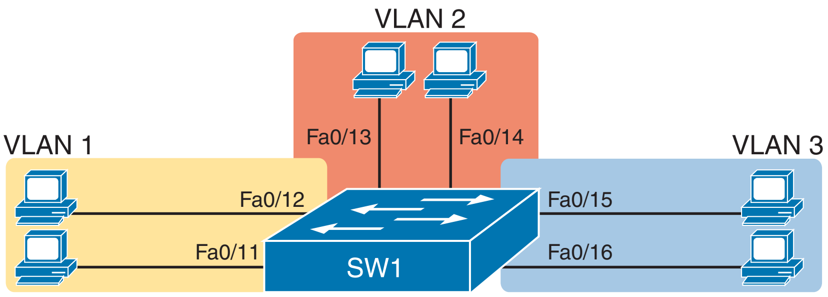


Figure 8-9 *Network with One Switch and Three VLANs*

Example 8-1 *Configuring VLANs and Assigning VLANs to Interfaces*

```
SW1# show vlan brief
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                 Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                 Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                 Gi0/1, Gi0/2
1002 fddi-default                      act/unsup
1003 token-ring-default                act/unsup
1004 fddinet-default                   act/unsup
1005 trnet-default                     act/unsup
```

The example begins with the **show vlan brief** command, confirming the default settings of five nondeletable VLANs, with all interfaces assigned to VLAN 1. VLAN 1 cannot be deleted but can be used. VLANs 1002–1005 cannot be deleted and cannot be used as access VLANs today. In particular, note that this switch has 24 Fast Ethernet ports (Fa0/1–Fa0/24) and two Gigabit Ethernet ports (Gi0/1 and Gi0/2), all of which are listed as being in VLAN 1 per that first command's output, confirming that by default, Cisco switches assign all ports to VLAN 1.

Next, Example 8-2 shows steps that mirror the VLAN configuration checklist, namely the configuration of VLAN 2, plus the assignment of VLAN 2 as the access VLAN on two ports: Fa0/13 and Fa0/14.

Example 8-2 *Configuring VLANs and Assigning VLANs to Interfaces*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# vlan 2
SW1(config-vlan)# name Freds-vlan
SW1(config-vlan)# exit
SW1(config)# interface range fastethernet 0/13 - 14
SW1(config-if)# switchport access vlan 2
SW1(config-if)# switchport mode access
SW1(config-if)# end

SW1# show vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/15, Fa0/16, Fa0/17, Fa0/18
                                                Fa0/19, Fa0/20, Fa0/21, Fa0/22
                                                Fa0/23, Fa0/24, Gi0/1, Gi0/2
2    Freds-vlan                       active    Fa0/13, Fa0/14
1002 fddi-default                  act/unsup
1003 token-ring-default            act/unsup
1004 fddinet-default               act/unsup
1005 trnet-default                 act/unsup
```

Take a moment to compare the output of the **show vlan brief** commands in Example 8-2 (after adding the configuration) versus Example 8-1. Example 8-2 shows new information about VLAN 2, with ports Fa0/13 and Fa0/14 no longer being listed with VLAN 1, but now listed as assigned to VLAN 2.

To complete this scenario, Example 8-3 shows a little more detail about the VLAN itself. First, the **show running-config** command lists both the **vlan 2** and **switchport access vlan 2** commands as configured in Example 8-2. Also, note that earlier Example 8-2 uses the **interface range** command, with one instance of the **switchport access vlan 2** interface subcommand. However, Example 8-3 shows how the switch actually applied that command to both Fa0/13 and Fa0/14. Example 8-3 ends with the **show vlan id 2** command, which confirms the operational status that ports Fa0/13 and Fa0/14 are assigned to VLAN 2.

Example 8-3 *Configuring VLANs and Assigning VLANs to Interfaces*

```
SW1# show running-config
! Many lines omitted for brevity
! Early in the output:
vlan 2
 name Freds-vlan
!
! more lines omitted for brevity
interface FastEthernet0/13
 switchport access vlan 2
 switchport mode access
!
interface FastEthernet0/14
 switchport access vlan 2
 switchport mode access
!

SW1# show vlan id 2
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
2    Freds-vlan                       active    Fa0/13, Fa0/14

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
2    enet  100010     1500  -      -      -        -    -        0      0

Remote SPAN VLAN
----------------
Disabled

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
```

The example surrounding Figure 8-9 uses six switch ports, all of which need to operate as access ports. That is, each port should not use trunking but instead should be assigned to a single VLAN, as assigned by the **switchport access vlan** *vlan-id* command. For ports that should always act as access ports, add the optional interface subcommand **switchport mode access.** This command tells the switch to always be an access interface and disables the protocol that negotiates trunking (Dynamic Trunking Protocol [DTP]) with the device on the other end of the link. (The upcoming section "VLAN Trunking Configuration" discusses more details about the commands that allow a port to negotiate whether it should use trunking.)

NOTE The companion website for this book includes a video that works through a different VLAN configuration example.

VLAN Configuration Example 2: Shorter VLAN Configuration

Example 8-2 shows how to configure a VLAN and add two ports to the VLAN as access ports. Example 8-4 does the same, this time with VLAN 3, and this time with a much briefer alternative configuration. The configuration completes the configuration of the design shown in Figure 8-9, by adding two ports to VLAN 3.

Example 8-4 *Shorter VLAN Configuration Example (VLAN 3)*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# interface range Fastethernet 0/15 - 16
SW1(config-if-range)# switchport access vlan 3
% Access VLAN does not exist. Creating vlan 3
SW1(config-if-range)# ^Z

SW1# show vlan brief

VLAN Name                              Status     Ports
---- -------------------------------- --------- -------------------------------
1 default                            active      Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                 Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                 Gi0/1, Gi0/2
2 Freds-vlan                        active      Fa0/13, Fa0/14
3 VLAN0003                          active      Fa0/15, Fa0/16
1002 fddi-default                   act/unsup
1003 token-ring-default             act/unsup
1004 fddinet-default                act/unsup
1005 trnet-default                  act/unsup
```

Example 8-4 shows how a switch can dynamically create a VLAN—the equivalent of the **vlan** *vlan-id* global config command—when the **switchport access vlan** interface subcommand refers to a currently unconfigured VLAN. This example begins with SW1 not knowing about VLAN 3. With the addition of the **switchport access vlan 3** interface subcommand, the switch realized that VLAN 3 did not exist, and as noted in the shaded message in the example, the switch created VLAN 3, using a default name (VLAN0003). The engineer did not need to type the **vlan 3** global command to create VLAN 3; the switch did that automatically. No other steps are required to create the VLAN. At the end of the process, VLAN 3 exists in the switch, and interfaces Fa0/15 and Fa0/16 are in VLAN 3, as noted in the shaded part of the **show vlan brief** command output.

VLAN Trunking Protocol

Before showing more configuration examples, you also need to know something about a Cisco protocol and tool called the VLAN Trunking Protocol (VTP). VTP is a Cisco proprietary tool on Cisco switches that advertises each VLAN configured in one switch (with the **vlan** *number* command) so that all the other switches in the campus learn about that VLAN.

This book does not discuss VTP as an end to itself for a few different reasons. First, the current CCNA 200-301 exam blueprint ignores VTP, as do the CCNP Enterprise Core and CCNP Enterprise Advanced Routing blueprints. Additionally, many enterprises choose to disable VTP.

Also, you can easily disable VTP so that it has no impact on your switches in the lab, which is exactly what I did when building all the examples in this book.

However, VTP has some small impact on how every Cisco Catalyst switch works, even if you avoid using VTP. This brief section introduces enough details of VTP so that you can understand the impact of VTP in a Cisco Catalyst switch.

First, all examples in this book (and in Volume 2) use switches that disable VTP in some way. Interestingly, for much of VTP's decades of existence, most switches did support an option to completely disable VTP. Instead, to effectively disable VTP, the engineer would set the switch to use **VTP transparent mode** (with the **vtp mode transparent** global command). Many newer switches now have an option to disable VTP completely with the **vtp mode off** global command. For the purposes of this book, configuring a switch with either transparent mode or off mode disables VTP.

Note that both transparent and off modes prevent VTP from learning and advertising about VLAN configuration. Those modes allow a switch to configure all VLANs, including standard- and extended-range VLANs. Additionally, switches using transparent or off modes list the **vlan** configuration commands in the running-config file.

In contrast, switches in VTP server or client mode behave differently. Just in case you do lab exercises with real switches or with simulators, and you see unusual results with VLANs, check the VTP status with the **show vtp status** command. If your switch uses VTP server or client mode, you will find

- The server switches can configure VLANs in the standard range only (1–1005).
- The client switches cannot configure VLANs.
- Both servers and clients may be learning new VLANs from other switches and seeing their VLANs deleted by other switches because of VTP.
- The **show running-config** command does not list any **vlan** commands; you must use other **show** commands to find out about the configured VLANs.

If possible in the lab, disable VTP for your switch configuration practice until you decide to learn more about VTP for other purposes.

NOTE Do not change VTP settings on any switch that also connects to the production network until you know how VTP works and you talk with experienced colleagues. Doing so can cause real harm to your LAN. For example, if the switch you configure connects to other switches, which in turn connect to switches used in the production LAN, you could accidentally change the VLAN configuration in other switches with serious impact to the operation of the network. You could delete VLANs and cause outages. Be careful and never experiment with VTP settings on a switch unless it and the other switches connected to it have absolutely no physical links connected to the production LAN.

VLAN Trunking Configuration

Trunking configuration between two Cisco switches can be simple if you just statically configure trunking. You could literally add one interface subcommand for the switch interface on each side of the link (**switchport mode trunk**), and you would create a VLAN trunk that supported all the VLANs known to each switch.

However, trunking configuration on Cisco switches includes many more options, including several options for dynamically negotiating various trunking settings. The configuration can either predefine different settings or tell the switch to negotiate the settings, as follows:

- **The type of trunking:** IEEE 802.1Q, ISL, or negotiate which one to use, on switches that support both types of trunking.
- **The administrative mode:** Whether to always trunk, always not trunk, or negotiate whether to trunk or not.

First, consider the type of trunking. For many years, Cisco has not bothered to include ISL support in new switch product families, preferring IEEE 802.1Q. On older switches that support both, use the **switchport trunk encapsulation** {**dot1q** | **isl** | **negotiate**} interface subcommand to either configure the type to use or to allow Dynamic Trunking Protocol (DTP) to negotiate the type (which prefers ISL if both support it).

DTP can also negotiate whether the two devices on the link agree to trunk at all, as guided by the local switch port's administrative mode. The **administrative mode** refers to the configuration setting for whether trunking should be used. Cisco switches use the **switchport mode** interface subcommand to define the administrative trunking mode, as listed in Table 8-2. The switch interface's **operational mode** shows whether the interface operates as a trunk or an access port, which depends on the configuration on both switches plus other factors.

Key Topic

Table 8-2 Trunking Administrative Mode Options with the **switchport mode** Command

Command Option	Description
access	Always act as an access (nontrunk) port
trunk	Always act as a trunk port
dynamic desirable	Initiates negotiation messages and responds to negotiation messages to dynamically choose whether to start using trunking
dynamic auto	Passively waits to receive trunk negotiation messages, at which point the switch will respond and negotiate whether to use trunking

For example, consider the two switches shown in Figure 8-10. This figure expands the design shown earlier in Figure 8-9, with a trunk to a new switch (SW2) and with parts of VLANs 1 and 3 on ports attached to SW2. The two switches use a Gigabit Ethernet link for the trunk. In this case, the trunk does not dynamically form by default because both switches default to an administrative mode of *dynamic auto*, meaning that neither switch initiates the trunk negotiation process. When one switch is changed to use *dynamic desirable* mode, which does initiate the negotiation, the switches negotiate to use trunking, specifically 802.1Q because the switches support only 802.1Q.

Example 8-5 begins with SW1 configured as shown in Examples 8-2 and 8-4; that is, SW1 has two ports each assigned to VLANs 1, 2, and 3. However, both SW1 and SW2 currently have all default settings on the interfaces that connect the two switches. With the default setting of **switchport mode dynamic auto**, the two switches do not trunk.

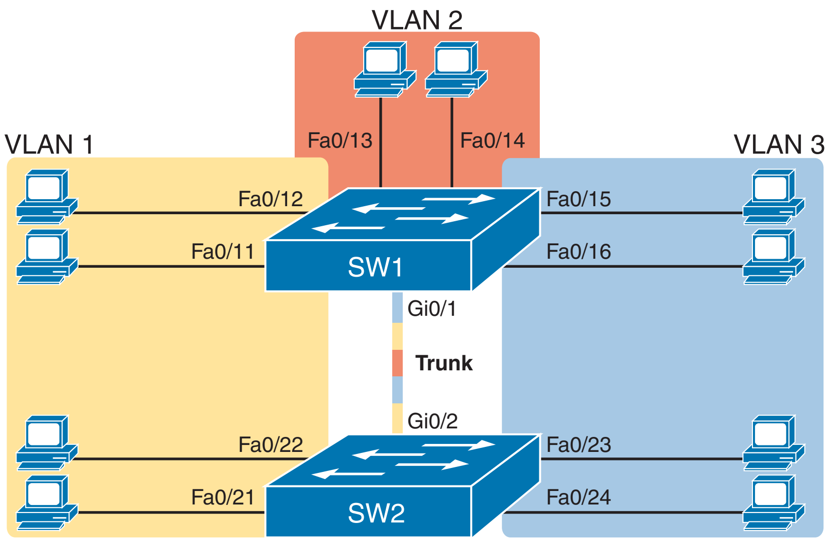


Figure 8-10 *Network with Two Switches and Three VLANs*

Example 8-5 *Initial (Default) State: Not Trunking Between SW1 and SW2*

```
SW1# show interfaces gigabit 0/1 switchport
Name: Gi0/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: native
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk private VLANs: none
Operational private-vlan: none
```

```
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Protected: false
Unknown unicast blocked: disabled
Unknown multicast blocked: disabled
Appliance trust: none

! Note that the next command results in a single empty line of output.
SW1# show interfaces trunk
SW1#
```

First, focus on the highlighted items from the output of the **show interfaces switchport** command at the beginning of Example 8-5. The output lists the default administrative mode setting of dynamic auto. Because SW2 also defaults to dynamic auto, the command lists SW1's operational status as "access," meaning that it is not trunking. ("Dynamic auto" tells both switches to sit there and wait on the other switch to start the negotiations.) The third shaded line points out the only supported type of trunking (802.1Q). (On a switch that supports both ISL and 802.1Q, this value would by default list "negotiate," to mean that the type of encapsulation is negotiated.) Finally, the operational trunking type is listed as "native," which is a reference to the 802.1Q native VLAN.

The end of the example shows the output of the **show interfaces trunk** command, but with no output. This command lists information about all interfaces that currently operationally trunk; that is, it lists interfaces that currently use VLAN trunking. With no interfaces listed, this command also confirms that the link between switches is not trunking.

Next, consider Example 8-6, which shows the new configuration that enables trunking. In this case, SW1 is configured with the **switchport mode dynamic desirable** command, which asks the switch to both negotiate as well as to begin the negotiation process, rather than waiting on the other device. The example shows that as soon as the command is issued, log messages appear showing that the interface goes down and then back up again, which happens when the interface transitions from access mode to trunk mode.

Example 8-6 *SW1 Changes from Dynamic Auto to Dynamic Desirable*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# interface gigabit 0/1
SW1(config-if)# switchport mode dynamic desirable
SW1(config-if)# ^Z
SW1#
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to
down
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to
up
SW1# show interfaces gigabit 0/1 switchport
```

```
Name: Gi0/1
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
! lines omitted for brevity
```

Example 8-6 repeats the **show interfaces gi0/1 switchport** command seen in Example 8-5, but after configuring VLAN trunking, so this time the output shows that SW1's G0/1 interface now operates as a trunk. Note that the command still lists the administrative settings, which denote the configured values along with the operational settings, which list what the switch is currently doing. SW1 now claims to be in an operational mode of *trunk*, with an operational trunking encapsulation of dot1Q.

Example 8-7 now repeats the same **show interfaces trunk** command that showed no output at all back in Example 8-5. Now that SW1 trunks on its G0/1 port, the output in Example 8-7 lists G0/1, confirming that G0/1 is now operationally trunking. The next section discusses the meaning of the output of this command. Also, note that the end of the example repeats the **show vlan id 2** command; of note, it includes the trunk port G0/1 in the output because the trunk port can forward traffic in VLAN 2.

Example 8-7 *A Closer Look at SW1's G0/1 Trunk Port*

```
SW1# show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Gi0/1       desirable        802.1q         trunking      1

Port        Vlans allowed on trunk
Gi0/1       1-4094

Port        Vlans allowed and active in management domain
Gi0/1       1-3

Port        Vlans in spanning tree forwarding state and not pruned
Gi0/1       1-3

SW1# show vlan id 2
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
2    Freds-vlan                       active    Fa0/13, Fa0/14, G0/1
```

```
VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
2    enet  100010     1500  -      -      -        -    -        0      0

Remote SPAN VLAN
----------------


Disabled
Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
```

For the exams, you should be ready to interpret the output of the **show interfaces switchport** command, realize the administrative mode implied by the output, and know whether the link should operationally trunk based on those settings. Table 8-3 lists the combinations of the trunking administrative modes and the expected operational mode (trunk or access) resulting from the configured settings. The table lists the administrative mode used on one end of the link on the left and the administrative mode on the switch on the other end of the link across the top of the table.

Key Topic

Table 8-3 Expected Trunking Operational Mode Based on the Configured Administrative Modes

Administrative Mode	Access	Dynamic Auto	Trunk	Dynamic Desirable
access	Access	Access	Do Not Use*	Access
dynamic auto	Access	Access	Trunk	Trunk
trunk	Do Not Use*	Trunk	Trunk	Trunk
dynamic desirable	Access	Trunk	Trunk	Trunk

* When two switches configure a mode of "access" on one end and "trunk" on the other, problems occur. Avoid this combination.

8

Finally, before we leave the discussion of configuring trunks, Cisco recommends disabling trunk negotiation on most ports for better security. The majority of switch ports on most switches will be used to connect to users and configured with the command **switchport mode access**, which also disables DTP. For ports without the **switchport mode access** command—for instance, ports statically configured to trunk with the **switchport mode trunk** command—DTP still operates, but you can disable DTP negotiations altogether using the **switchport nonegotiate** interface subcommand.

Implementing Interfaces Connected to Phones

This next topic is strange, at least in the context of access links and trunk links. In the world of IP telephony, telephones use Ethernet ports to connect to an Ethernet network so they can use IP to send and receive voice traffic sent via IP packets. To make that work, the switch's Ethernet port acts like an access port, but at the same time, the port acts like a trunk in some ways. This last topic of the chapter works through those main concepts.

Data and Voice VLAN Concepts

Before IP telephony, a PC could sit on the same desk as a phone. The phone happened to use UTP cabling, with that phone connected to some voice device (often called a *voice switch* or a *private branch exchange [PBX]*). The PC, of course, connected using a unshielded twisted-pair (UTP) cable to the usual LAN switch that sat in the wiring closet—sometimes in the same wiring closet as the voice switch. Figure 8-11 shows the idea.

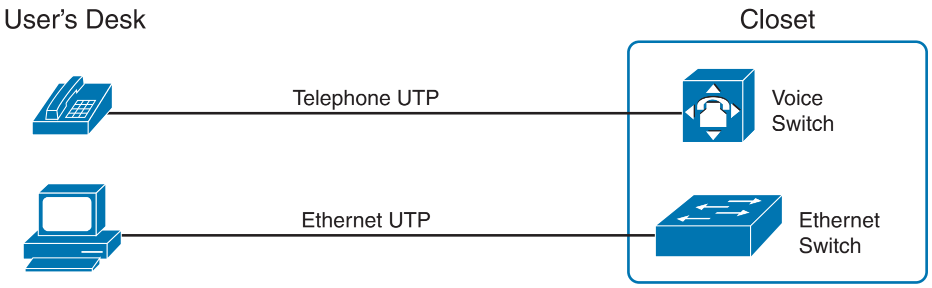


Figure 8-11 *Before IP Telephony: PC and Phone, One Cable Each, Connect to Two Different Devices*

The term *IP telephony* refers to the branch of networking in which the telephones use IP packets to send and receive voice as represented by the bits in the data portion of the IP packet. The phones connect to the network like most other end-user devices, using either Ethernet or Wi-Fi. These new IP phones did not connect via cable directly to a voice switch, instead connecting to the IP network using an Ethernet cable and an Ethernet port built in to the phone. The phones then communicated over the IP network with software that replaced the call setup and other functions of the PBX. (The current products from Cisco that perform this IP telephony control function are called *Cisco Unified Communication Manager*.)

The migration from using the already-installed telephone cabling to these new IP phones that needed UTP cables that supported Ethernet caused some problems in some offices. In particular:

- The older non-IP phones used a category of UTP cabling that often did not support 100-Mbps or 1000-Mbps Ethernet.
- Most offices had a single UTP cable running from the wiring closet to each desk, but now two devices (the PC and the new IP phone) both needed a cable from the desktop to the wiring closet.
- Installing a new cable to every desk would be expensive; plus you would need more switch ports.

To solve this problem, Cisco embedded small three-port switches into each phone.

IP telephones have included a small LAN switch, on the underside of the phone, since the earliest IP telephone products. Figure 8-12 shows the basic cabling, with the wiring closet cable connecting to one physical port on the embedded switch, the PC connecting with a short patch cable to the other physical port, and the phone's internal CPU connecting to an internal switch port.

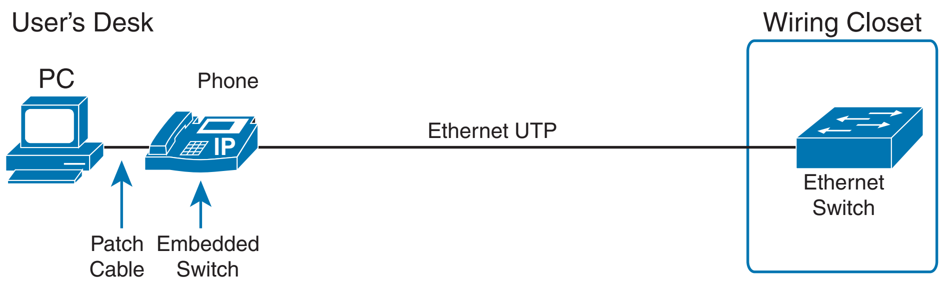


Figure 8-12 *Cabling with an IP Phone, a Single Cable, and an Integrated Switch*

Sites that use IP telephony, which includes most every company today, now have two devices off each access port. In addition, Cisco best practices for IP telephony design tell us to put the phones in one VLAN and the PCs in a different VLAN. To make that happen, the switch port acts a little like an access link (for the PC's traffic) and a little like a trunk (for the phone's traffic). The configuration defines two VLANs on that port, as follows:

Key Topic

Data VLAN: Same idea and configuration as the access VLAN on an access port but defined as the VLAN on that link for forwarding the traffic for the device connected to the phone on the desk (typically the user's PC).

Voice VLAN: The VLAN defined on the link for forwarding the phone's traffic. Traffic in this VLAN is typically tagged with an 802.1Q header.

Figure 8-13 illustrates this design with two VLANs on access ports that support IP telephones.

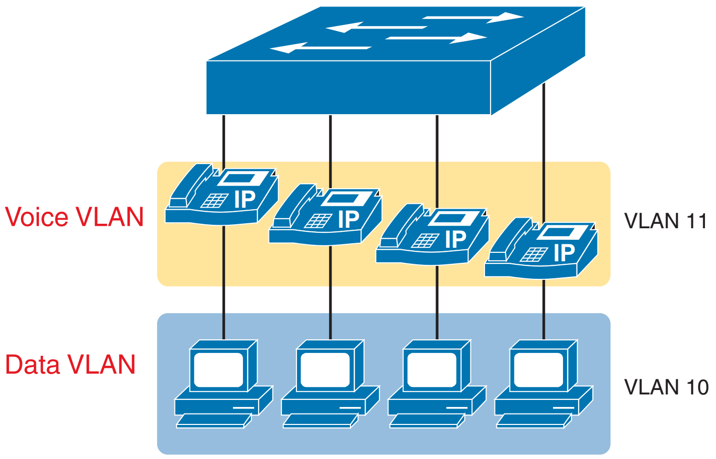


Figure 8-13 *A LAN Design, with Data in VLAN 10 and Phones in VLAN 11*

Data and Voice VLAN Configuration and Verification

Configuring a switch port to support IP phones, once you know the planned voice and data VLAN IDs, requires just a few easy commands. Making sense of the **show** commands once it is configured, however, can be a challenge. The port acts like an access port in many ways. However, with most configuration options, the voice frames flow with an 802.1Q header so that the link supports frames in both VLANs on the link. But that makes for some different **show** command output.

Example 8-8 shows an example configuration. In this case, all four switch ports F0/1–F0/4 begin with a default configuration. The configuration adds the new data and **voice VLANs**. The example then configures all four ports as access ports and defines the access VLAN, which is also called the **data VLAN** when discussing IP telephony. Finally, the configuration includes the **switchport voice vlan 11** command, which defines the voice VLAN used on the port. The example matches Figure 8-13, using ports F0/1–F0/4.

Example 8-8 *Configuring the Voice and Data VLAN on Ports Connected to Phones*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# vlan 10
SW1(config-vlan)# vlan 11
SW1(config-vlan)# interface range FastEthernet0/1 - 4
SW1(config-if)# switchport mode access
SW1(config-if)# switchport access vlan 10
SW1(config-if)# switchport voice vlan 11
SW1(config-if)#^Z
SW1#
```

NOTE Cisco phones exchange configuration data with the switch using the Cisco Discovery Protocol (CDP) or Link-Layer Discovery Protocol (LLDP). Switches enable CDP by default, so the configuration shows no specific configuration commands for it. *CCNA 200-301 Official Cert Guide, Volume 2*, Second Edition, Chapter 13, "Device Management Protocols," discusses CDP concepts and configuration.

The following list details the configuration steps for easier review and study:

Config Checklist

Step 1. Use the **vlan** *vlan-id* command in global configuration mode to create the data and voice VLANs if they do not already exist on the switch.

Step 2. Configure the data VLAN like an access VLAN, as usual:

- **a.** Use the **interface** *type number* command in global configuration mode to move into interface configuration mode.
- **b.** Use the **switchport access vlan** *id-number* command in interface configuration mode to define the data VLAN.
- **c.** Use the **switchport mode access** command in interface configuration mode to make this port always operate in access mode (that is, to not trunk).

Step 3. Use the **switchport voice vlan** *id-number* command in interface configuration mode to set the voice VLAN ID.

Verifying the status of a switch port configured like Example 8-8 shows some different output compared to the pure access port and pure trunk port configurations seen earlier in this chapter. For example, the **show interfaces switchport** command shows details about the operation of an interface, including many details about access ports. Example 8-9 shows those details for port F0/4 after the configuration in Example 8-8 was added.

Example 8-9 *Verifying the Data VLAN (Access VLAN) and Voice VLAN*

```
SW1# show interfaces FastEthernet 0/4 switchport
Name: Fa0/4
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 10 (VLAN0010)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: 11 (VLAN0011)
! The rest of the output is omitted for brevity
```

Working through the first three highlighted lines in the output, all those details should look familiar for any access port. The **switchport mode access** configuration command statically configures the administrative mode to be an access port, so the port of course operates as an access port. Also, as shown in the third highlighted line, the **switchport access vlan 10** configuration command defined the access mode VLAN as highlighted here.

The fourth highlighted line shows the one small new piece of information: the voice VLAN ID, as set with the **switchport voice vlan 11** command in this case. This small line of output is the only piece of information in the output that differs from the earlier access port examples in this chapter.

These ports act more like access ports than trunk ports. In fact, the **show interfaces** *type number* **switchport** command boldly proclaims, "Operational Mode: static access." However, one other **show** command reveals just a little more about the underlying operation with 802.1Q tagging for the voice frames.

As mentioned earlier, the **show interfaces trunk** command—that is, the command that does not include a specific interface in the middle of the command—lists the operational trunks on a switch. With IP telephony ports, the ports do not show up in the list of trunks either—providing evidence that these links are *not* treated as trunks. Example 8-10 shows just such an example.

However, the **show interfaces trunk** command with the interface listed in the middle of the command, as is also shown in Example 8-10, does list some additional information. Note that in this case, the **show interfaces F0/4 trunk** command lists the status as not-trunking, but with VLANs 10 and 11 allowed on the trunk. (Normally, on an access port, only the access VLAN is listed in the "VLANs allowed on the trunk" list in the output of this command.)

Example 8-10 *Allowed VLAN List and the List of Active VLANs*

```
SW1# show interfaces trunk
SW1# show interfaces F0/4 trunk

Port        Mode             Encapsulation  Status        Native vlan
Fa0/4       off              802.1q         not-trunking  1

Port        Vlans allowed on trunk
Fa0/4       10-11

Port        Vlans allowed and active in management domain
Fa0/4       10-11

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/4       10-11
```

Summary: IP Telephony Ports on Switches

It might seem as though this short topic about IP telephony and switch configuration includes a lot of small twists and turns and trivia, and it does. The most important items to remember are as follows:

Key Topic

- Configure these ports like a normal access port to begin: Configure it as a static access port and assign it an access VLAN.
- Add one more command to define the voice VLAN (**switchport voice vlan** *vlan-id*).
- Look for the mention of the voice VLAN ID, but no other new facts, in the output of the **show interfaces** *type number* **switchport** command.
- Look for both the voice and data (access) VLAN IDs in the output of the **show interfaces** *type number* **trunk** command.
- Do not expect to see the port listed in the list of operational trunks as listed by the **show interfaces trunk** command.

Troubleshooting VLANs and VLAN Trunks

A switch's data plane forwarding processes depend in part on VLANs and VLAN trunking. This final section of the chapter focuses on issues related to VLANs and VLAN trunks that could prevent LAN switching from working properly, focusing on a few items not yet discussed in the chapter. In particular, this section examines these steps an engineer can take to avoid issues:

Step 1. Confirm that the correct access VLANs have been assigned.

Step 2. Confirm that all VLANs are both defined and active.

Step 3. Check the allowed VLAN lists on both ends of each trunk to ensure that all VLANs intended to be used are included.

Step 4. Check for incorrect trunk configuration settings that result in one switch operating as a trunk, with the neighboring switch not operating as a trunk.

Step 5. Check the native VLAN settings on both ends of the trunk to ensure the settings match.

Confirm the Correct Access VLAN Is Assigned

To ensure that each access interface has been assigned to the correct VLAN, engineers need to confirm an interface operates as an access interface (as opposed to a **trunk interface**), determine the access VLAN assigned to the access interface, and compare the information to the documentation. The **show** commands listed in Table 8-4 can be particularly helpful in this process.

Key Topic

Table 8-4 Commands That Can Find Access Ports and VLANs

EXEC Command	Description
show vlan brief **show vlan**	Lists each VLAN and all interfaces assigned to that VLAN (but does not include operational trunks)
show vlan id *num*	Lists both access and trunk ports in the VLAN
show interfaces status	On ports operating as access ports, it lists the access VLAN, and on ports operating as trunk ports, it lists the word *trunk*.
show interfaces *type number* **switchport**	Identifies the interface's access VLAN and voice VLAN, plus the configured and operational mode (access or trunk)
show mac address-table	Lists MAC table entries, including the associated VLAN

If possible, start this step with the **show vlan** and **show vlan brief** commands. Both commands list all known VLANs and all access interfaces assigned to each VLAN, whether the interface is in a working or nonworking state. Be aware, however, that these two commands do not list operational trunks, so if you want to also see the trunk ports that support a VLAN, use the **show vlan id** *number* command.

After you determine the access interfaces and associated VLANs, if the interface is assigned to the wrong VLAN, use the **switchport access vlan** *vlan-id* interface subcommand to assign the correct VLAN ID.

Access VLANs Undefined or Disabled

Switches do not forward frames for VLANs that are (a) not known because the VLAN is not configured or has not been learned with VTP or (b) the VLAN is known but is disabled (shut down). This next topic summarizes the best ways to confirm that a switch knows that a particular VLAN exists, and if it exists, determines the shutdown state of the VLAN.

First, on the issue of whether a VLAN exists on a switch, a VLAN can be defined to a switch in two ways: using the **vlan** *number* global configuration command, or it can be learned from another switch using VTP. As mentioned earlier in this chapter, the examples in this book assume that you are not using VTP. If you discover that a VLAN does not exist on a switch, simply configure the VLAN as discussed earlier in the section, "Creating VLANs and Assigning Access VLANs to an Interface."

In addition to checking the configuration, you can check for the status of the VLAN (as well as whether it is known to the switch) using the **show vlan** command. No matter the VTP mode, this command will list all VLANs known to the switch, plus one of two VLAN state values, depending on the current state: either *active* or *act/lshut*. The second of these states means that the VLAN is shut down. Shutting down a VLAN disables the VLAN on that switch only, so *the switch will not forward frames in that VLAN*.

Switch IOS gives you two similar configuration methods with which to disable (**shutdown**) and enable (**no shutdown**) a VLAN. Example 8-11 shows how, first by using the global command [**no**] **shutdown vlan** *number* and then using the VLAN mode subcommand [**no**] **shutdown.** The example shows the global commands enabling and disabling VLANs 10 and 20, respectively, and using VLAN subcommands to enable and disable VLANs 30 and 40, respectively. The **show vlan brief** command at the end of the example confirms the shut-down (act/lshut) state of VLANs 10 and 30.

Example 8-11 *Enabling and Disabling VLANs on a Switch*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# no shutdown vlan 10
SW1(config)# shutdown vlan 20
SW1(config)# vlan 30
SW1(config-vlan)# no shutdown
SW1(config-vlan)# vlan 40
SW1(config-vlan)# shutdown
SW2(config-vlan)# end

SW1# show vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Gi1/0/1, Gi1/0/2, Gi1/0/3
                                                Gi1/0/4, Gi1/0/5, Gi1/0/6
                                                Gi1/0/7, Gi1/0/8, Gi1/0/9
                                                Gi1/0/10, Gi1/0/11, Gi1/0/12
                                                Gi1/0/13, Gi1/0/14, Gi1/0/15
                                                Gi1/0/16, Gi1/0/17, Gi1/0/18
                                                Gi1/0/19, Gi1/0/20, Gi1/0/21
                                                Gi1/0/22, Gi1/0/23, Gi1/0/24
                                                Te1/1/1, Te1/1/2, Te1/1/3
                                                Te1/1/4
10   VLAN0010                         act/lshut
20   VLAN0020                         active
30   VLAN0030                         act/lshut
40   VLAN0040                         active
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
```

NOTE The output of the **show vlan brief** command also lists a state of "act/unsup" for the reserved VLAN IDs 1002–1005, with "unsup" meaning "unsupported."

Example 8-12 shows another way to find the access VLANs on different ports by using the **show interfaces status** command. The switch has been configured before gathering the output in Example 8-12. First, the switch configures the first three ports as access ports in VLAN 10 and the next three as access ports in VLAN 20. The switch also configures two TenGigabitEthernet interfaces as trunks; note the word *trunk* under the VLAN heading for those interfaces.

Example 8-12 *Displaying Access Port VLANs with* **show interfaces status**

```
SW1# show interfaces status

Port      Name               Status       Vlan       Duplex  Speed Type
Gi1/0/1                      connected    10         a-full a-1000 10/100/1000BaseTX
Gi1/0/2                      connected    10         a-full a-1000 10/100/1000BaseTX
Gi1/0/3                      connected    10         a-full a-1000 10/100/1000BaseTX
Gi1/0/4                      connected    20         a-full a-1000 10/100/1000BaseTX
Gi1/0/5                      connected    20         a-full a-1000 10/100/1000BaseTX
Gi1/0/6                      connected    20         a-full a-1000 10/100/1000BaseTX
! Lines For G1/0/7 - G1/0/24 omitted for brevity
Te1/1/1                      connected    trunk        full    10G SFP-10Gbase-SR
Te1/1/2                      connected    trunk        full    10G SFP-10Gbase-SR
Te1/1/3                      notconnect   1            auto   auto unknown
Te1/1/4                      notconnect   1            auto   auto unknown
```

Mismatched Trunking Operational States

Trunking can be configured correctly so that both switches use trunking. However, trunks can also be misconfigured, with a couple of different results: either both switches do not trunk, or one switch trunks and the other does not. Both results cause problems.

The most common incorrect configuration—which results in both switches not trunking—is a configuration that uses the **switchport mode dynamic auto** command on both switches on the link. The word *auto* just makes us all want to think that the link would trunk automatically, but this command is both automatic and passive. As a result, both switches passively wait on the other device on the link to begin negotiations. Example 8-13 highlights those parts of the output from the **show interfaces switchport** command that confirm both the configured and operational states. Note that the output lists the operational mode as "static access" rather than "trunking."

Example 8-13 *Operational Trunking State*

```
SW1# show interfaces TenGigabitEthernet1/1/1 switchport
Name: Te1/1/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: static access
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: native
! lines omitted for brevity
```

A different incorrect trunking configuration has an even worse result: one switch trunks, sending tagged frames, but the neighboring switch does not trunk, so the neighboring switch discards any frames it receives that have a VLAN tag in the header. When this combination of events happens, the interface works in that the status on each end will be up/up or connected. Traffic in the native VLAN will actually cross the link successfully because those frames have no VLAN tags (headers). However, traffic in all the rest of the VLANs will not cross the link.

Figure 8-14 shows the incorrect configuration along with which side trunks and which does not. The side that trunks (SW1 in this case) enables trunking using the command **switchport mode trunk** but also disables Dynamic Trunking Protocol (DTP) negotiations using the **switchport nonegotiate** command. SW2's configuration also helps create the problem, by using one of the two trunking options that rely on DTP. Because SW1 has disabled DTP, SW2's DTP negotiations fail, and SW2 chooses to not trunk.

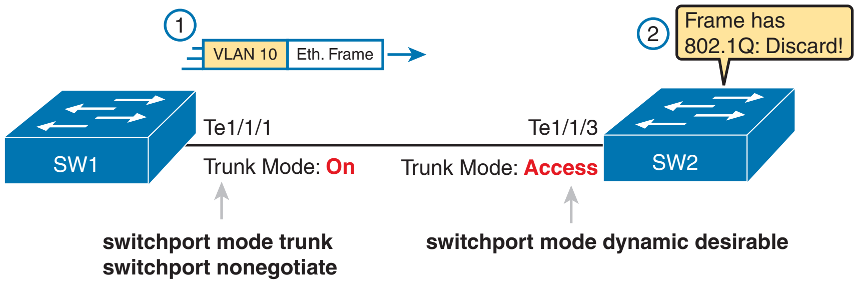


Figure 8-14 *Mismatched Trunking Operational States*

The figure shows what happens when using this incorrect configuration. At Step 1, SW1 could (for example) forward a frame in VLAN 10. However, SW2 would view any frame that arrives with an 802.1Q header as illegal because the frame has an 802.1Q header, and SW2 treats its G0/2 port as an access port. So, SW2 discards any 802.1Q frames received on that port.

The trunking issues shown here can be easily avoided by checking the configuration and by checking the trunk's operational state (mode) on both sides of the trunk. The best commands to check trunking-related facts are **show interfaces trunk** and **show interfaces switchport.** Just be aware that the switches do not prevent you from making these configuration mistakes.

The Supported VLAN List on Trunks

A Cisco switch can forward traffic for all defined and active VLANs. However, a particular VLAN trunk may not forward traffic for a defined and active VLAN for a variety of other reasons. You should learn how to identify which VLANs a particular trunk port currently supports and the reasons why the switch might not be forwarding frames for a VLAN on that trunk port.

The first category in this step can be easily done using the **show interfaces** *interface-id* **trunk** command, which lists information only about currently operational trunks. The best place to begin with this command is the last section of output, which lists the VLANs whose traffic will be forwarded over the trunk. Any VLANs that make it to this final list of VLANs in the command output meet the following criteria:

- The VLAN has not been removed from the *allowed VLAN list* on the trunk (as configured with the **switchport trunk allowed vlan** interface subcommand).
- The VLAN exists and is active on the local switch (as seen in the **show vlan** command).
- The VLAN has not been VTP-pruned from the trunk because the switch has disabled VTP. The trunk is in an STP forwarding state in that VLAN (as also seen in the **show spanning-tree vlan** *vlan-id* command).

The **switchport trunk allowed vlan** interface subcommand gives the network engineer a method to administratively limit the VLANs whose traffic uses a trunk. If the engineer wants all defined VLANs to be supported on a trunk, the engineer simply does not configure this command. If the engineer would like to limit the trunk to support a subset of the VLANs known to the switch, however, the engineer can add one or more **switchport trunk allowed vlan** interface subcommands.

For instance, in a switch that has configured VLANs 1 through 100, but no others, by default the switch would allow traffic in all 100 VLANs. However, the trunk interface command **switchport trunk allowed vlan 1-60** would limit the trunk to forward traffic for VLANs 1 through 60, but not the rest of the VLANs. Example 8-14 shows a sample of the command output from the **show interfaces trunk** command, which confirms the first list of VLAN IDs now lists VLANs 1–60. Without the **switchport trunk allowed vlan** command, the first list would have included VLANs 1–4094.

8

Example 8-14 *Allowed VLAN List and List of Active VLANs*

```
SW1# show interfaces trunk

Port        Mode         Encapsulation  Status       Native vlan
Te1/1/1     desirable    802.1q         trunking     1

Port        Vlans allowed on trunk
Te1/1/1     1-60

Port        Vlans allowed and active in management domain
Te1/1/1     1-59

Port        Vlans in spanning tree forwarding state and not pruned
Te1/1/1     1-58
```

The output of the **show interfaces trunk** command creates three separate lists of VLANs, each under a separate heading. These three lists show a progression of reasons why a VLAN is not forwarded over a trunk. Table 8-5 summarizes the headings that precede each list and the reasons why a switch chooses to include or not include a VLAN in each list. For instance, in Example 8-14, VLAN 60 has been shut down, and VLAN 59 happens to be in an STP blocking state. (Chapter 9, "Spanning Tree Protocol Concepts," has more information about STP.)

Key Topic

Table 8-5 VLAN Lists in the **show interfaces trunk** Command

List Position	Heading	Reasons
First	VLANs allowed	VLANs 1–4094, minus those removed by the **switchport trunk allowed** command
Second	VLANs allowed and active…	The first list, minus VLANs not defined to the local switch (that is, there is not a **vlan** global configuration command or the switch has not learned of the VLAN with VTP), and also minus those VLANs in shutdown mode
Third	VLANs in spanning tree…	The second list, minus VLANs in an STP blocking state for that interface, and minus VLANs VTP pruned from that trunk

Mismatched Native VLAN on a Trunk

Unfortunately, it *is* possible to set the native VLAN ID to different VLANs on either end of the trunk, using the **switchport trunk native vlan** *vlan-id* command. If the native VLANs differ according to the two neighboring switches, the switches will cause frames sent in the native VLAN to jump from one VLAN to the other.

For example, if switch SW1 sends a frame using native VLAN 1 on an 802.1Q trunk, SW1 does not add a VLAN header, as is normal for the native VLAN. When switch SW2 receives the frame, noticing that no 802.1Q header exists, SW2 assumes that the frame is part of SW2's configured native VLAN. If SW2 has been configured to think VLAN 2 is the native VLAN on that trunk, SW2 will try to forward the received frame into VLAN 2. (This effect of a frame being sent in one VLAN but then being believed to be in a different VLAN is called *VLAN hopping*.)

Chapter Review

Review this chapter's material using either the tools in the book or the interactive tools for the same material found on the book's companion website. Table 8-6 outlines the key review elements and where you can find them. To better track your study progress, record when you completed these activities in the second column.

Table 8-6 Chapter Review Tracking

Review Element	Review Date(s)	Resource Used
Review key topics		Book, website
Review key terms		Book, website
Answer DIKTA questions		Book, PTP

Review Element	Review Date(s)	Resource Used
Review config checklists		Book, website
Review command tables		Book
Review memory tables		Website
Do labs		Sim Lite, blog
Watch Video		Website

Review All the Key Topics

Key Topic

Table 8-7 Key Topics for Chapter 8

Key Topic Element	Description	Page Number
Figure 8-2	Basic VLAN concept	191
List	Reasons for using VLANs	192
Figure 8-5	Diagram of VLAN trunking	194
Figure 8-6	802.1Q header	195
Table 8-2	Options of the **switchport mode** command	203
Table 8-3	Expected trunking results based on the configuration of the **switchport mode** command	207
List	Definitions of data VLAN and voice VLAN	209
List	Summary of data and voice VLAN concepts, configuration, and verification	212
Table 8-4	Commands to find access ports and assigned VLANs	213
Table 8-5	Analysis of the three VLAN lists in the output from the **show interfaces** *interface-id* **trunk** command	218

Key Terms You Should Know

802.1Q, access interface, data VLAN, default VLAN, Layer 3 switch, native VLAN, static access interface, trunk, trunk interface, trunking administrative mode, trunking operational mode, VLAN, voice VLAN, VTP, VTP transparent mode

Do Labs

The Sim Lite software is a version of Pearson's full simulator learning product with a subset of the labs, included free with this book. The Sim Lite with this book includes a couple of labs about VLANs. Also, check the author's blog site pages for configuration exercises (Config Labs) at https://www.certskills.com.

Command References

Tables 8-8 and 8-9 list configuration and verification commands used in this chapter, respectively. As an easy review exercise, cover the left column in a table, read the right column, and try to recall the command without looking. Then repeat the exercise, covering the right column, and try to recall what the command does.

Table 8-8 Chapter 8 Configuration Command Reference

Command	Description
vlan *vlan-id*	Global config command that both creates the VLAN and puts the CLI into VLAN configuration mode.
name *vlan-name*	VLAN subcommand that names the VLAN.
[**no**] **shutdown**	VLAN mode subcommand that enables (**no shutdown**) or disables (**shutdown**) the VLAN.
[**no**] **shutdown vlan** *vlan-id*	Global config command that has the same effect as the [**no**] **shutdown** VLAN mode subcommands.
vtp mode {**server** | **client** | **transparent** | **off**}	Global config command that defines the VTP mode.
switchport mode {**access** | **dynamic** {**auto** | **desirable**} | **trunk**}	Interface subcommand that configures the trunking administrative mode on the interface.
switchport access vlan *vlan-id*	Interface subcommand that statically configures the interface into that one VLAN.
switchport trunk encapsulation {**dot1q** | **isl** | **negotiate**}	Interface subcommand that defines which type of trunking to use, assuming that trunking is configured or negotiated.
switchport trunk native vlan *vlan-id*	Interface subcommand that defines the native VLAN for a trunk port.
switchport nonegotiate	Interface subcommand that disables the negotiation of VLAN trunking.
switchport voice vlan *vlan-id*	Interface subcommand that defines the voice VLAN on a port, meaning that the switch uses 802.1Q tagging for frames in this VLAN.
switchport trunk allowed vlan *vlan-list*	Interface subcommand that defines the list of allowed VLANs. Ignores the existing list of allowed VLANs.
switchport trunk allowed vlan {**add** | **remove**} *vlan-list*	Interface subcommand that adds to or removes from the current set of allowed VLANs on a trunk, adjusting from the existing list of allowed VLANs.
switchport trunk allowed vlan {**all** | **none** | **except** *vlan-list*}	Interface subcommand that defines the allowed VLAN list as either all VLANs, no VLANs, or all except those in the configured list. Ignores the existing list of allowed VLANs.

Table 8-9 Chapter 8 EXEC Command Reference

Command	Description
show interfaces status	On ports operating as access ports, it lists the access VLAN, and on ports operating as trunk ports, it lists the word *trunk*.
show interfaces *interface-id* **switchport**	Lists information about any interface regarding administrative settings and operational state.

Command	Description
show interfaces trunk	Lists information about all operational trunks (but no other interfaces), including the list of VLANs that can be forwarded over the trunk.
show interfaces *interface-id* **trunk**	Lists trunking status about the listed interface, regardless of whether the interface currently operates as a trunk.
show vlan [**brief**]	Lists each VLAN and all interfaces assigned to that VLAN (but does not include operational trunks).
show vlan {**id** *vlan-id* | **name** *vlan-name*}	Lists information about a specific VLAN by ID or name, and interfaces, including trunks.
show vtp status	Lists VTP configuration and status information.

CHAPTER 9

Spanning Tree Protocol Concepts

This chapter covers the following exam topics:

2.0 Network Access

2.4 Configure and verify (Layer 2/Layer 3) EtherChannel (LACP)

2.5 Interpret basic operations of Rapid PVST+ Spanning Tree Protocol

2.5.a Root port, root bridge (primary/secondary), and other port names

2.5.b Port states and roles

2.5.c PortFast

2.5.d Root Guard, loop guard, BPDU filter, and BPDU guard

Spanning Tree Protocol (STP) allows Ethernet LANs to have redundant links in a LAN while overcoming the known problems that occur when adding those extra links. Using redundant links in a LAN design allows the LAN to keep working even when some links fail or even when some entire switches fail. Proper LAN design should add enough redundancy so that no single point of failure crashes the LAN; STP allows the design to use redundancy without causing other problems.

Historically, the IEEE first standardized STP as part of the IEEE 802.1D standard back in 1990, with pre-standard versions working even before that time. Over time, the industry and IEEE improved STP, with the eventual replacement of STP with an improved protocol: Rapid Spanning Tree Protocol (RSTP). The IEEE first released RSTP as amendment 802.1w and, in 2004, integrated RSTP into the 802.1D standard.

Today, most networks use RSTP rather than STP; however, STP and RSTP share many of the same mechanisms, and RSTP's improvements can be best understood in comparison to STP. For that reason, this chapter presents some details that apply only to STP, as a learning tool to help you understand RSTP.

This chapter organizes the material into four sections. The first section presents some core concepts about how both STP and RSTP discover a tree made of nodes (switches) and links so that no loops exist in a network. The second section then takes a brief look at the area in which STP differs the most from RSTP: in how STP reacts to changes in the network. The third section then shows how RSTP works much better than STP when reacting to changes. The final section touches on a variety of small optional STP and RSTP features.

Finally, be warned that this chapter, as well as Chapter 10, are a little longer than expected. If you like to think of each chapter as one study session, you might need to think about splitting this chapter into two study sessions. For the first study session, you should stop at the section, "Rapid STP Concepts." The second study session would consist of the "Rapid STP Concepts" and "Optional STP Features" sections.

“Do I Know This Already?” Quiz

Take the quiz (either here or use the PTP software) if you want to use the score to help you decide how much time to spend on this chapter. The letter answers are listed at the bottom of the page following the quiz. Appendix C, found both at the end of the book as well as on the companion website, includes both the answers and explanations. You can also find both answers and explanations in the PTP testing software.

Table 9-1 “Do I Know This Already?” Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
STP and RSTP Basics	1–2
Details Specific to STP (and Not RSTP)	3–4
Rapid STP Concepts	5–6
Optional STP Features	7–8

1. Which of the following port states are stable states used when STP has completed convergence? (Choose two answers.)
   - **a.** Blocking
   - **b.** Forwarding
   - **c.** Listening
   - **d.** Learning
   - **e.** Discarding
2. Which of the following bridge IDs wins election as root, assuming that the switches with these bridge IDs are in the same network?
   - **a.** 32769:0200.1111.1111
   - **b.** 32769:0200.2222.2222
   - **c.** 4097:0200.1111.1111
   - **d.** 4097:0200.2222.2222
   - **e.** 40961:0200.1111.1111
3. Which of the following are transitory port states used only during the process of STP convergence? (Choose two answers.)
   - **a.** Blocking
   - **b.** Forwarding
   - **c.** Listening
   - **d.** Learning
   - **e.** Discarding
4. Which of the following facts determines how often a nonroot bridge or switch sends an STP Hello BPDU message?
   - **a.** The Hello timer as configured on that switch.
   - **b.** The Hello timer as configured on the root switch.

- **c.** It is always every 2 seconds.
- **d.** The switch reacts to BPDUs received from the root switch by sending another BPDU 2 seconds after receiving the root BPDU.

5. Which of the following RSTP port states have the same name and purpose as a port state in traditional STP? (Choose two answers.)

- **a.** Blocking
- **b.** Forwarding
- **c.** Listening
- **d.** Learning
- **e.** Discarding

6. RSTP adds features beyond STP that enable ports to be used for a role if another port on the same switch fails. Which of the following statements correctly describe a port role that is waiting to take over for another port role? (Choose two answers.)

- **a.** An alternate port waits to become a root port.
- **b.** A backup port waits to become a root port.
- **c.** An alternate port waits to become a designated port.
- **d.** A backup port waits to become a designated port.

7. What STP/RSTP feature causes an interface to be placed in the forwarding state as soon as the interface is physically active?

- **a.** STP
- **b.** EtherChannel
- **c.** Root Guard
- **d.** PortFast

8. Which optional STP feature reacts to a subset of incoming STP BPDUs to disable the port, but allows and processes some other STP BPDUs?

- **a.** Loop Guard
- **b.** BPDU Guard
- **c.** Root Guard
- **d.** PortFast

Foundation Topics

STP and RSTP Basics

Without some mechanism like **Spanning Tree Protocol (STP)** or **Rapid STP (RSTP)**, a LAN with redundant links would cause Ethernet frames to loop for an indefinite period of time. With STP or RSTP enabled, some switches block ports so that these ports do not forward frames. STP and RSTP intelligently choose which ports block, with two goals in mind:

- All devices in a VLAN can send frames to all other devices. In other words, STP or RSTP does not block too many ports, cutting off some parts of the LAN from other parts.
- Frames have a short life and do not loop around the network indefinitely.

STP and RSTP strike a balance, allowing frames to be delivered to each device, without causing the problems that occur when frames loop through the network over and over again.

NOTE This first major section of the chapter explains details of both STP and RSTP, so this section uses the term *STP/RSTP* to refer to these protocols together. Note that this term is just a convenient shorthand. Later in the chapter, the text will point out differences between STP and RSTP and begin using the terms *STP* and *RSTP* separately, referring to only the specific protocol.

STP/RSTP prevents looping frames by adding an additional check on each interface before a switch uses it to send or receive user traffic. That check: If the port is in STP/RSTP **forwarding state** in that VLAN, use it as normal; if it is in STP/RSTP **blocking state**, however, block all user traffic and do not send or receive user traffic on that interface in that VLAN.

Note that these STP/RSTP states do not change the other information you already know about switch interfaces. The interface's state of connected/notconnect does not change. The interface's operational state as either an access or trunk port does not change. STP/RSTP adds this additional state, with the blocking state basically disabling the interface.

In many ways, the preceding two paragraphs sum up what STP/RSTP does. However, the details of how STP/RSTP does its work can take a fair amount of study and practice. This first major section of the chapter begins by explaining the need for STP/RSTP and the basic ideas of what STP/RSTP does to solve the problem of looping frames. The majority of this section then looks at how STP/RSTP goes about choosing which switch ports to block to accomplish its goals.

The Need for Spanning Tree

STP/RSTP prevents three common problems in Ethernet LANs. All three problems occur as a side effect of one fact: without STP/RSTP, some Ethernet frames would loop around the network for a long time (hours, days, literally forever if the LAN devices and links never failed).

Just one looping frame causes what is called a *broadcast storm*. Broadcast storms happen when any kind of Ethernet frames—broadcast frames, multicast frames, or unknown-destination unicast frames—loop around a LAN indefinitely. Broadcast storms can saturate all the links with copies of that one single frame, crowding out good frames, as well as significantly impacting end-user device performance by making the PCs process too many broadcast frames.

To help you understand how this occurs, Figure 9-1 shows a sample network in which Bob sends a broadcast frame. The dashed lines show how the switches forward the frame when STP/RSTP does not exist.

Remember that LAN switch? That logic tells switches to flood broadcasts out all interfaces in the same VLAN except the interface in which the frame arrived. In Figure 9-1, that means SW3 forwards Bob's frame to SW2, SW2 forwards the frame to SW1, SW1 forwards the frame back to SW3, and SW3 forwards it back to SW2 again.

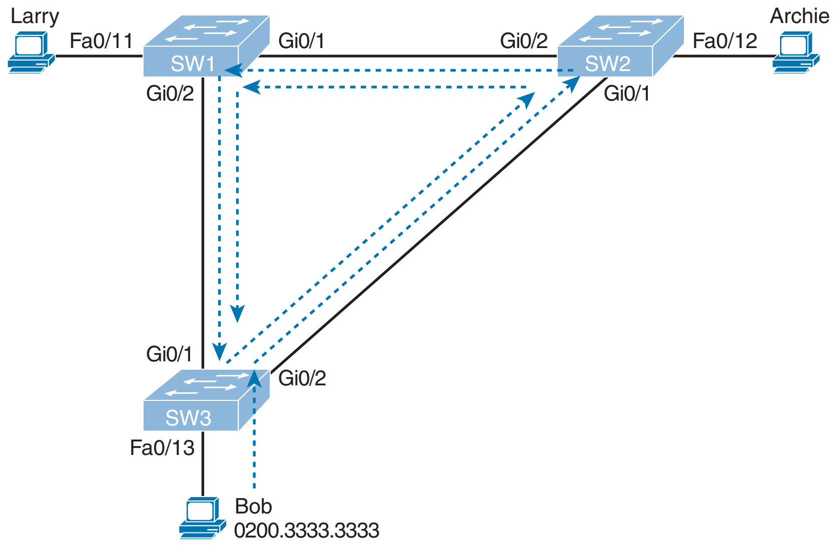


Figure 9-1 *Broadcast Storm*

When broadcast storms happen, frames like the one in Figure 9-1 keep looping until something changes—someone shuts down an interface, reloads a switch, or does something else to break the loop. Also note that the same event happens in the opposite direction. When Bob sends the original frame, SW3 also forwards a copy to SW1, SW1 forwards it to SW2, and so on.

The storm also causes a much more subtle problem called *MAC table instability*. MAC table instability means that the switches' MAC address tables keep changing because frames with the same source MAC arrive on different ports. To see why, follow this example, in which SW3 begins Figure 9-1 with a MAC table entry for Bob, at the bottom of the figure, associated with port Fa0/13:

```
0200.3333.3333   Fa0/13   VLAN 1
```

However, now think about the switch-learning process that occurs when the looping frame goes to SW2, then SW1, and then back into SW3's Gi0/1 interface. SW3 thinks, "Hmm…the source MAC address is 0200.3333.3333, and it came in my Gi0/1 interface. Update my MAC table!" This results in the following entry on SW3, with interface Gi0/1 instead of Fa0/13:

```
0200.3333.3333   Gi0/1   VLAN 1
```

At this point, SW3 itself cannot correctly deliver frames to Bob's MAC address. At that instant, if a frame arrives at SW3 destined for Bob—a different frame than the looping frame that causes the problems—SW3 incorrectly forwards the frame out Gi0/1 to SW1, creating even more congestion.

The looping frames in a broadcast storm also cause a third problem: multiple copies of the frame arrive at the destination. Consider a case in which Bob sends a frame to Larry but

Answers to the "Do I Know This Already?" quiz:

1 A, B **2** C **3** C, D **4** B **5** B, D **6** A, D **7** D, **8** C none of the switches know Larry's MAC address. Switches flood frames sent to unknown destination unicast MAC addresses. When Bob sends the frame destined for Larry's MAC address, SW3 sends a copy to both SW1 and SW2. SW1 and SW2 also flood the frame, causing copies of the frame to loop. SW1 also sends a copy of each frame out Fa0/11 to Larry. As a result, Larry gets multiple copies of the frame, which may result in an application failure, if not more pervasive networking problems.

Table 9-2 summarizes the main three classes of problems that occur when STP/RSTP is not used in a LAN that has redundancy.

Key Topic

Table 9-2 Three Classes of Problems Caused by Not Using STP in Redundant LANs

Problem	Description
Broadcast storms	The forwarding of a frame repeatedly on the same links, consuming significant parts of the links' capacities
MAC table instability	The continual updating of a switch's MAC address table with incorrect entries, in reaction to looping frames, resulting in frames being sent to the wrong locations
Multiple frame transmission	A side effect of looping frames in which multiple copies of one frame are delivered to the intended host, confusing the host

What Spanning Tree Does

STP/RSTP prevents loops by placing each switch port in either a forwarding state or a blocking state. Interfaces in the forwarding state act as normal, forwarding and receiving frames. However, interfaces in a blocking state do not process any frames except STP/RSTP messages (and some other overhead messages). Interfaces that block do not forward user frames, do not learn MAC addresses of received frames, and do not process received user frames.

Figure 9-2 shows a simple STP/RSTP tree that solves the problem shown in Figure 9-1 by placing one port on SW3 in the blocking state.

Key Topic

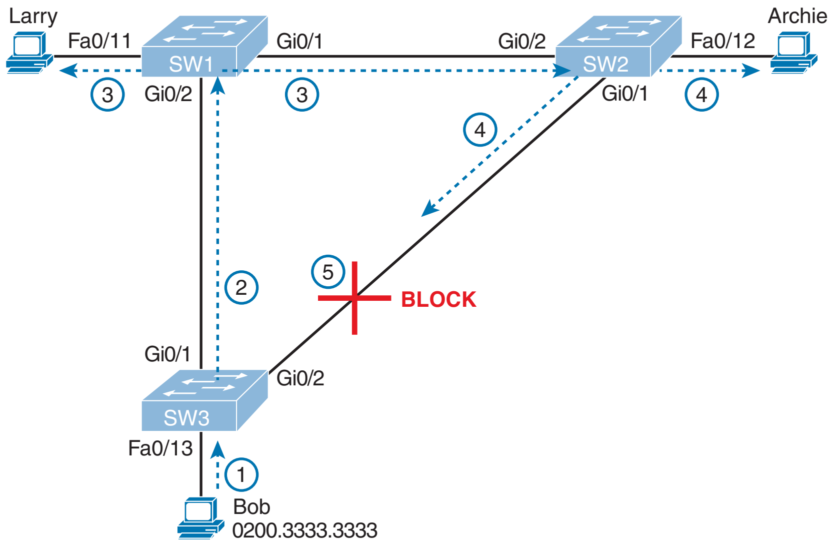


Figure 9-2 *What STP/RSTP Does: Blocks a Port to Break the Loop*

9

Now when Bob sends a broadcast frame, the frame does not loop. As shown in the steps in the figure:

Step 1. Bob sends the frame to SW3.

Step 2. SW3 forwards the frame only to SW1, but not out Gi0/2 to SW2, because SW3's Gi0/2 interface is in a blocking state.

Step 3. SW1 floods the frame out both Fa0/11 and Gi0/1.

Step 4. SW2 floods the frame out Fa0/12 and Gi0/1.

Step 5. SW3 physically receives the frame, but it ignores the frame received from SW2 because SW3's Gi0/2 interface is in a blocking state.

With the STP/RSTP topology in Figure 9-2, the switches simply do not use the link between SW2 and SW3 for traffic in this VLAN, which is the minor negative side effect of STP. However, if either of the other two links fails, STP/RSTP converges so that SW3 forwards instead of blocks on its Gi0/2 interface.

NOTE The term *STP convergence* refers to the process by which the switches collectively realize that something has changed in the LAN topology and determine whether they need to change which ports block and which ports forward.

That completes the description of what STP/RSTP does, placing each port into either a forwarding or blocking state. The more interesting question, and the one that takes a lot more work to understand, is how and why STP/RSTP makes its choices. How does STP/RSTP manage to make switches block or forward on each interface? And how does it converge to change state from blocking to forwarding to take advantage of redundant links in response to network outages? The following pages answer these questions.

How Spanning Tree Works

The STP/RSTP algorithm creates a spanning tree of interfaces that forward frames. The tree structure of forwarding interfaces creates a single path to and from each Ethernet link, just like you can trace a single path in a living, growing tree from the base of the tree to each leaf.

NOTE STP existed before LAN switches, with the STP processes performed by devices called bridges. Some terms and command output still use the older term *bridge*, so when working with STP/RSTP, consider the terms *bridge* and *switch* synonymous.

The process used by STP, sometimes called the *spanning-tree algorithm* (STA), chooses the interfaces that should be placed into a forwarding state. For any interfaces not chosen to be in a forwarding state, STP/RSTP places the interfaces in a blocking state. In other words, STP/RSTP simply picks which interfaces should forward, and any interfaces left over go to a blocking state.

STP/RSTP uses three criteria to choose whether to put an interface in a forwarding state:

- STP/RSTP elects a **root switch**. STP puts all working interfaces on the root switch in a forwarding state.
- Each nonroot switch calculates the least-cost path between itself and the root switch based on STP/RSTP interface costs. The switch port that begins that least-cost path is its *root port* (RP), and the cost is the switch's **root cost**.
- In a modern Ethernet that uses switches, each physical link connects two devices only. With two switches on a link, the switch with the lowest root cost becomes the *designated switch*, and its port connected to that link is the **designated port** (DP) on the link.

NOTE A root switch places all interfaces into a forwarding state because each port on the root switch always wins its designated port (DP) election. However, it is easier to just remember that all the root switches' working interfaces will forward frames.

All other interfaces are placed in a blocking state. Table 9-3 summarizes the reasons STP/RSTP places a port in a forwarding or blocking state.

Key Topic

Table 9-3 STP/RSTP: Reasons for Forwarding or Blocking

Characterization of Port	STP State	Description
All the root switch's ports	Forwarding	The root switch is always the designated switch on all connected segments.
Each nonroot switch's root port	Forwarding	The port through which the switch has the least cost to reach the root switch (lowest root cost).
Each LAN's designated port	Forwarding	The switch forwarding the Hello on to the segment, with the lowest root cost, is the designated switch for that segment.
All other working ports	Blocking	The port is not used for forwarding user frames, nor are any frames received on these interfaces considered for forwarding.

NOTE STP/RSTP should not use nonworking interfaces in the working STP/RSTP topology. To remove all nonworking interfaces from consideration—that is, all that are not in a connected (up/up) interface state—STP/RSTP assigns nonworking ports with the *disabled port role*. STP/RSTP does not consider these ports as potential RP or DP ports. STP also places these ports into a **disabled state** (much like blocking).

The STP Bridge ID and Hello BPDU

The STA begins with an election of one switch to be the root switch. To better understand this election process, you need to understand the STP/RSTP messages sent between switches as well as the concept and format of the identifier used to uniquely identify each switch.

The STP/RSTP **bridge ID** (BID) is an 8-byte value unique to each switch. The bridge ID consists of a 2-byte priority field and a 6-byte system ID, with the system ID being based on a universal (burned-in) MAC address in each switch. Using a burned-in MAC address ensures that each switch's bridge ID will be unique.

STP/RSTP defines messages called **bridge protocol data units (BPDU)**, also called configuration BPDUs, which switches use to exchange information with each other. The most common BPDU, called a **Hello BPDU**, lists many details, including the sending switch's BID. By listing its own unique BID, switches can tell which switch sent which Hello BPDU. Table 9-4 lists some of the key information in the Hello BPDU.

Key Topic

Table 9-4 Fields in the STP Hello BPDU

Field	Description
Root bridge ID	The bridge ID of the switch that the sender of this Hello currently believes to be the root switch
Sender's bridge ID	The bridge ID of the switch sending this Hello BPDU
Sender's root cost	The STP/RSTP cost between this switch and the current root
Timer values on the root switch	Includes the Hello timer, MaxAge timer, and **forward delay** timer

For the time being, just keep the first three items from Table 9-4 in mind as the following sections work through the three steps in how STP/RSTP chooses the interfaces to place into a forwarding state. Next, the text examines the three main steps in the STP/RSTP process.

Electing the Root Switch

Switches elect a root switch based on the BIDs in the BPDUs. The root switch is the switch with the lowest numeric value for the BID. Because the two-part BID starts with the priority value, essentially the switch with the lowest priority becomes the root. For example, if one switch has priority 4096, and another switch has priority 8192, the switch with priority 4096 wins, regardless of what MAC address was used to create the BID for each switch.

If a tie occurs based on the priority portion of the BID, the switch with the lowest MAC address portion of the BID is the root. No other tiebreaker should be needed because switches use one of their own universal (burned-in) MAC addresses as the second part of their BIDs. So, if the priorities tie, and one switch uses a MAC address of 0200.0000.0000 as part of the BID and the other uses 0811.1111.1111, the first switch (MAC 0200.0000.0000) becomes the root switch.

STP/RSTP elects a root switch in a manner not unlike a political election. The process begins with all switches claiming to be the root by sending Hello BPDUs listing their own BID as the root BID. If a switch hears a Hello that lists a better (lower) BID, that switch stops advertising itself as root and starts forwarding the superior Hello. The Hello sent by the better switch lists the better switch's BID as the root. It works like a political race in which a less-popular candidate gives up and leaves the race, throwing his support behind the more popular candidate. Eventually, everyone agrees which switch has the best (lowest) BID, and everyone supports the elected switch—which is where the political race analogy falls apart.

NOTE A better Hello, meaning that the listed root's BID is better (numerically lower), is called a *superior Hello*; a worse Hello, meaning that the listed root's BID is not as good (numerically higher), is called an *inferior Hello*.

Figure 9-3 shows the beginning of the root election process. In this case, SW1 has advertised itself as root, as have SW2 and SW3. However, SW2 now believes that SW1 is a better root, so SW2 is now forwarding the Hello originating at SW1. So, at this point, the figure shows SW1 is saying Hello, claiming to be root; SW2 agrees and is forwarding SW1's Hello that lists SW1 as root; but SW3 is still claiming to be best, sending its own Hello BPDUs, listing SW3's BID as the root.

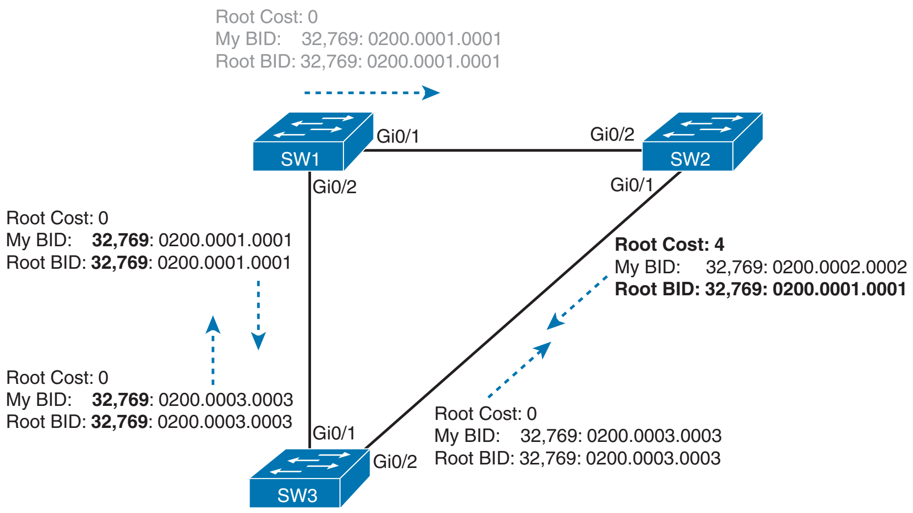

Root Cost: 0
My BID: 32,769: 0200.0001.0001
Root BID: 32,769: 0200.0001.0001
SW1
Gi0/1
Gi0/2
SW2
Gi0/2
Gi0/1
Root Cost: 0
My BID: **32,769**: 0200.0001.0001
Root BID: **32,769**: 0200.0001.0001
Root Cost: 4
My BID: 32,769: 0200.0002.0002
Root BID: 32,769: 0200.0001.0001
Root Cost: 0
My BID: **32,769**: 0200.0003.0003
Root BID: **32,769**: 0200.0003.0003
Root Cost: 0
My BID: 32,769: 0200.0003.0003
Root BID: 32,769: 0200.0003.0003
Gi0/1
Gi0/2
SW3

Figure 9-3 *Beginnings of the Root Election Process*

Two candidates still exist in Figure 9-3: SW1 and SW3. So, who wins? Well, from the BID, the lower-priority switch wins; if a tie occurs, the lower MAC address wins. As shown in the figure on the left, the switch priority values (32,769) tie, but SW1 has a lower BID (32,769:0200.0001.0001) than SW3 (32,769:0200.0003.0003) due to its lower MAC address. SW1 wins, and SW3 now also believes that SW1 is the better switch. Figure 9-4 shows the resulting Hello messages sent by the switches.

Summarizing, the root election happens through each switch claiming to be root, with the best switch being elected based on the numerically lowest BID. Breaking down the BID into its components, the comparisons can be made as:

Key Topic

- The lowest priority
- If that ties, the lowest switch MAC address

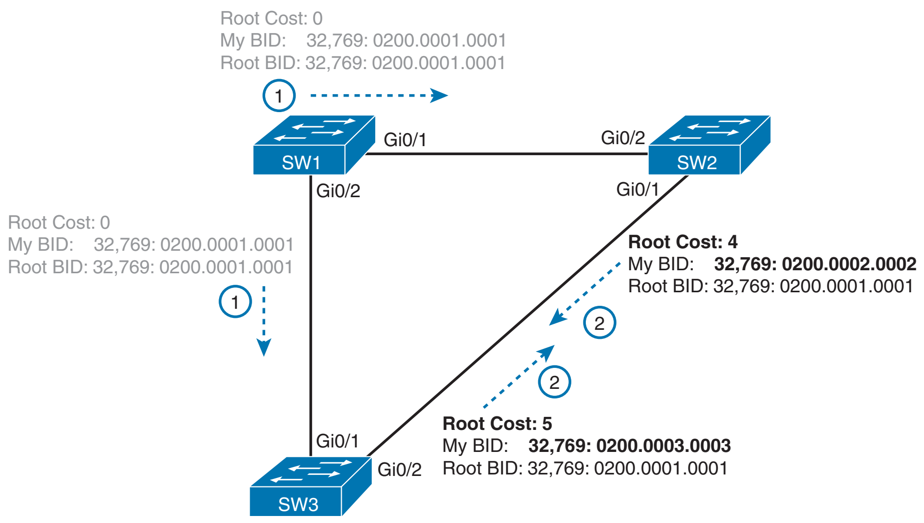

Root Cost: 0
My BID: 32,769: 0200.0001.0001
Root BID: 32,769: 0200.0001.0001
1
SW1
Gi0/1
Gi0/2
SW2
Gi0/2
Gi0/1
Root Cost: 0
My BID: 32,769: 0200.0001.0001
Root BID: 32,769: 0200.0001.0001
1
Root Cost: 4
My BID: 32,769: 0200.0002.0002
Root BID: 32,769: 0200.0001.0001
2
2
Root Cost: 5
My BID: 32,769: 0200.0003.0003
Root BID: 32,769: 0200.0001.0001
Gi0/1
Gi0/2
SW3

Figure 9-4 *SW1 Wins the Election*

Choosing Each Switch's Root Port

The second part of the STP/RSTP process occurs when each nonroot switch chooses its one and only *root port* (RP). A switch's RP is its interface through which it has the least STP/RSTP cost to reach the root switch (least root cost).

You can easily see the idea behind a switch's cost to reach the root. Just look at a network diagram that shows the root switch, lists the STP/RSTP cost associated with each switch port, and identifies the nonroot switch in question. Switches use a different process than you can see by looking at a network diagram, of course, but using a diagram can make it easier to learn the idea.

Figure 9-5 shows just such a figure, with the same three switches shown in the last several figures. SW1 has already won the election as root, and the figure considers the cost from SW3's perspective. (Note that the figure uses some nondefault cost settings.)

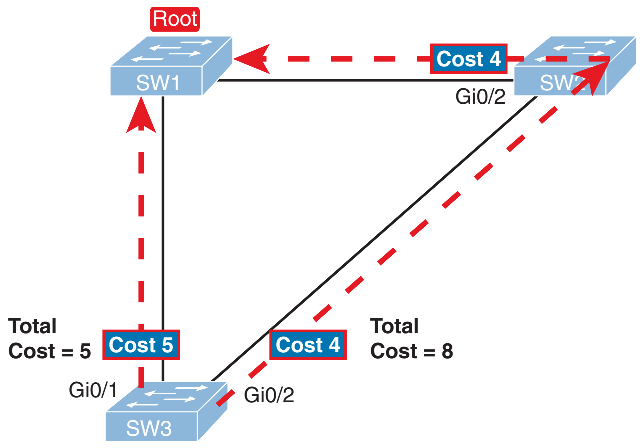


Figure 9-5 *How a Human Might Calculate STP/RSTP Cost from SW3 to the Root (SW1)*

SW3 has two possible physical paths to send frames to the root switch: the direct path to the left and the indirect path to the right through switch SW2. The cost is the sum of the costs of all the *switch ports the frame would exit* if it flowed over that path. (The calculation ignores the inbound ports.) As you can see, the cost over the direct path out SW3's G0/1 port has a total cost of 5, and the other path has a total cost of 8. SW3 picks its G0/1 port as root port because it is the port that is part of the least-cost path to send frames to the root switch.

Switches come to the same conclusion but using a different process. Instead, they add their local interface STP/RSTP cost to the root cost listed in each received Hello BPDU. The STP/RSTP port cost is simply an integer value assigned to each interface, per VLAN, for the purpose of providing an objective measurement that allows STP/RSTP to choose which interfaces to add to the STP/RSTP topology. The switches also look at their neighbor's root cost, as announced in Hello BPDUs received from each neighbor.

Figure 9-6 shows an example of how switches calculate their best root cost and then choose their root port, using the same topology and STP/RSTP costs as shown in Figure 9-5. STP/RSTP on SW3 calculates its cost to reach the root over the two possible paths by adding the advertised cost (in Hello messages) to the interface costs listed in the figure.

Key Topic

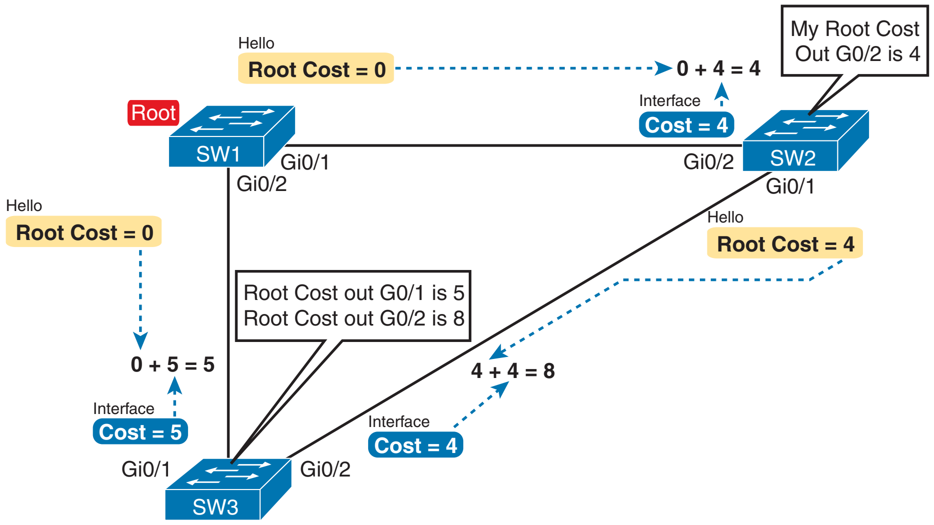


Figure 9-6 *How STP/RSTP Actually Calculates the Cost from SW3 to the Root*

Focus on the process for a moment. The root switch sends Hellos, with a listed root cost of 0. The idea is that the root's cost to reach itself is 0.

Next, look on the left of the figure. SW3 takes the received cost (0) from the Hello sent by SW1 and adds the interface cost (5) of the interface on which that Hello was received. SW3 calculates that the cost to reach the root switch, out that port (G0/1), is 5.

On the right side, SW2 has realized its best cost to reach the root is cost 4. So, when SW2 forwards the Hello toward SW3, SW2 lists a root cost 4. SW3's STP/RSTP port cost on port G0/2 is 4, so SW3 determines a total cost to reach root out its G0/2 port of 8.

As a result of the process depicted in Figure 9-6, SW3 chooses Gi0/1 as its RP because the cost to reach the root switch through that port (5) is lower than the other alternative (Gi0/2, cost 8). Similarly, SW2 chooses Gi0/2 as its RP, with a cost of 4 (SW1's advertised cost of 0 plus SW2's Gi0/2 interface cost of 4). Each switch places its root port into a forwarding state.

Switches need a tiebreaker to use in case the best root cost ties for two or more paths. If a tie occurs, the switch applies these three tiebreakers to the paths that tie, in order, as follows:

1. Choose based on the lowest neighbor bridge ID.
2. Choose based on the lowest neighbor port priority.
3. Choose based on the lowest neighbor internal port number.

Choosing the Designated Port on Each LAN Segment

The final STP/RSTP step determines the designated port on each LAN segment. The switch with the lowest root cost wins the competition. With two switches on the ends of a link, the switch with the lower cost to reach the root becomes the designated switch, with its port on that link becoming the DP on that segment. On a link with a switch and a second device that does not use STP/RSTP (for instance, a router or PC), the switch becomes the DP.

The process to choose the DP begins with the switch(es) forwarding Hellos onto the link. The Hello messages include a field to list the root cost of the switch that forwarded the Hello. Determining which switch has the lowest root cost requires a simple comparison. Once a switch port becomes the DP, it continues to forward Hellos onto the link.

For example, earlier Figure 9-4 shows in bold text the parts of the Hello messages from both SW2 and SW3 that determine the choice of DP on that segment. Note that both SW2 and SW3 list their respective root cost (cost 4 on SW2 and cost 5 on SW3). SW2 lists the lower cost, so SW2's Gi0/1 port is the designated port on that LAN segment.

All DPs are placed into a forwarding state; in this case, SW2's Gi0/1 interface will be in a forwarding state.

If the advertised costs tie, the switches break the tie by choosing the switch with the lower BID. In this case, SW2 would also have won, with a BID of 32769:0200.0002.0002 versus SW3's 32769:0200.0003.0003.

NOTE Two additional tiebreakers are needed in some cases, although these would be unlikely today. A single switch can connect two or more interfaces to the same collision domain by connecting to a hub. In that case, the one switch hears its own BPDUs. So, if a switch ties with itself, two additional tiebreakers are used: the lowest interface STP/RSTP priority and, if that ties, the lowest internal interface number.

The only interface that does not have a reason to be in a forwarding state on the three switches in the examples shown in Figures 9-3 through 9-6 is SW3's Gi0/2 port. So, the STP/RSTP process is now complete. Table 9-5 outlines the state of each port and shows why it is in that state.

Table 9-5 State of Each Interface

Switch Interface	State	Reason Why the Interface Is in Forwarding State
SW1, Gi0/1	Forwarding	The interface is on the root switch, so it becomes the DP on that link.
SW1, Gi0/2	Forwarding	The interface is on the root switch, so it becomes the DP on that link.
SW2, Gi0/2	Forwarding	The root port of SW2.
SW2, Gi0/1	Forwarding	The designated port on the LAN segment to SW3.
SW3, Gi0/1	Forwarding	The root port of SW3.
SW3, Gi0/2	Blocking	Not the root port and not the designated port.

Note that the examples in this section focus on the links between the switches, but switch ports connected to endpoint devices should become DPs and settle into a forwarding state. Working through the logic, each switch will forward BPDUs on each port as part of the process to determine the DP on that LAN. Endpoints should ignore those messages because they do not run STP/RSTP, so the switch will win and become the DP on every access port.

Configuring to Influence the STP Topology

STP/RSTP works by default on Cisco switches, so all the settings needed by a switch have a useful default. Switches have a default BID, based on a default priority value and adding a universal MAC address that comes with the switch hardware. Additionally, switch interfaces have default STP/RSTP costs based on the current operating speed of the switch interfaces.

Network engineers often want to change the STP/RSTP settings to then change the choices STP/RSTP makes in a given LAN. Two main tools available to the engineer are to configure the bridge ID and to change STP/RSTP port costs.

First, to change the BID, the engineer can set the priority used by the switch, while continuing to use the universal MAC address as the final 48 bits of the BID. For instance, giving a switch the lowest priority value among all switches will cause that switch to win the root election.

Port costs also have default values, per port, per VLAN. You can configure these port costs, which will in turn impact many switch's calculations of the root cost. For instance, to favor one link, give the ports on that link a lower cost, or to avoid a link, give the ports a higher cost.

Table 9-6 lists the default port costs suggested by IEEE. IOS on Cisco switches has long used the default settings as defined as far back as the 1998 version of the STP standard. The latest IEEE standards suggest values that are more useful when using links faster than 10 Gbps. You can configure a switch to use the old (short) or new (long) defaults using the global command (**spanning-tree pathcost method** {**short** | **long**})—but it makes sense to use the same setting on all switches in the same campus.

Key Topic

Table 9-6 Default Port Costs According to IEEE

Ethernet Speed	IEEE Cost: 1998 (and Before)	IEEE Cost: 2004 (and After)
10 Mbps	100	2,000,000
100 Mbps	19	200,000
1 Gbps	4	20,000
10 Gbps	2	2000
100 Gbps	N/A	200
1 Tbps	N/A	20

Also, be aware that Cisco switches choose the default cost based on the operating speed of the link, not the maximum speed. That is, if a 10/100/1000 port runs at 10 Mbps for some reason, its default STP cost on a Cisco switch is 100.

Details Specific to STP (and Not RSTP)

As promised in the introduction to this chapter, the first section showed features that apply to both STP and RSTP. This next heading acts as the turning point, with the next several pages being about STP only. The upcoming section titled "Rapid STP Concepts" then shows details specific to RSTP, in contrast to STP.

Once the engineer has finished all STP configuration, the STP topology should settle into a stable state and not change, at least until the network topology changes. This section examines the ongoing operation of STP while the network is stable, and then it covers how STP converges to a new topology when something changes.

Note that almost all the differences between STP and RSTP revolve around the activities of waiting for and reacting to changes in the topology. STP performed well for the era and circumstances in which it was created. The "rapid" in RSTP refers to the improvements to how fast RSTP could react when changes occur—so understanding how STP reacts will be useful to understand why RSTP reacts faster. These next few pages show the specifics of STP (and not RSTP) and how STP reacts to and manages convergence when changes happen in an Ethernet LAN.

STP Activity When the Network Remains Stable

When the network remains stable, the root switch generates a new Hello BPDU, with the nonroot switches forwarding copies of the Hello so that the Hellos traverse the spanning tree. The root generates a new Hello every Hello time (default 2 seconds), with each nonroot switch forwarding the Hello out any interfaces in a forwarding state after updating the Hello. The following steps summarize the steady-state operation when nothing is currently changing in the STP topology:

Key Topic

Step 1. The root creates and sends a Hello BPDU, with a root cost of 0, out all its working interfaces (those in a forwarding state).

Step 2. The nonroot switches receive the Hello on their root ports. After changing the Hello to list their own BID as the sender's BID and listing that switch's root cost, the switch forwards the Hello out all designated ports.

Step 3. Steps 1 and 2 repeat every Hello time until something changes.

When a switch fails to receive a Hello in its root port, it knows a problem might be occurring in the network. Each switch relies on these periodically received Hellos from the root as a way to know that its path to the root is still working. When a switch ceases to receive the Hellos, or receives a Hello that lists different details, something has failed, so the switch reacts and starts the process of changing the spanning-tree topology.

STP Timers That Manage STP Convergence

For various reasons, the STP convergence process requires the use of three timers, listed in Table 9-7. Note that all switches use the timers as dictated by the root switch, which the root lists in its periodic Hello BPDU messages.

Key Topic

Table 9-7 STP Timers

Timer	Default Value	Description
Hello	2 seconds	The time period between Hellos created by the root.
MaxAge	10 times Hello	How long any switch should wait, after ceasing to hear Hellos, before trying to change the STP topology.
Forward delay	15 seconds	Delay that affects the process that occurs when an interface changes from a blocking state to a forwarding state. A port stays in an interim **listening state**, and then an interim **learning state**, for the number of seconds defined by the forward delay timer.

If a switch does not get one expected Hello BPDU within the Hello time, the switch continues as normal. However, if the Hellos do not show up again within MaxAge time, the switch reacts by taking steps to change the STP topology. With default settings, MaxAge is 20 seconds (ten times the default Hello timer of 2 seconds). So, a switch would go 20 seconds without hearing a Hello before reacting.

After MaxAge expires, the switch essentially makes all its STP choices again, based on any Hellos it receives from other switches. It reevaluates which switch should be the root switch. If the local switch is not the root, it chooses its RP. And it determines whether it is the DP on each of its other links.

The best way to describe STP convergence is to show an example using the same familiar topology. Figure 9-7 shows the same familiar figure, with SW3's Gi0/2 in a blocking state, but SW1's Gi0/2 interface has just failed.

In the scenario shown in the figure, SW3 reacts to the change because SW3 fails to receive its expected Hellos on its Gi0/1 interface. However, SW2 does not need to react because SW2 continues to receive its periodic Hellos in its Gi0/2 interface. In this case, SW3 reacts either when **MaxAge** time passes without hearing the Hellos, or as soon as SW3 notices that interface Gi0/1 has failed. (If the interface fails, the switch can assume that the Hellos will not be arriving in that interface anymore.)

Now that SW3 can act, it begins by reevaluating the choice of root switch. SW3 still receives the Hellos from SW2, as forwarded from the root (SW1). SW1 still has a lower BID than SW3; otherwise, SW1 would not have already been the root. So, SW3 decides that SW1 wins the root election and that SW3 is not the root.

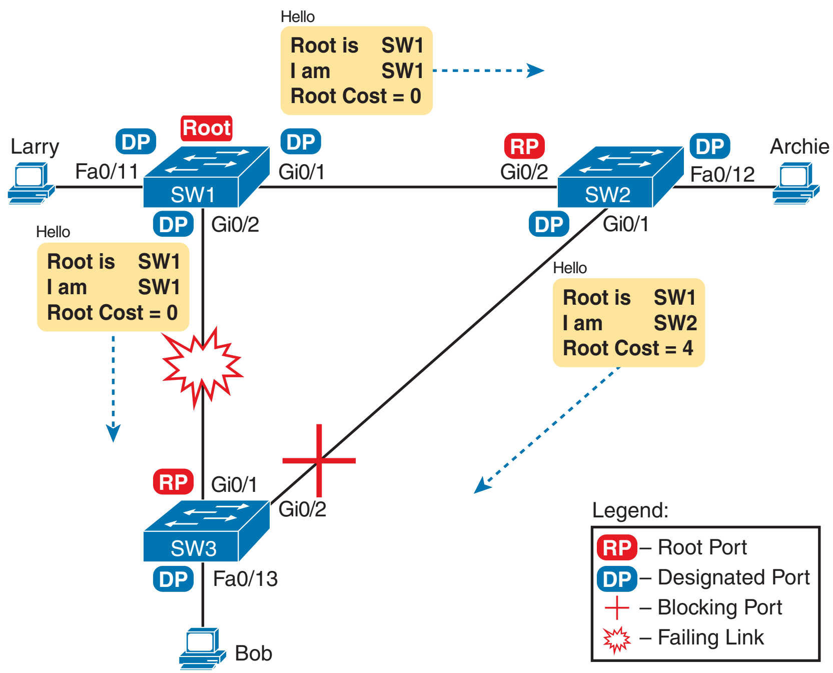


Figure 9-7 *Initial STP State Before SW1-SW3 Link Fails*

Next, SW3 reevaluates its choice of RP. At this point, SW3 is receiving Hellos on only one interface: Gi0/2. Whatever the calculated root cost, Gi0/2 becomes SW3's new RP. (The cost would be 8, assuming the same STP costs in earlier Figures 9-5 and 9-6.)

SW3 then reevaluates its role as DP on any other interfaces. In this example, no real work needs to be done. SW3 was already the DP on interface Fa0/13, and it continues to be the DP because no other switches connect to that port.

Changing Interface States with STP

STP uses the idea of roles and states. *Roles*, like root port role and designated port role, relate to how STP analyzes the LAN topology. *States*, like forwarding and blocking, tell a switch whether to send or receive frames. When STP converges, a switch chooses new port roles, and the port roles determine the state (forwarding or blocking).

Switches using STP can simply move immediately from a forwarding to a blocking state, but they must take extra time to transition from a blocking state to a forwarding state. For instance, when switch SW3 in Figure 9-7 formerly used port G0/1 as its RP (a role), that port was in a forwarding state. After convergence, G0/1 might be neither an RP nor a DP; the switch can immediately move that port to a blocking state.

However, when a port that formerly blocked needs to transition to forwarding, the switch first puts the port through two intermediate interface states. These temporary STP states help prevent temporary loops:

Key Topic

- **Listening:** Like the blocking state, the interface does not forward frames. The switch removes old stale (unused) MAC table entries for which no frames are received from each MAC address during this period. These stale MAC table entries could be the cause of the temporary loops.
- **Learning:** Interfaces in this state still do not forward frames, but the switch begins to learn the MAC addresses of frames received on the interface.

STP moves an interface from blocking to listening, then to learning, and then to a forwarding state. STP leaves the interface in each interim state for a time equal to the forward delay timer, which defaults to 15 seconds. As a result, a convergence event that causes an interface to change from blocking to forwarding requires 30 seconds to transition from blocking to forwarding. In addition, a switch might have to wait MaxAge seconds (default 20 seconds) before even choosing to move an interface from a blocking to a forwarding state.

For example, follow what happens with an initial STP topology as shown in Figures 9-3 through 9-6, with the SW1-to-SW3 link failing as shown in Figure 9-7. If SW1 simply quit sending Hello messages to SW3, but the link between the two did not fail, SW3 would wait MaxAge seconds before reacting (20 seconds is the default). SW3 would actually quickly choose its ports' STP roles, but then wait 15 seconds each in listening and learning states on interface Gi0/2, resulting in a 50-second convergence delay.

Table 9-8 summarizes spanning tree's various interface states for easier review.

Key Topic

Table 9-8 IEEE STP (not RSTP) States

State	Forwards Data Frames?	Learns MACs Based on Received Frames?	Transitory or Stable State?
Blocking	No	No	Stable
Listening	No	No	Transitory
Learning	No	Yes	Transitory
Forwarding	Yes	Yes	Stable
Disabled	No	No	Stable

Rapid STP Concepts

The original STP worked well given the assumptions about networks and networking devices in that era. However, as with any computing or networking standard, as time passes, hardware and software capabilities improve, so new protocols emerge to take advantage of those new capabilities. For STP, one of the most significant improvements over time has been the introduction of Rapid Spanning Tree Protocol (RSTP), introduced as standard IEEE 802.1w.

NOTE Just to make sure you are clear about the terminology: Throughout the rest of the chapter, *STP* refers to the original STP standard only, and use of the term *RSTP* does not include STP.

Before getting into the details of RSTP, it helps to make sense of the standards numbers a bit: 802.1w was actually an amendment to the 802.1D standard. The IEEE first published 802.1D in 1990, and anew in 1998. After the 1998 version of 802.1D, the IEEE published the 802.1w amendment to 802.1D in 2001, which first standardized RSTP.

Over the years, other meaningful changes happened in the standards as well, although those changes probably do not impact most networkers' thinking when it comes to working with STP or RSTP. But to be complete, the IEEE replaced STP with RSTP in the revised 802.1D standard in 2004. In another move, in 2011 the IEEE moved all the RSTP details into a revised 802.1Q standard. As it stands today, RSTP actually sits in the 802.1Q standards document.

As a result, when reading about RSTP, you will see documents, books, videos, and the like that refer to RSTP and include various references to 802.1w, 802.1D, and 802.1Q—and they might all be correct based on timing and context. At the same time, many people refer to RSTP as 802.1w because that was the first IEEE document to define it. However, for the purposes of this book, focus instead on the RSTP acronym rather than the IEEE standards numbers used with RSTP over its history.

NOTE The IEEE sells its standards, but through the "Get IEEE 802" program, you can get free PDFs of the current 802 standards. To read about RSTP today, you will need to download the 802.1Q standard and then look for the sections about RSTP.

Now on to the details about RSTP in this chapter. As discussed throughout this chapter, RSTP and STP have many similarities, so this section next compares and contrasts the two. Following that, the rest of this section discusses the concepts unique to RSTP that are not found in STP—alternate root ports, different port states, backup ports, and the port roles used by RSTP.

Comparing STP and RSTP

RSTP works just like STP in several ways, as discussed in the first major section of the chapter. To review:

Key Topic

- RSTP and STP elect the root switch using the same rules and tiebreakers.
- RSTP and STP switches select their root ports with the same rules.
- RSTP and STP elect designated ports on each LAN segment with the same rules and tiebreakers.
- RSTP and STP place each port in either forwarding or blocking state, although RSTP calls the blocking state the **discarding state**.

In fact, RSTP works so much like STP that they can both be used in the same network. RSTP and STP switches can be deployed in the same network, with RSTP features working in switches that support it and traditional STP features working in the switches that support only STP.

With all these similarities, you might be wondering why the IEEE bothered to create RSTP in the first place. The overriding reason is convergence. STP takes a relatively long time to converge (50 seconds with the default settings when all the wait times must be followed). RSTP improves network convergence when topology changes occur, usually converging within a few seconds (or in slow conditions, in about 10 seconds).

RSTP changes and adds to STP in ways that avoid waiting on STP timers, resulting in quick transitions from forwarding to discarding (blocking) state and vice versa. Specifically, RSTP, compared to STP, defines more cases in which the switch can avoid waiting for a timer to expire, such as the following:

Key Topic

- RSTP adds a mechanism by which a switch can replace its root port, without any waiting to reach a forwarding state (in some conditions).
- RSTP adds a new mechanism to replace a designated port, without any waiting to reach a forwarding state (in some conditions).
- RSTP lowers waiting times for cases in which RSTP must wait for a timer.

For instance, imagine a failure case in which a link remains up, but for some reason, a nonroot switch stops hearing the Hello BPDUs it had been hearing in the past. STP requires a switch to wait for MaxAge seconds, which STP defines based on ten times the Hello timer, or 20 seconds, by default. RSTP shortens this timer, defining MaxAge as three times the Hello timer. Additionally, RSTP can send messages to the neighboring switch to inquire whether a problem has occurred rather than wait for timers.

The best way to get a sense for these mechanisms is to see how the RSTP alternate port role and the backup port role both work. RSTP uses the term *alternate port* to refer to a switch's other ports that could be used as the root port if the root port ever fails. The *backup port* concept provides a backup port on the local switch for a designated port. (Note that backup ports apply only to designs that use hubs, so they are unlikely to be useful today.) However, both are instructive about how RSTP works. Table 9-9 lists these RSTP port roles.

Key Topic

Table 9-9 Port Roles in RSTP

Function	Port Role
Nonroot switch's best path to the root	**Root port**
Port that will be used to replace the root port when the root port fails	**Alternate port**
Switch port designated to forward onto a collision domain	**Designated port**
Port that will be used to replace a designated port when a designated port fails	**Backup port**
Port in a nonworking interface state—that is, anything other than connected (up/up)	**Disabled port**

RSTP differs from STP in a few other ways as well. For instance, with STP, the root switch creates a Hello, with all other switches updating and forwarding the Hello. With RSTP, each switch independently generates its own Hellos. Additionally, RSTP allows for queries between neighbors, rather than waiting on timers to expire, as a means to avoid waiting to learn information. These types of protocol changes help RSTP-based switches isolate what has changed in a network and react quickly to choose a net RSTP topology.

The next few pages work through some of those overt RSTP features that differ from STP.

RSTP and the Alternate (Root) Port Role

With STP, each nonroot switch places one port in the STP root port (RP) role. RSTP follows that same convention, with the same exact rules for choosing the RP. RSTP then takes another step beyond STP, naming other possible RPs, identifying them as *alternate ports*.

To be an alternate port, both the RP and the alternate port must receive Hellos that identify the same root switch. For instance, in Figure 9-8, SW1 is the root. SW3 will receive Hello BPDUs on two ports: G0/1 and G0/2. Both Hellos list SW1's bridge ID (BID) as the root switch, so whichever port is not the root port meets the criteria to be an alternate port. SW3 picks G0/1 as its root port in this case and then makes G0/2 an alternate port.

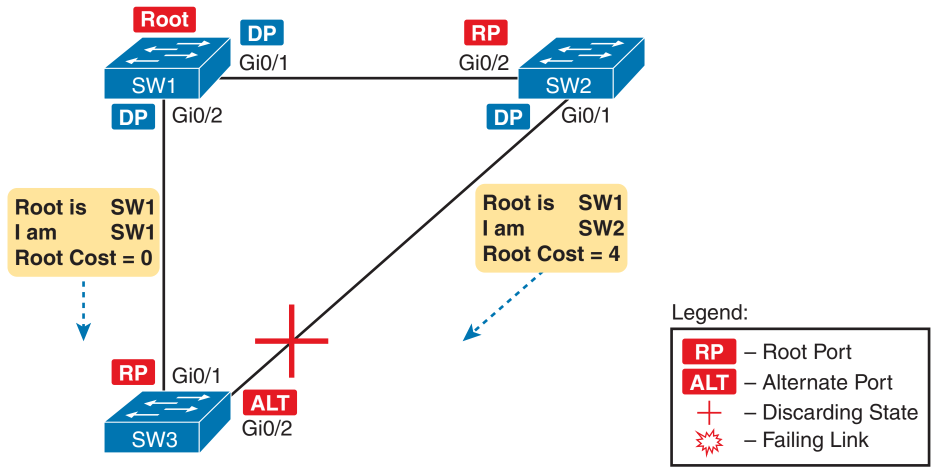


Figure 9-8 *Example of SW3 Making G0/2 Become an Alternate Port*

An alternate port basically works like the second-best option for the root port. The alternate port can take over for the former root port, often very rapidly, without requiring a wait in other interim RSTP states. For instance, when the root port fails, the switch changes the former root port's role and state: (a) the role from root port to a *disabled port* (because the interface failed), and (b) the state from forwarding to discarding (RSTP uses the discarding state for ports in the disabled port role). Then, without waiting on any timers, the switch changes the alternate port to be the root port and immediately changes its port state from discarding to forwarding.

Notably, the new root port also does not need to spend time in other states, such as the learning state, instead moving immediately to the forwarding state.

Figure 9-9 shows an example of RSTP convergence. SW3's root port before the failure shown in this figure is SW3's G0/1, the link connected directly to SW1 (the root switch). Then SW3's link to SW1 fails as shown in Step 1 of the figure.

Following the steps in Figure 9-9:

Step 1. The link between SW1 and SW3 fails, so SW3's current root port (Gi0/1) fails.

Step 2. SW3 and SW2 exchange RSTP messages to confirm that SW3 will now transition its former alternate port (Gi0/2) to be the root port. This action causes SW2 to flush the required MAC table entries.

Step 3. SW3 transitions G0/1 to the disabled port role and G0/2 to the root port role.

Step 4. SW3 transitions G0/2 to a forwarding state immediately, without using the learning state, because this is one case in which RSTP knows the transition will not create a loop.

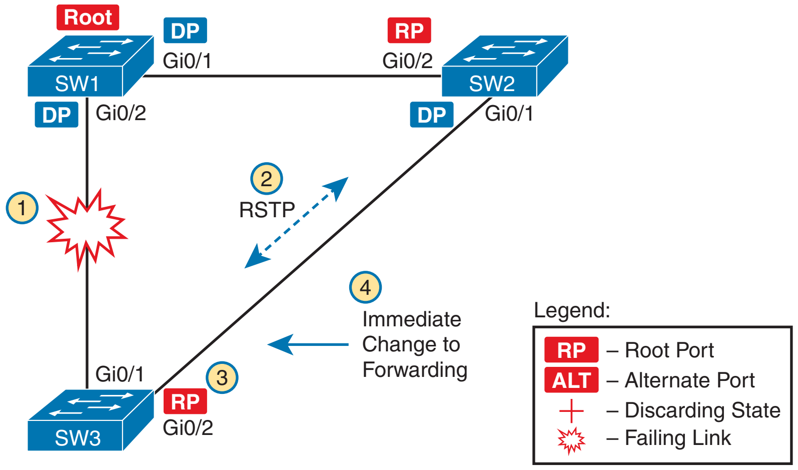


Figure 9-9 *Convergence Events with SW3 G0/1 Failure*

As soon as SW3 realizes its G0/1 interface has failed, the process shown in the figure takes very little time. None of the processes rely on timers, so as soon as the work can be done, the convergence completes. (This particular convergence example takes about 1 second in a lab.)

RSTP States and Processes

The depth of the previous example does not point out all the details of RSTP, of course; however, the example does show enough details to discuss RSTP states and internal processes.

Both STP and RSTP use *port states*; however, RSTP differs from STP in a few of the details. Both use the forwarding state for the same purpose. For the two interim states used by STP during convergence (listening and learning), RSTP does not use the listening state, considering it unnecessary, but it uses the learning state for the same purpose as STP. Finally, RSTP uses the *discarding* state instead of the STP *blocking* state.

To be complete, note that both RSTP and STP have port roles and states related to ports that are not currently working. Those rules are

- Both STP and RSTP define and use a disabled port role for any port in a nonworking interface state. (For instance, interfaces with no cable installed or interfaces configured with the **shutdown** command configured.)
- STP uses the disabled state for such ports. RSTP does not have a separate disabled state but instead uses the discarding state.

Honestly, the intricacies of the disabled role and states probably do not matter much; it seems obvious that nonworking interfaces cannot be a useful part of a working spanning tree. Regardless, Table 9-10 shows the list of STP and RSTP states for comparison and easier study.

Key Topic

Table 9-10 Port States Compared: STP and RSTP

Function	STP State	RSTP State
Port is not in a working (connected) state, either due to failure or due to shutdown	Disabled	Discarding
Stable state that ignores incoming data frames and is not used to forward data frames	Blocking	Discarding
Interim state without MAC learning and without forwarding	Listening	Not used
Interim state with MAC learning and without forwarding	Learning	Learning
Stable state that allows MAC learning and forwarding of data frames	Forwarding	Forwarding

RSTP uses different internal processes compared to STP in an effort to speed convergence. As an example, consider what happens when STP changes a port's role, resulting in the need to move from a blocking to a forwarding state:

1. The switch changes the timer it uses to timeout MAC table entries to be equal to the STP forward delay timer (default 15 seconds).
2. The switch transitions the interface to the listening state, remaining there for the forward delay time (default 15 seconds).
3. After the forward delay time passes:
   a. All MAC table entries affected by this topology change will have been timed out of the table, removing the possibility of loops.
   b. The switch can now transition the port to the learning state, remaining there for the forward delay time.
4. After spending the forward delay time in the learning state, the switch transitions the port to the forwarding state.

To converge more quickly, RSTP avoids relying on timers. RSTP switches tell each other (using messages) that the topology has changed. Those messages also direct neighboring switches to flush the contents of their MAC tables in a way that removes all the potentially loop-causing entries, without a wait. As a result, RSTP creates more scenarios in which a formerly discarding port can immediately transition to a forwarding state, without waiting, and without using the learning state, as shown in the example in Figure 9-9.

RSTP and the Backup (Designated) Port Role

The RSTP backup port role also improves convergence. It provides fast failover for the designated port role, but only in designs like Figure 9-10, when one switch connects with multiple ports to the same hub. Because moderns LANs do not use hubs, you will be unlikely to see this RSTP feature in practice, but to be ready for all RSTP features, consider this example.

Figure 9-10 shows a design in which switch SW3 connects with one port and SW4 connects with two ports to the same hub. SW4's port F0/1 happens to win the election as designated port (DP). In this topology, switch SW4's port F0/2 can act as a backup port.

With a backup port, if the current designated port fails, SW4 can start using the backup port as the new designated port with rapid convergence. For instance, if SW4's F0/1 interface were to fail, SW4 could transition F0/2 to the designated port role, without any delay in moving from the discarding state to a forwarding state.

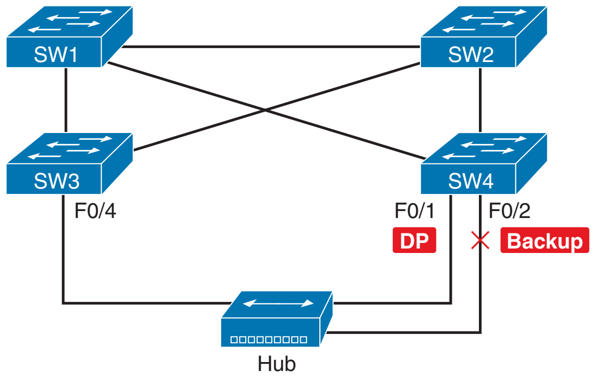


Figure 9-10 *RSTP Backup Port Example*

RSTP Port Types

The final RSTP concept included here relates to some terms RSTP uses to refer to different types of ports and the links that connect to those ports.

To begin, consider the basic image in Figure 9-11. It shows several links between two switches. RSTP considers these links to be point-to-point links and the ports connected to them to be point-to-point ports because the link connects exactly two devices (points).

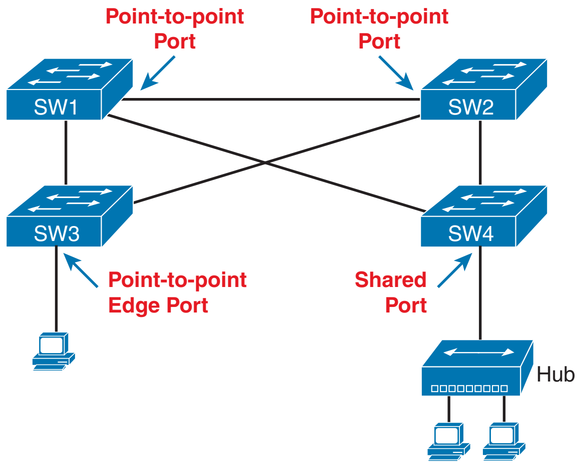


Figure 9-11 *RSTP Link Types*

RSTP further classifies point-to-point ports into two categories. Point-to-point ports that connect two switches are not at the edge of the network and are simply called *point-to-point ports*. Ports that instead connect to a single endpoint device at the edge of the network, like a PC or server, are called *point-to-point edge ports*, or simply *edge ports*. In Figure 9-11, SW3's switch port connected to a PC is an edge port.

Finally, RSTP defines the term *shared* to describe ports connected to a hub. The term *shared* comes from the fact that hubs create a shared Ethernet; hubs also force the attached switch port to use half-duplex logic. RSTP assumes that all half-duplex ports may be connected to hubs, treating ports that use half duplex as shared ports. RSTP converges more slowly on shared ports as compared to all point-to-point ports.

Optional STP Features

To close out the chapter, the last major section describes optional features that make STP work even better or be more secure: EtherChannel, PortFast, BPDU Guard, Root Guard, and Loop Guard.

EtherChannel

One of the best ways to lower STP's convergence time is to avoid convergence altogether. **EtherChannel** provides a way to prevent STP convergence from being needed when only a single port or cable failure occurs.

EtherChannel combines multiple parallel segments of equal speed (up to eight) between the same pair of switches, bundled into an EtherChannel. The switches treat the EtherChannel as a single interface with regard to STP. As a result, if one of the links fails, but at least one of the links is up, STP convergence does not have to occur. For example, Figure 9-12 shows the familiar three-switch network, but now with two Gigabit Ethernet connections between each pair of switches.

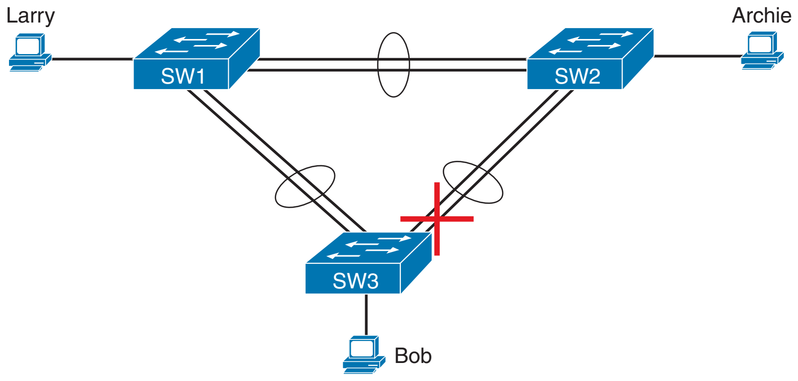


Figure 9-12 *Two-Segment EtherChannels Between Switches*

With each pair of Ethernet links configured as an EtherChannel, STP treats each EtherChannel as a single link. In other words, both links to the same switch must fail for a switch to need to cause STP convergence. Without EtherChannel, if you have multiple parallel links between two switches, STP blocks all the links except one. With EtherChannel, all the parallel links can be up and working simultaneously while reducing the number of times STP must converge, making the network more available.

The current CCNA exam blueprint includes a topic for configuring both Layer 2 EtherChannels (as described here) and Layer 3 EtherChannels. Chapter 10, "RSTP and EtherChannel Configuration," shows how to configure Layer 2 EtherChannels, while Chapter 18, "IP Routing in the LAN," shows how to configure Layer 3 EtherChannels. Note that Layer 2 EtherChannels combine links that switches use as switch ports, with the switches using Layer 2 switching logic to forward and receive Ethernet frames over the EtherChannels. Layer 3 EtherChannels also combine links, but the switches use Layer 3 routing logic to forward packets over the EtherChannels.

PortFast

Switch ports that connect directly to endpoints rather than to other switches eventually use the designated port (DP) role and a forwarding state. However, with default port settings, they go through some interim steps:

1. After the interface reaches the connected (up/up) state, the switch STP logic initially places the port in the STP discarding (blocking) state.
2. The switch begins sending BPDUs out of the port and listens for incoming BPDUs, to decide whether the switch port should take on a root port (RP) or a DP role.
3. Because the connected device is not a switch, the local switch receives no incoming BPDUs. As a result, the switch port wins the DP election and becomes a DP.
4. The switch takes interim steps to transition to a forwarding state:
   a. STP moves the port first to the listening state, then to the learning state, and then to the forwarding state. The interim states require 15 seconds each by default (per the default Forward Delay timer).
   b. RSTP follows the same process, except it has no listening state. It moves from discarding to learning to forwarding.

PortFast bypasses the process. After the interface reaches a connected state, STP PortFast immediately moves the port to the DP role and the forwarding state without delay. As long as the interface remains up, the STP role remains as DP with a forwarding state.

PortFast exists to support access links connected to endpoints, as shown in Figure 9-13. You know the switch always wins the DP election on those links because the endpoints do not use STP. Those endpoints can include single computers, phones with connected computers, or a trunk connected to a server.

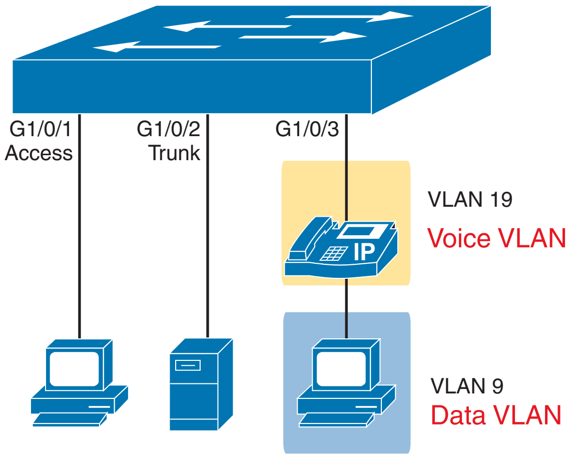


Figure 9-13 *Access Ports Appropriate for the PortFast Feature*

As you might guess from the fact that PortFast speeds convergence, STP did not include the PortFast feature, but RSTP included it as one of its rapid convergence optimizations. You might recall the mention of RSTP port types, particularly point-to-point edge port types, around Figure 9-11. RSTP, by design of the protocol, converges quickly on these ports of type point-to-point edge by bypassing the learning state. Note that Cisco introduced this idea with its PortFast years before the IEEE finalized RSTP, which is why we still call the feature PortFast many decades later. In practice, Cisco switches enable RSTP point-to-point edge ports by enabling PortFast on the port.

While very useful and popular, PortFast also introduces the risk of creating loops. PortFast ports should never connect to bridges or switches. With PortFast enabled, the port ignores incoming BPDUs, always acts as a designated port, and always forwards. As a result, an unexpected or rogue switch connected to a PortFast-enabled port can create a forwarding loop.

BPDU Guard

PortFast creates an exposure: the possibility of a forwarding loop if an unexpected switch connects to a PortFast port. The Cisco **BPDU Guard** feature helps defeat these kinds of problems by disabling a port if it receives any BPDUs in the port. So, this feature is handy on ports that should be used only as an access port and never connected to another switch.

Figure 9-14 shows the basic idea. The device on the right should be an endpoint device such as a PC. Instead, a new switch named Rogue-1 connects to the legitimate switch SW3—a switch that uses BPDU Guard to protect against this scenario. At Step 2, as soon as switch Rogue-1 sends a BPDU into SW3, SW3's BPDU Guard logic reacts, disabling the port.

Key Topic

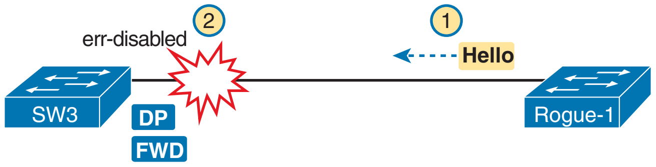


Figure 9-14 *Basic Logic of BPDU Guard*

Most switch access port configurations combine PortFast and BPDU Guard, although both can be used independently. Using PortFast without protection against the scenario in Figure 9-14 risks a loop. Using BPDU Guard on these same ports makes sense because if another switch connects to such a port, the local switch can disable the port to prevent a loop.

BPDU Filter

The **BPDU Filter** feature defines two different functions under one named feature. Both fall under the name BPDU filter because, behind the scenes, they filter (discard) STP BPDUs.

First, the BPDU Filter feature can be used as an alternative to BPDU Guard when used along with PortFast. Both BPDU Guard and BPDU Filter prevent loops when unexpected BPDUs arrive in a PortFast-enabled port. However, BPDU Filter reacts to those incoming BPDUs by disabling PortFast logic (which creates the possibility of a forwarding loop) and restoring normal STP logic on the port (which prevents the possibility of a forwarding loop).

The other BPDU Filter feature disables STP on an interface. When enabled, BPDU Filter discards all sent and received BPDUs on an interface.

The following pages take a closer look at each logic branch.

BPDU Filter to Prevent Loops on PortFast Ports

Most campus LANs make widespread use of PortFast, both on access ports and on server ports that use trunking. However, with PortFast alone but with no other features, you risk someone unplugging those devices and replacing them with switches to create a forwarding loop. Figure 9-15 shows such a case, where switch SW1's two ports use PortFast but the endpoints were replaced with two rogue switches.

Key Topic

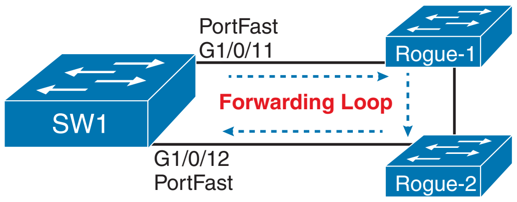


Figure 9-15 *Possible Loop on PortFast Ports with Rogue Switches*

To protect against such a case, you can use either BPDU Guard or BPDU Filter. Ports that use PortFast should not receive BPDUs under normal circumstances, but nothing prevents people from replacing devices as shown in Figure 9-15. If that happens, BPDU Filter reacts by disabling PortFast on the port. The port then reverts to normal STP rules. STP will then determine a new port role and status as needed, use STP rules, and prevent any forwarding loop. The port may block or forward, depending on the results of that normal STP logic.

The previous paragraph gets at the reason why an engineer might choose this BPDU Filter feature; however, the following list details the specific steps, which includes some BPDU filtering.

Key Topic

1. To enable this branch of BPDU Filter logic, use the related global command **spanning-tree portfast bpdufilter default**.
2. IOS finds all ports that currently use PortFast and enables conditional BPDU Filter on those ports.
3. When any such port comes up:
   a. BPDU Filter allows STP to send BPDUs as normal for 10 × Hello Time (2 × 10 seconds, or 20 seconds total by default).
   b. If the port receives zero BPDUs in that time, BPDU Filter begins filtering (discarding) sent BPDUs.
4. BPDU Filter monitors to notice any incoming BPDUs. If they begin to arrive:
   a. It disables PortFast on the port.
   b. It reverts the port to use STP with normal STP rules.

BPDU Filter to Disable STP on a Port

The BPDU Filter feature can be configured to enable a completely different style of logic. As you will see in Chapter 10's section titled "BPDU Filter," the configuration does not make it obvious which style the configuration enables. However, when enabled with an interface subcommand, BPDU Filter does just what the name says: It filters (discards) all BPDUs on the ports, both outgoing and incoming.

Using BPDU Filter in this way effectively disables STP on the port. Using this feature is dangerous: Do not use it in production until you fully test and understand it.

As an example of why you might want to use BPDU Filter to disable STP on a port, consider the scenario in Figure 9-16. Two IT groups build networks inside the same company. They want to connect their LANs together, but the two groups (Left and Right) do not want the other group's STP to influence theirs—for example, which switch would be allowed to win the root election, a left or a right switch? The two groups meet and decide to keep their STP topologies separate. To do so, they filter all BPDUs sent between the two using BPDU Filter, disabling STP on the one link that connects the two LANs.

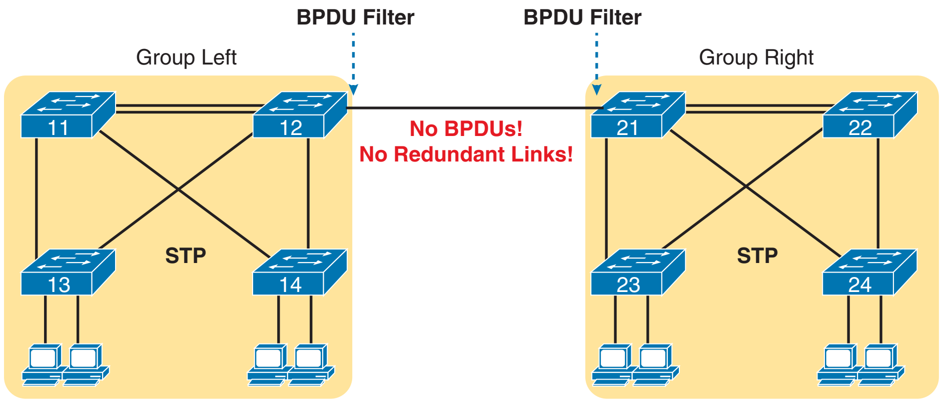


Figure 9-16 *A Scenario to Consider Disabling STP on a Link (BPDU Filtering)*

Root Guard

The **Root Guard** feature monitors incoming BPDUs on a port and reacts if the received BPDU would change the root switch in the VLAN. To understand why that might be useful, consider the topology with six switches in Figure 9-17. The notation by each switch defines how good the priority is on each switch. If you configure the switches with different priorities, you know and control which switches will become the root switch, with switches SW5 and SW6 being the least likely.

Key Topic

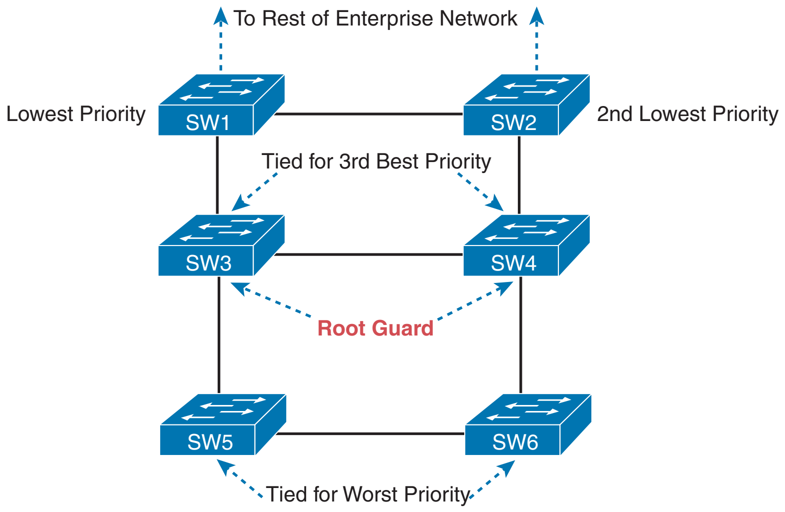


Figure 9-17 *Larger LAN Showing Most and Least Likely Root Switches*

Now imagine that your organization planned the design in the figure, but you directly control switches SW1 through SW4 while another organization controls switches SW5 and SW6. Both groups agree that SW1 should be the root switch, with SW2 taking over if SW2 fails; however, that agreement does not prevent a configuration mistake on switch SW5 or SW6.

For example, someone in the other group could configure switch SW5 with a priority lower than SW1's priority, and SW5 would become the root switch, creating a suboptimal STP topology. Switch SW5 would send a superior BPDU into the network if such an unplanned configuration occurred. With this topology, SW3 would receive the superior BPDU. Similarly, SW4 might receive a superior BPDU from SW6 if someone lowered switch SW6's priority.

Root Guard, enabled per port, enables the port to operate normally with STP except in this one special case: when it receives a **superior BPDU** in the port. So, you enable Root Guard on ports connected to other switches, usually trunks, and STP works normally; however, Root Guard on a port also prevents the election of a different root switch that is reachable through that port. Root Guard achieves that by listening for incoming Hellos on the port, and if it is a superior Hello, the switch disables the port by using a special STP port state (the **broken state**). Root Guard recovers the port automatically when the superior Hellos no longer occur.

The choice to use Root Guard begins with a close analysis of the topology. You must agree which switches should never become root and which ports should never receive superior BPDUs.

Loop Guard

The final optional tool mentioned here, Loop Guard, protects against a specific case in an STP topology. This feature helps protect the worst priority switch (highest BID) in a typical STP design, such as switch SW3 in Figure 9-18. The mechanisms apply to regular operation in a typical STP design, forcing some review of topics from earlier in the chapter.

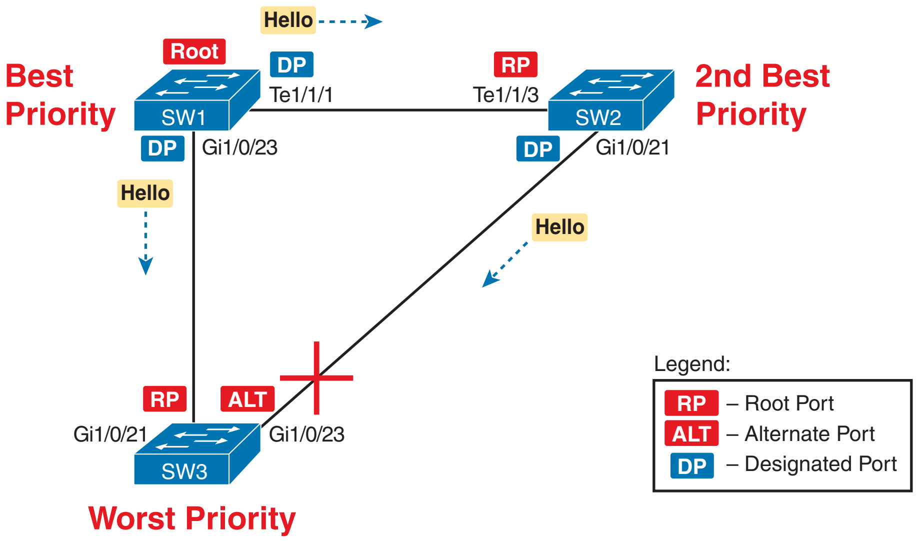


Figure 9-18 *Typical STP Design*

A typical STP design uses one switch as the known root under normal circumstances. To achieve that, you assign that switch the lowest priority. Another switch has the role of being the best backup, so you assign it the second-best priority value. Switches SW1 and SW2 play those roles in Figure 9-18. All the other switches have a lower priority than SW1 and SW2, such as switch SW3.

Next, consider an observation about this design, about the worst switch (SW3). SW3's ports on its switch-to-switch links seldom become designated ports. Why? STP determines the DP based on lowest root cost of the switches on the link, and breaks ties based on the better (lower) BID. SW3 loses the DP election tiebreaker in this scenario.

However, SW3's ports have other roles, just not the DP role, as follows:

- One link will be its root port (G1/0/21 in Figure 9-18) and settle into a forwarding state.
- The other will be an alternate port (G1/0/23 in Figure 9-18) and settle into a discarding state.

The second meaningful observation is that the worst switch (SW3) receives repeated Hellos in both switch-to-switch links. The figure shows the Hellos. The rule is that switches forward Hellos out designated ports every Hello time (the default is 2 seconds).

The third fact that helps you understand Loop Guard involves unidirectional links. Figure 9-19 shows the idea. The interfaces on the ends of a fiber link will be up and in a connected state, but one of the two fibers has a problem. The problem could prevent frames from arriving at the other end of the fiber. Figure 9-19 shows such a case, with only the right-to-left direction working.

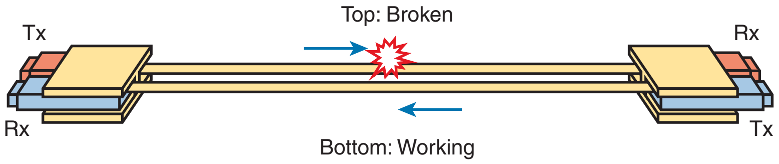


Figure 9-19 *Unidirectional Fiber Link*

To summarize, Loop Guard relies on these underlying facts:

- A switch with the worst (highest) priority in a design has switch-to-switch ports in the root port (RP) and alternate port (ALT) roles but seldom the designated port (DP) role.
- Those same ports normally receive Hellos every Hello time because each link's neighboring switch is the DP.
- A **unidirectional link** can occur, in which the interface state remains up (connected) on both ends but frames cannot flow in one direction.

The **Loop Guard** feature takes advantage of the above observations to protect against a class of failures that, without Loop Guard, results in one of switch SW3's switch-switch links becoming a DP. Figure 9-20 shows an adjustment to Figure 9-18 for one scenario, with the SW2-SW3 link becoming unidirectional:

1. The SW2-SW3 link's fiber for transmitting from SW2 to SW3 fails.
2. Because SW2's G1/0/21 and SW3's G1/0/23 ports remain in a connected state, there is no interface failure to influence STP.
3. SW3 ceases to receive incoming Hello messages on its G1/0/23 port.
4. After the appropriate timeouts, SW3 believes it is the only switch on the link connected to its G1/0/23 interface, so it changes the port to a DP and moves toward a forwarding state.

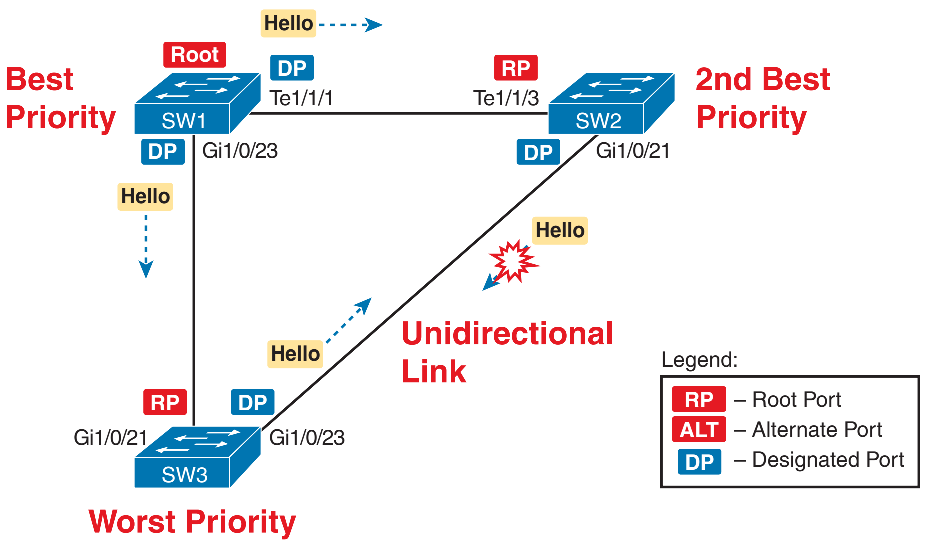


Figure 9-20 *SW3's ALT Port Becomes DP, Forwards; All Links Forward, Create Loop*

Take a moment and look at the ends of the switch-to-switch links. All ports are either an RP or DP, both of which use a forwarding state. As a result, a forwarding loop exists. Note that with the unidirectional link between switches SW2 and SW3, broken in the direction from SW2 to SW3, the forwarding loop exists only in the opposite (counter-clockwise) direction.

Finally, you can now appreciate the Loop Guard feature. You enable it on an interface and it takes the following actions:

Key Topic

If the port is a root or alternate port, prevent it from becoming a designated port by moving it to the special broken STP state.

As with Root Guard, you must work through the underlying concepts to decide the ports on which to enable Loop Guard. Generally, you need it only on fiber-optic links connected to other switches. You should also only choose ports on switches with a poor (high) STP priority, so they normally have switch-to-switch ports in the RP or ALT roles but not the DP role.

Chapter Review

One key to doing well on the exams is to perform repetitive spaced review sessions. Review this chapter's material using either the tools in the book or interactive tools for the same material found on the book's companion website. Refer to the "Your Study Plan" element for more details. Table 9-11 outlines the key review elements and where you can find them. To better track your study progress, record when you completed these activities in the second column.

Table 9-11 Chapter Review Tracking

Review Element	Review Date(s)	Resource Used
Review key topics		Book, website
Review key terms		Book, website
Answer DIKTA questions		Book, PTP
Review memory tables		Website

Review All the Key Topics

Key Topic

Table 9-12 Key Topics for Chapter 9

Key Topic Element	Description	Page Number
Table 9-2	Lists the three main problems that occur when not using STP in a LAN with redundant links	227
Figure 9-2	How STP blocks to break a loop	227
Table 9-3	Lists the reasons why a switch chooses to place an interface into forwarding or blocking state	229
Table 9-4	Lists the most important fields in Hello BPDU messages	230
List	Logic for the root switch election	231
Figure 9-6	Shows how switches calculate their root cost	233
Table 9-6	Lists the original and current default STP port costs for various interface speeds	236
Step list	A summary description of steady-state STP operations	236
Table 9-7	STP timers	237
List	Definitions of what occurs in the listening and learning states	239
Table 9-8	Summary of STP and RSTP states	239
List	Key similarities between 802.1D STP and 802.1w RSTP	240
List	RSTP mechanisms for faster convergence compared to STP	241
Table 9-9	List of 802.1w port roles	241
Table 9-10	Comparisons of port states with 802.1D and 802.1w	244
Figure 9-14	Basic logic for BPDU Guard	248
Figure 9-15	An example forwarding loop risk with PortFast	249
List	Conditional BPDU Filter logic applied to PortFast ports	249
Figure 9-17	Locations to apply Root Guard	250
Paragraph	Loop Guard rules	253

Key Terms You Should Know

alternate port (role), backup port (role), blocking state, BPDU Filter, BPDU Guard, bridge ID, bridge protocol data unit (BPDU), broken state, designated port, designated port (role), disabled port (role), disabled state, discarding state, EtherChannel, forward delay, forwarding state, Hello BPDU, learning state, listening state, Loop Guard, MaxAge, PortFast, Rapid STP (RSTP), root cost, Root Guard, root port (role), root switch, Spanning Tree Protocol (STP), superior BPDU, unidirectional link

CHAPTER 10

RSTP and EtherChannel Configuration

This chapter covers the following exam topics:

2.0 Network Access

2.4 Configure and verify (Layer 2/Layer 3) EtherChannel (LACP)

2.5 Interpret basic operations of Rapid PVST+ Spanning Tree Protocol

2.5.a Root port, root bridge (primary/secondary), and other port names

2.5.b Port states and roles

2.5.c PortFast

2.5.d Root guard, loop guard, BPDU filter, and BPDU guard

This chapter shows how to configure Rapid Spanning Tree Protocol (RSTP) and Layer 2 EtherChannels.

The first two sections take a little different approach than many other CLI-focused parts of the book, based on exam topic 2.5. That exam topic begins with the phrase "interpret basic operations," which emphasizes the concepts behind what happens in the CLI. The first major section examines STP concepts using some configuration, with much focus on interpreting **show** command output. The second major section, about exam topics 2.5.c and 2.5.d, takes the same approach.

The EtherChannel content, in the final major section of the chapter, follows a typical flow for most configuration/verification topics in a certification guide: it reviews concepts, shows configurations, followed by **show** commands that point out the configuration settings and operational state. The details include how to manually configure a channel, how to cause a switch to dynamically create a channel, and how EtherChannel load distribution works.

I encourage you to practice and experiment with STP from the CLI. If you're looking for lab exercises, check out the Config Labs at my blog site related to this chapter, as detailed in the introduction to this book. (Or, just go to certskills.com/config-labs for more info.)

Finally, like Chapter 9, this chapter is longer than I prefer. If you like to think of each chapter as one study session, you might need to think about splitting this chapter into two study sessions. Stop the first study session just as you reach the third major section, titled "Configuring Layer 2 EtherChannel." The second study session would consist of the "Configuring Layer 2 EtherChannel" section.

"Do I Know This Already?" Quiz

Take the quiz (either here or use the PTP software) if you want to use the score to help you decide how much time to spend on this chapter. The letter answers are listed at the bottom of the page following the quiz. Appendix C, found both at the end of the book as well as on the companion website, includes both the answers and explanations. You can also find both answers and explanations in the PTP testing software.

Table 10-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
Understanding RSTP Through Configuration	1–3
Identifying Optional STP Features	4–5
Configuring Layer 2 EtherChannel	6–8

1. Which type value on the **spanning-tree mode** *type* global command enables the use of RSTP?
   a. **rapid-pvst**
   b. **pvst**
   c. **rstp**
   d. **rpvst**
2. Examine the following output from the **show spanning-tree vlan 5** command, which describes a root switch in a LAN. Which answers accurately describe facts related to the root's bridge ID? (Choose two answers.)

```
SW1# show spanning-tree vlan 5

VLAN0005
   Spanning tree enabled protocol rstp
   Root ID Priority 32773
            Address      1833.9d7b.0e80
            Cost         15
            Port         25 (GigabitEthernet0/1)
            Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
```

   a. The system ID extension value, in decimal, is 5.
   b. The root's configured priority value is 32773.
   c. The root's configured priority value is 32768.
   d. The system ID extension value, in hexadecimal, is 1833.9d7b.0e80.
3. With Cisco's RPVST+, which of the following actions does a switch take to identify which VLAN is described by a BPDU? (Choose three answers.)
   a. Adds a VLAN tag when forwarding a BPDU on trunks
   b. Adds the VLAN ID in an extra TLV in the BPDU
   c. Lists the VLAN ID as the middle 12 bits of the System ID field of the BPDU
   d. Lists the VLAN ID in the System ID Extension field of the BPDU

4. A switch port in access mode connects to a single laptop. An attacker replaces the laptop with a LAN switch that uses STP. Which combination of features, enabled on the access port, results in the switch changing the port's interface state to err-disabled? (Choose two answers.)
   a. PortFast enabled but BPDU Guard not enabled
   b. PortFast and BPDU Guard enabled
   c. PortFast disabled but BPDU Guard enabled
   d. PortFast and BPDU Guard disabled
5. Root Guard has acted to disable port G1/0/1. An engineer uses the **show interfaces status** and **show spanning-tree vlan 1** commands to investigate the current status. Which answers list facts expected due to Root Guard's actions? (Choose two answers.)
   a. The STP port state per **show spanning-tree vlan 1** shows BKN (broken).
   b. The interface state per **show interfaces status** shows err-disabled.
   c. The STP port state per **show spanning-tree vlan 1** shows FWD (forwarding).
   d. The STP port state per **show spanning-tree vlan 1** shows BLK (blocking).
   e. The interface state per **show interfaces status** shows connected.
6. An engineer configures a switch to put interfaces G0/1 and G0/2 into the same Layer 2 EtherChannel. Which of the following terms is used in the configuration commands?
   a. **EtherChannel**
   b. **PortChannel**
   c. **Ethernet-Channel**
   d. **Channel-group**
7. Which combinations of keywords on the **channel-group** interface subcommand on two neighboring switches will cause the switches to use LACP and attempt to add the link to the EtherChannel? (Choose two answers.)
   a. **desirable** and **active**
   b. **passive** and **active**
   c. **active** and **auto**
   d. **active** and **active**
8. A Cisco Catalyst switch needs to send frames over a Layer 2 EtherChannel. Which answer best describes how the switch balances the traffic over the four active links in the channel?
   a. Breaks each frame into fragments of approximately one-fourth of the original frame, sending one fragment over each link
   b. Sends the entire frame over one link, alternating links in sequence for each successive frame
   c. Sends the entire frame over one link, choosing the link by applying some math to fields in each frame's headers
   d. Sends the entire frame over one link, using the link with the lowest percent utilization as the next link to use

Foundation Topics

Understanding RSTP Through Configuration

Cisco IOS switches today typically default to using RSTP rather than STP, with default settings so that RSTP works with no configuration. You can buy some Cisco switches and connect them with Ethernet cables in a redundant topology, and RSTP will ensure that frames do not loop. And even if some switches use RSTP and some use STP, the switches can interoperate and still build a working spanning tree—and you never even have to think about changing any settings!

Although RSTP works without any configuration, most medium-size to large-size campus LANs benefit from some RSTP configuration. For instance, Figure 10-1 shows a typical LAN design model, with two distribution layer switches (D1 and D2). The design may have dozens of access layer switches that connect to end users; the figure shows just three access switches (A1, A2, and A3). For a variety of reasons, most network engineers make the distribution layer switches be the root.

Key Topic

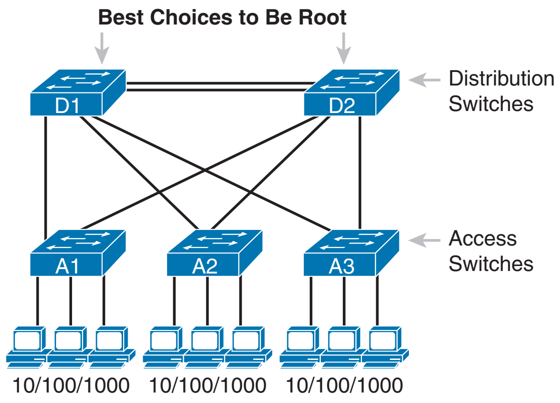


Figure 10-1 *Typical Configuration Choice: Making Distribution Switch Be Root*

NOTE Cisco uses the term *access switch* to refer to switches used to connect to endpoint devices. The term *distribution switch* refers to switches that do not connect to endpoints but rather connect to each access switch, providing a means to distribute frames throughout the LAN. The term *uplink* refers to the switch-to-switch links, usually trunks, between access and distribution switches. If you want to read more about LAN design concepts and terms, refer to the *CCNA 200-301 Official Cert Guide, Volume 2*, Second Edition, Chapter 17, "LAN Architecture."

As discussed in the introduction to this chapter, this first section of the chapter examines a variety of STP/RSTP configuration topics, but with a goal of revealing a few more details about how STP/RSTP operate. Following this opening section about STP/RSTP configuration, the next section examines how to configure Layer 2 EtherChannels and how that impacts STP/RSTP.

The Need for Multiple Spanning Trees

The IEEE first standardized STP as the IEEE 802.1D standard, first published back in 1990. To put some perspective on that date, Cisco did not have a LAN switch product line at the time, and virtual LANs did not exist yet. Instead of multiple VLANs in a physical Ethernet LAN, the physical Ethernet LAN existed as one single broadcast domain, with one instance of STP.

By the mid-1990s, VLANs had appeared on the scene, along with LAN switches. The emergence of VLANs posed a challenge for STP—the only type of STP available at the time—because STP defined a single common spanning tree (CST) topology for the entire LAN. The IEEE needed an option to create multiple spanning trees so that traffic could be balanced across the available links, as shown in Figure 10-2. With two different STP instances, SW3 could block on a different interface in each VLAN, as shown in the figure.

Key Topi

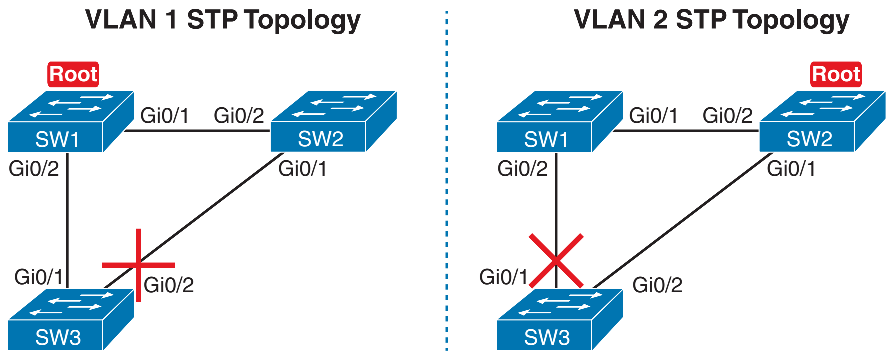


Figure 10-2 *Load Balancing with One Tree for VLAN 1 and Another for VLAN 2*

STP Modes and Standards

Because of the sequence of events over the history of the various STP family of protocols, vendors like Cisco needed to create their own proprietary features to create the per-VLAN spanning tree concept shown in Figure 10-2. That sequence resulted in the following:

- When STP was the only STP standard back in the 1990s with 802.1D, Cisco created the STP-based Per VLAN Spanning Tree Plus (**PVST+**) protocol, which creates one spanning tree instance per VLAN.
- When the IEEE introduced RSTP (in 802.1D amendment 802.1w, in 2001), they again defined it as a means to create a single spanning tree.
- When Cisco added support for RSTP to its switches, it created the **Rapid PVST+** (RPVST+) protocol. RPVST+ provided more features than standardized RSTP, including one tree per VLAN.
- To create multiple spanning trees, the IEEE did not adopt Cisco's PVST+ or RPVST+. Instead, the IEEE created a different method: Multiple Spanning Tree Protocol (MSTP), originally defined in 802.1Q amendment 802.1s.

Answers to the "Do I Know This Already?" quiz:

1 A **2** A, C **3** A, B, D **4** B, C **5** A, E **6** D **7** B, D **8** C

Figure 10-3 shows the features as a timeline for perspective.

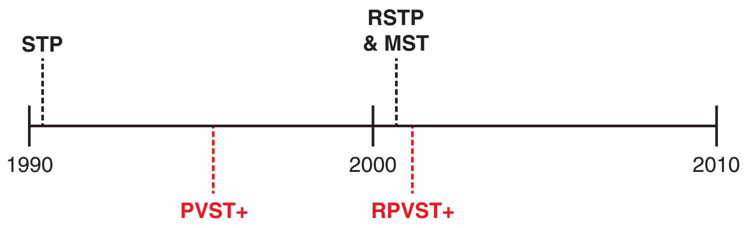


Figure 10-3 *Timeline of Per-VLAN and Multiple STP Features*

Today, Cisco Catalyst switches give us three options to configure on the **spanning-tree mode** command, which tells the switch which type of STP to use. Note that the switches do not support STP or RSTP with the single tree (CST). They can use either the Cisco-proprietary and STP-based PVST+, Cisco-proprietary and RSTP-based RPVST+, or the IEEE standard MSTP. Table 10-2 summarizes some of the facts about these standards and options, along with the keywords used on the **spanning-tree mode** global configuration command. Example 10-1, which follows, shows the command options in global configuration mode.

Key Topic

Table 10-2 STP Standards and Configuration Options

Name	Based on STP or RSTP?	# Trees	Original IEEE Standard	Config Parameter
STP	STP	1 (CST)	802.1D	N/A
PVST+	STP	1/VLAN	802.1D	**pvst**
RSTP	RSTP	1 (CST)	802.1w	N/A
Rapid PVST+	RSTP	1/VLAN	802.1w	**rapid-pvst**
MSTP	RSTP	1 or more*	802.1s	**mst**

* MSTP allows the definition of as many instances (multiple spanning tree instances, or MSTIs) as chosen by the network designer but does not require one per VLAN.

Example 10-1 *Cisco Switch Spanning Tree modes*

```
SW1(config)# spanning-tree mode ?
  mst          Multiple spanning tree mode
  pvst         Per-Vlan spanning tree mode
  rapid-pvst   Per-Vlan rapid spanning tree mode
SW1(config)#
```

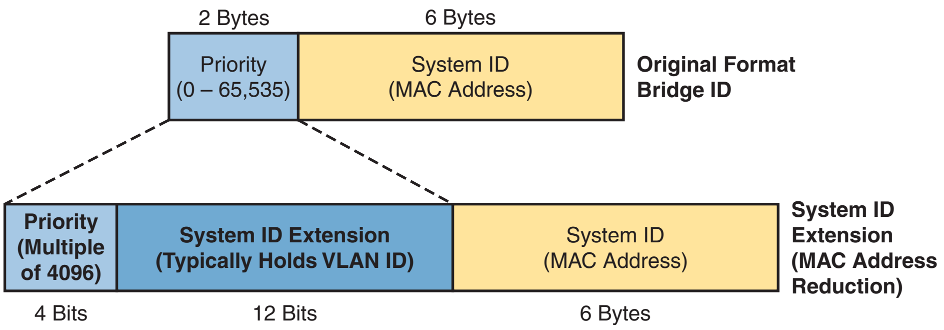
The Bridge ID and System ID Extension

To support the idea of multiple spanning trees, whether one per VLAN or simply multiple as created with MSTP, the protocols must consider the VLANs and VLAN trunking. (That's one reason why RSTP and MSTP now exist as part of the 802.1Q standard, which defines VLANs and VLAN trunking.) To help make that work, the IEEE redefined the format of the original BID value to help make per-VLAN instances of STP/RSTP become a reality.

Originally, a switch's BID was formed by combining the switch's 2-byte priority and its 6-byte MAC address. The IEEE later revised the 2-byte priority field as shown in

Figure 10-4 as a 4-bit priority field and a 12-bit subfield called the **system ID extension** (which represents the VLAN ID).

Key Topic



2 Bytes
6 Bytes
Priority (0 – 65,535)
System ID (MAC Address)
Original Format Bridge ID
Priority (Multiple of 4096)
System ID Extension (Typically Holds VLAN ID)
System ID (MAC Address)
System ID Extension (MAC Address Reduction)
4 Bits
12 Bits
6 Bytes

Figure 10-4 *STP System ID Extension*

Cisco switches enable you to configure only the priority part of the BID. The switch fills in its universal (burned-in) MAC address as the system ID. It also plugs in the VLAN ID of a VLAN in the 12-bit system ID extension field; you cannot change that behavior either.

However, configuring the priority field may be one of the strangest things to configure on a Cisco router or switch. Focusing on the top of Figure 10-4, the priority field was originally a 16-bit number, which represented a decimal number from 0 to 65,535. Because of that history, the configuration command (**spanning-tree vlan** *vlan-id* **priority** *x*) requires a decimal number between 0 and 65,535. However, because the modern use of this field reserves the final 12 bits for the VLAN ID, IOS restricts the command to multiples of 4096. Table 10-3 shows the reason: The allowed decimal values, when viewed as 16-bit binary values, have all zeros in the final 12 bits.

Table 10-3 STP/RSTP Configurable Priority Values

Decimal Value	16-bit Binary Equivalent	Decimal Value	16-bit Binary Equivalent
0	0000 0000 0000 0000	32768	1000 0000 0000 0000
4096	0001 0000 0000 0000	36864	1001 0000 0000 0000
8192	0010 0000 0000 0000	40960	1010 0000 0000 0000
12288	0011 0000 0000 0000	45056	1011 0000 0000 0000
16384	0100 0000 0000 0000	49152	1100 0000 0000 0000
20480	0101 0000 0000 0000	53248	1101 0000 0000 0000
24576	0110 0000 0000 0000	57344	1110 0000 0000 0000
28672	0111 0000 0000 0000	61440	1111 0000 0000 0000

Example 10-2 shows how to configure the priority setting for each VLAN. Note that switches default to a base priority of 32,768.

Example 10-2 *Help Shows Requirements for Using Increments of 4096 for Priority*

```
SW1(config)# spanning-tree vlan 1 priority ?
  <0-61440>  bridge priority in increments of 4096
SW1(config)#
```

Identifying Switch Priority and the Root Switch

Cisco Catalyst switches configure the priority value using a number that represents a 16-bit value; however, the system ID extension exists as the low-order 12 bits of that same number. This next topic works through connecting those ideas.

When the switch builds its BID to use for RSTP in a VLAN, it must combine the configured priority with the VLAN ID of that VLAN. Interestingly, the configured priority results in a 16-bit priority that always ends with 12 binary 0s. That fact makes the process of combining values to create the BID a little simpler for the switch and possibly a little simpler for network engineers once you understand it all.

First, consider the process shown in Figure 10-5. The top shows the configured priority value (decimal 32768), in 16-bit binary form, with a System ID Extension of 12 zeros. Moving down the figure, you see the binary version of a VLAN ID (decimal 9). At the last step, the switch replaces those last 12 bits of the System ID Extension with the value that matches the VLAN ID and uses that value as the first 16 bits of the BID.

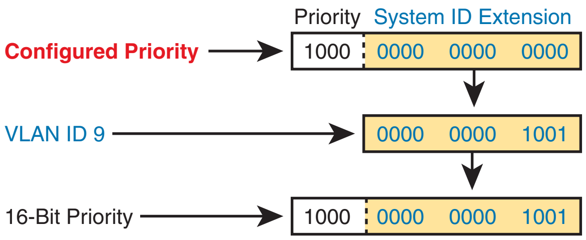


Figure 10-5 *Configured Priority (16-Bit) and System ID Extension (12-Bit) Added*

Switch Priority and Identifying the Root Switch

In most STP designs, you favor two switches to become the root switch. Often, the design calls for one switch to be the root switch, with a second switch ready to take over as root if the first switch fails. Figure 10-6 shows the idea, with switch SW1 as the primary and switch SW2 as the secondary. Additionally, note that the figure uses the interface type abbreviation of Te for Ten-GigabitEthernet. Also, the switch shows only links between switches to simplify the discussion.

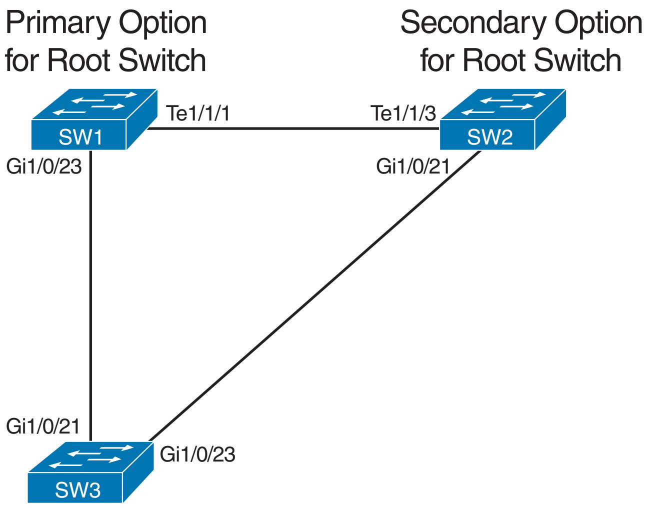

Primary Option for Root Switch
Secondary Option for Root Switch
SW1
Te1/1/1
Te1/1/3
SW2
Gi1/0/23
Gi1/0/21
Gi1/0/21
Gi1/0/23
SW3

Figure 10-6 *Sample Network for Root Switch Election Configuration*

Cisco switches default to use a default base priority of 32,768. To achieve the STP goals in the figure, the engineer needs to lower switch SW2's priority to lower than the default value and the value on switch SW1 to even lower. For example, to do just that for the STP instance for VLAN 9, the engineer could do the following:

- **On SW1:** Configure the **spanning-tree vlan 9 priority 24576** global command.
- **On SW2:** Configure the **spanning-tree vlan 9 priority 28672** global command.
- **On SW3:** Rely on the default base priority of 32,768.

The **show spanning-tree vlan 9** command, featured in Example 10-3, shows many facts about STP operation. Of particular importance, it lists about five lines of output about the root switch, starting with the line that begins with "Root ID." Following that, it lists several lines about the local switch: the switch on which this command was run. In this case, the output comes from nonroot switch SW3, so the section about the root switch refers to another switch (SW1).

Example 10-3 *Examining the 16-bit Priority from Nonroot Switch SW3*

```
SW3# show spanning-tree vlan 9
VLAN0009
  Spanning tree enabled protocol rstp
  Root ID    Priority    24585
             Address     4488.165a.f200
             Cost        4
             Port        21 (GigabitEthernet1/0/21)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32777  (priority 32768 sys-id-ext 9)
             Address     5cfc.6608.2880
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/1             Desg FWD 4         128.1    P2p
Gi1/0/2             Desg FWD 4         128.2    P2p
Gi1/0/3             Desg FWD 4         128.3    P2p
Gi1/0/21            Root FWD 4         128.21   P2p
Gi1/0/23            Altn BLK 4         128.23   P2p
```

Look closely at those early message groups that begin with Root ID (about the root switch) and bridge ID (about the local switch). Both sections identify a bridge ID in two parts: the priority and MAC address. The output for the local switch breaks the priority down into the base priority and the VLAN ID. You can also confirm that the local switch is not the root switch based on two different facts:

- The bridge ID values (priority plus MAC address) in the two message sections differ, with the one in the root ID section identifying the root switch.

- The "Root ID" section lists a line with a port number (G1/0/21). That line identifies the local switch's root port. Only nonroot switches have a root port, confirming the local switch is not the root switch.

The output also notes the priority of the root switch and the local switch. The highlighted line for the local switch shows a priority of 32,777, broken down in the same line as a base priority of the (default) value of 32,768 and the system ID extension, or VLAN, of 9 in this case. Earlier, the section about the root switch lists a priority of 24,585. Knowing the output is about VLAN 9, subtract 9 to get the base priority of 24,576.

Consider the output in Example 10-4 from root switch SW1 for comparison. Of note:

- The sections about the root switch and the local switch both list the same priority and MAC address, confirming the local switch is the root switch.
- The section about the root switch does not list a root port; root switches do not have root ports.
- The section about the root switch states, "This bridge is the root."

Example 10-4 *Examining the 16-bit Priority from Root Switch SW1*

```
SW1# show spanning-tree vlan 9
VLAN0009
  Spanning tree enabled protocol rstp
  Root ID    Priority    24585
             Address     4488.165a.f200
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    24585  (priority 24576 sys-id-ext 9)
             Address     4488.165a.f200
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/23            Desg FWD 4         128.23   P2p
Te1/1/1             Desg FWD 2         128.25   P2p
```

Finally, for one more fact of many in the detailed output from this command, note that the first highlighted line in Examples 10-3 and 10-4 both show the phrase "protocol rstp." That phrase occurs when using PVST+, per the **spanning-tree mode rapid-pvst** global command.

Switch Priority Using Root Primary and Secondary

Examples 10-3 and 10-4 relied on the direct configuration of the best priority in switch SW1 (24,576) and second best in switch SW2 (28,672). However, knowing that most STP designs identify the best and second-best switches to use as the root switch, Cisco provides two related commands that mirror that idea. To configure two switches to be the two most likely switches to be the root switch, simply configure

spanning-tree vlan *x* **root primary** (on the switch that should be primary)

spanning-tree vlan *x* **root secondary** (on the switch that should be secondary)

Both of these commands use some different IOS logic compared to most configuration commands. Both commands cause IOS to choose a priority value when the command is added to the configuration. Then, IOS does not store the above commands; instead, it stores the priority setting in the **spanning-tree vlan** *x* **priority** *value* command. The command with **root primary** or **root secondary** does not appear in the configuration.

When configuring **root primary**, the switch looks at the priority of the current root switch and chooses either (a) 24,576 or (b) 4096 less than the current root's priority (if the current root's priority is 24,576 or less). The **root secondary** option always results in that switch using a priority of 28,672; the value will be less (better) than other switches that use the default of 32,768 and higher (worse) than any switch configured as **root primary**.

RSTP (One Tree) and RPVST+ (One Tree Per VLAN)

To complete some of the conceptual discussion about the bridge ID, focus on the standard RSTP and its Cisco-proprietary cousin RPVST+. Both use the RSTP mechanisms as discussed in Chapter 9, "Spanning Tree Protocol Concepts," but RPVST+ uses the mechanisms for every VLAN, while standard RSTP does not. So how do their methods differ?

Key Topic

- RSTP creates one tree—the common spanning tree (CST)—while RPVST+ creates one tree for each and every VLAN.
- RSTP sends one set of RSTP messages (BPDUs) in the network, while RPVST+ sends one set of messages per VLAN.
- RSTP sends messages to multicast address 0180.C200.0000 (per the IEEE standard), while RPVST+ uses multicast address 0100.0CCC.CCCD (an address chosen by Cisco).
- On VLAN trunks, RSTP sends all BPDUs in the native VLAN without a VLAN header/tag. RPVST+ sends BPDUs for each VLAN, respectively. For instance, BPDUs about VLAN 9 have an 802.1Q header that lists VLAN 9.
- RSTP sets the BID's VLAN field (extended system ID) value to 0000.0000.0000, meaning "no VLAN," while RPVST+ uses the VLAN ID.

In other words, standard RSTP behaves as if VLANs do not exist, while Cisco's RPVST+ integrates VLAN information into the entire process.

NOTE Some documents refer to the feature of sending BPDUs over trunks with VLAN tags matching the same VLAN as *BPDU tunneling*.

Identifying Port Cost, Role, and State

Only the bridge ID values impact the root switch choice, and that choice has a direct impact on the STP topology. However, changes to STP interface costs change each nonroot switch's calculation of its root cost. The root cost then impacts the choice of root port and designated ports, which can impact the STP topology.

To explore those concepts, configure Figure 10-7. It repeats the topology of Figure 10-6, assuming that switch SW1 is the root switch and all port costs in VLAN 9 use default values. The figure focuses the discussion on switch SW3, which calculates possible root costs over two paths. The lower root cost, which uses SW3's port G1/0/21, becomes SW3's root port. SW3's G1/0/23 loses the designated port election due to SW3's root cost of 4 versus SW2's root cost of 2.

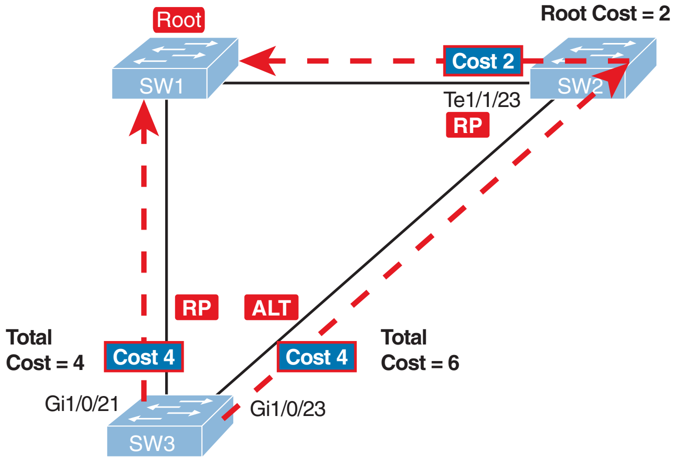


Figure 10-7 *Sample Network for Root Switch Election Configuration*

The **show spanning-tree vlan 9** command identifies those same facts. Example 10-5 shows an excerpt, with only the two interfaces that connect to other switches listed. The Role column lists the STP roles of Root and Altn (Alternate), while the abbreviated Sts heading refers to the State or Status. (Note that even though the switch uses RSTP, the output still uses the term BLK for blocking rather than the correct RSTP term of discarding.) Finally, the output lists the STP interface cost.

Example 10-5 *Switch SW3 Port Costs, Roles, and States with Default Costs*

```
SW1# show spanning-tree vlan 9 | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/21            Root FWD 4         128.21   P2p
Gi1/0/23            Altn BLK 4         128.23   P2p
```

As for the root cost, look back to Example 10-3, which also shows output from switch SW3. Find the initial section about the root switch and the line that shows "Cost 4"—that line identifies switch SW3's root cost rather than an interface's cost.

The STP topology shown in Figure 10-7 works well and you would likely not want to change it. But just to show how to do so, consider Example 10-6. It changes SW3's post cost on its root port (G1/0/21) from 4 to 10. As a result, the path out SW3's G1/0/23 port toward switch SW2 has a lower root cost. The example begins with a **debug** command, so you can see interesting facts about the actions IOS takes with this straightforward configuration.

Example 10-6 *Changing SW3 Port Cost Setting Triggers New Root and Alternate Port Choices*

```
SW3# debug spanning-tree events
Spanning Tree event debugging is on

SW3# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)# int g1/0/21
SW3(config-if)# spanning-tree vlan 9 cost 10
SW3(config-if)#
Oct 19 11:34:38.983: RSTP(9): updt roles, received superior bpdu on Gi1/0/23
Oct 19 11:34:38.983: RSTP(9): Gi1/0/23 is now root port
Oct 19 11:34:38.987: RSTP(9): Gi1/0/21 blocked by re-root
Oct 19 11:34:38.987: RSTP(9): Gi1/0/21 is now alternate
Oct 19 11:34:38.987: STP[9]: Generating TC trap for port GigabitEthernet1/0/23
SW3(config-if)# end
SW3# show spanning-tree vlan 9
VLAN0009
  Spanning tree enabled protocol rstp
  Root ID    Priority    24585
             Address     4488.165a.f200
             Cost        6
             Port        23 (GigabitEthernet1/0/23)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32777  (priority 32768 sys-id-ext 9)
             Address     5cfc.6608.2880
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/21            Altn BLK 10        128.21   P2p
Gi1/0/23            Root FWD 4         128.23   P2p
```

Comparing the output from Example 10-6 to Example 10-5, you can see that switch SW3's ports swapped roles. G1/0/21 now blocks, with G1/0/23 now in a FWD or forwarding state. Also, in the highlighted lines toward the top of the **show spanning-tree vlan 9** output, you see the updated root cost of 6 and another reference to the root port, G1/0/23.

Author's Note Alternately, you can configure the STP port cost on an interface, for all VLANs, with the **spanning-tree cost** *value* interface subcommand. Doing so sets that value as the cost on that interface for any VLANs for which the interface does not also have a VLAN-specific version of the command (as shown in Example 10-6).

Identifying Optional STP Features

Chapter 9's section titled "Optional STP Features" introduced the concepts behind four STP features: PortFast, BPDU Guard, Root Guard, and Loop Guard. This next major section, with a similar title, focuses on identifying the presence of each feature and the results of using it. Along the way, you will also learn the basics of how to configure each.

PortFast and BPDU Guard

As discussed back in Chapter 9, PortFast provides both a useful feature but also a notable risk. Once the interface reaches a connected state, PortFast logic moves a port immediately to the STP designated port (DP) role and to a forwarding state. However, that behavior can cause a forwarding loop if the port becomes connected to a switch (with other connections to the rest of the network) instead of to an endpoint device. By combining PortFast with BPDU Guard, you prevent loops by disabling the port if any BPDUs arrive in the port. So, it makes sense to examine both features together.

PortFast and BPDU Guard on an Access Port with One Endpoint

First, consider the classic case to use both features: a switch port (G1/0/1) connected to a PC. Example 10-7 shows the configuration for both features on that port. Take time to read the long warning messages IOS generates in response to the **spanning-tree portfast** interface subcommand, basically suggesting the use of BPDU Guard.

Example 10-7 *Enabling PortFast and BPDU Guard on Access Port G1/0/1*

```
SW1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)#
SW1(config)# interface g1/0/1
SW1(config-if)# switchport mode access
SW1(config-if)# switchport access vlan 9
SW1(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 GigabitEthernet1/0/1 but will only
have effect when the interface is in a non-trunking mode.
SW1(config-if)# spanning-tree bpduguard ?
  disable  Disable BPDU guard for this interface
  enable   Enable BPDU guard for this interface

SW1(config-if)# spanning-tree bpduguard enable
SW1(config-if)#
```

First, the **spanning-tree portfast** subcommand (with no additional keywords) tells IOS to enable PortFast logic only on an access port. If the port operates as a trunk, IOS does not apply PortFast logic. In fact, the highlighted portion of the long message that IOS generates when you configure the command reminds you of that fact.

As for BPDU Guard, the **spanning-tree bpduguard enable** interface subcommand applies BPDU Guard logic to the port regardless of whether operating as an access or trunk port and regardless of whether PortFast is used. Once enabled, BPDU Guard uses the following trigger, action, and recovery steps:

Key Topic

- **Trigger:** Any BPDU arrives in a port that has BPDU Guard enabled.
- **Actions:**
  - IOS places the interface into an error disabled (err-disabled) interface state.
  - STP removes the interface from the STP instance because the interface fails—that it, it is no longer up (connected).
- **Recovery:**
  - By default, the interface must be configured first with a **shutdown** command and then with a **no shutdown** command.
  - Alternately, and not discussed here, you can configure error disable recovery parameters to automatically recover the port after some time.

Example 10-8 shows a before-and-after example. It begins with switch SW1 port G1/0/1 as configured in Example 10-7, with both PortFast and BPDU Guard enabled. Example 10-8 begins with port G1/0/1 in an STP port role of Desg (designated) and a port state of FWD (forwarding), as consistent with a port in PortFast mode. It also reveals an interface state of connected. Also, in that first command's output, the highlighted port type of "P2p Edge" has great importance: the word "Edge" appears only if PortFast is both configured and enabled, so it confirms that the port uses PortFast.

Example 10-8 *Example of BPDU Guard Disabling a Port*

```
SW1# show spanning-tree interface g1/0/1
Vlan                Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
VLAN0009            Desg FWD 4         128.1    P2p Edge

SW1# show interfaces g1/0/1 status

Port         Name               Status       Vlan       Duplex  Speed Type
Gi1/0/1      Host A                          9          a-full  a-1000
10/100/1000BaseTX
SW1#
SW1# ! The cable was removed from the PC and connected to a LAN switch.
SW1#
*Jan 30 17:08:19.024: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEther-
net1/0/1, changed state to down
*Jan 30 17:08:20.024: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/1, changed state
to down
*Jan 30 17:08:30.364: %SPANTREE-2-BLOCK_BPDUGUARD: Received BPDU on port Gi1/0/1
with BPDU Guard enabled. Disabling port.
```

```
*Jan 30 17:08:30.364: %PM-4-ERR_DISABLE: bpduguard error detected on Gi1/0/1, put-
ting Gi1/0/1 in err-disable state
SW1#
SW1# show spanning-tree interface g1/0/1
no spanning tree info available for GigabitEthernet1/0/1

SW1# show interfaces g1/0/1 status

Port         Name              Status       Vlan       Duplex  Speed Type
Gi1/0/1      Host A            err-disabled 9            auto   auto
10/100/1000BaseTX
```

In the middle of the example we replaced the attached PC with a switch. The log messages reveal SW1's port G1/0/1 failing and recovering. As soon as the port came up again, BPDU Guard noticed the incoming BPDUs from the neighboring switch—and disabled the port, as seen in the highlighted log messages.

The bottom of the example repeats the same two **show** commands as the top of the example, revealing the actions taken. The interface state of err-disabled (error disabled) confirms BPDU Guard disabled the interface due to errors. The response from the **show spanning tree interface g1/0/1** command no longer lists information about this port, implying that the interface is no longer part of that spanning tree.

PortFast on VLAN Trunks and Voice Pseudo-Trunks

Cisco IOS also supports PortFast on trunk ports. You should not use PortFast on trunk ports connected to other switches, but you can use it on trunk ports connected to endpoints, as seen in the center of Figure 10-8. You can also use it on the pseudo-trunk created for voice ports connected to IP phones, as seen in port G1/0/3 in the figure. (For a review of voice VLAN configuration, refer to Chapter 8's section titled "Data and Voice VLAN Configuration and Verification.")

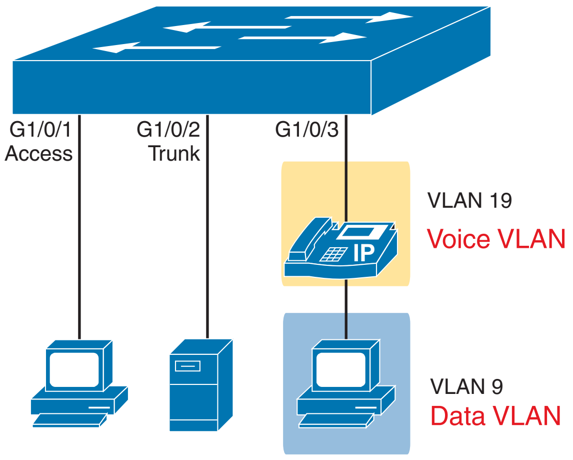


Figure 10-8 *Three Different Scenarios for Portfast and BPDU Guard*

The PortFast configuration for the voice port looks identical to the access ports. The only small difference happens with IOS automatically adding the **spanning-tree portfast** interface subcommand when you first configure the voice VLAN using the **switchport voice vlan** *vlan-id* interface subcommand.

Note that the **spanning-tree portfast** interface subcommand requires IOS to decide whether to apply PortFast logic or not, based on whether the command does or does not include the **trunk** keyword. The logic is:

spanning-tree portfast: Use PortFast if the port operates as an access port.

spanning-tree portfast trunk: Use PortFast if the port operates as a trunk.

Example 10-9 shows the configuration for switch SW1 port G1/0/2, attached via a trunk to a server. Note the different warning message in Example 10-9 versus Example 10-7's sample configuration.

Example 10-9 *Configuring Portfast and BPDU Guard on a Trunk*

```
SW1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)# interface g1/0/2
SW1(config-if)# switchport mode trunk
SW1(config-if)# spanning-tree portfast trunk
%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
SW1(config-if)# spanning-tree bpduguard enable
SW1(config-if)#
```

You must be ready to discover whether IOS decided to apply PortFast logic to a port. To do so, use the **show spanning-tree** command (without the **interface** keyword) as seen in Example 10-10. The lower part of the command output lists one line per interface for all interfaces active in that spanning tree, including both access and trunk links. The value under the Type heading in the **show spanning-tree** command output reveals whether PortFast is being used:

- **P2p Edge:** Port operates with PortFast logic
- **P2p (without "Edge"):** Port does not use PortFast logic

Example 10-10 *Confirming Portfast with Port Type Edge*

```
SW1# show spanning-tree vlan 9 | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/1             Desg FWD 4         128.1    P2p Edge
Gi1/0/2             Desg FWD 4         128.2    P2p Edge
Gi1/0/3             Desg FWD 4         128.3    P2p Edge
Gi1/0/23            Desg FWD 4         128.23   P2p
Te1/1/1             Desg FWD 2         128.25   P2p
```

Global Configuration of PortFast and BPDU Guard

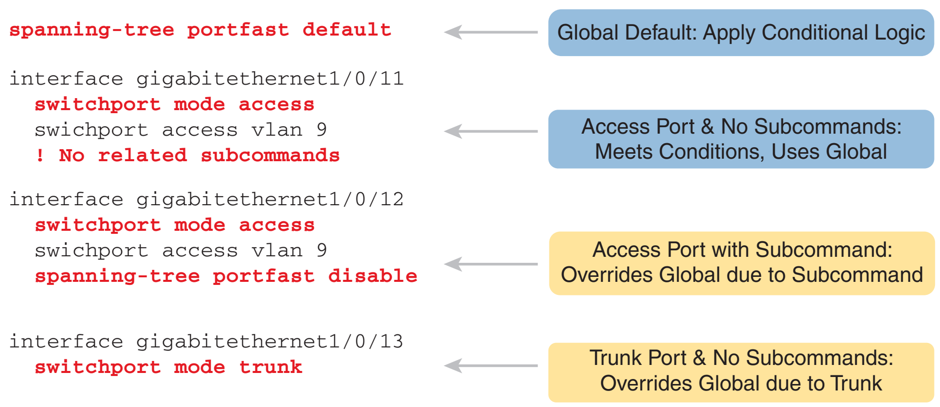
The configuration examples so far use interface subcommands that override any global settings. IOS defaults to global settings whose values disable both features per port; however, you can enable each feature globally. For interfaces with no related subcommands, IOS enables the feature on the interface. Then, you should identify the interfaces for which the feature should be disabled and use another interface subcommand to disable the feature per interface, as needed.

For example, consider an access layer switch with 48 access ports connected to endpoints, plus two trunk ports connected to other switches. You probably want to enable both PortFast and BPDU Guard on all 48 access ports. Rather than requiring the interface subcommands on all 48 of those ports, enable both the features globally, and then disable them on the uplink ports.

First, consider PortFast. Configuring the **spanning-tree portfast default** global command tells IOS to apply PortFast to some ports, based on the following conditions:

- Enable PortFast on ports operating as access ports only.
- Ignore ports configured to disable PortFast with the **spanning-tree portfast disable** interface subcommand.

To see that logic in action, work through the examples in Figure 10-9. The switch uses the global command **spanning-tree portfast default**. Port G1/0/11 has no **spanning-tree portfast** subcommands and is configured to be an access port, so IOS applies PortFast to that port. However, port G1/0/12 has a **spanning-tree portfast disable** subcommand, so IOS does not apply PortFast. Port G1/0/13 fails to meet the required conditions because it uses trunking.



```
spanning-tree portfast default

interface gigabitethernet1/0/11
  switchport mode access
  swichport access vlan 9
  ! No related subcommands

interface gigabitethernet1/0/12
  switchport mode access
  swichport access vlan 9
  spanning-tree portfast disable

interface gigabitethernet1/0/13
  switchport mode trunk
```

Global Default: Apply Conditional Logic
Access Port & No Subcommands: Meets Conditions, Uses Global
Access Port with Subcommand: Overrides Global due to Subcommand
Trunk Port & No Subcommands: Overrides Global due to Trunk

Figure 10-9 *Conditional PortFast Global Configuration Logic*

IOS also supports a similar configuration process for BPDU Guard, but with this configuration, BPDU Guard is tied to PortFast. You configure the **spanning-tree portfast bpduguard default** global command. Doing so asks IOS to enable BPDU Guard under these conditions:

- Enable BPDU Guard on ports that currently use PortFast.
- Ignore ports configured to disable BPDU Guard with the **spanning-tree bpduguard disable** interface subcommand.

For the exam, interpreting **show** command output might be more important than the intricacies of the configuration. For example, exam topic 2.5, the only one that mentions STP features, uses the verb *interpret* rather than *configure* or *verify*. So, be ready to interpret **show** command output and even predict the related configuration.

As an example of determining the configuration from the **show** commands, consider interfaces G1/0/7 and G1/0/8 on switch SW1. The configuration (not shown) uses this convention:

- **G1/0/7:** Uses interface subcommands **spanning-tree portfast** and **spanning-tree bpduguard enable**, not relying on the global configuration commands.
- **G1/0/8:** Uses no relevant interface subcommands, instead relying on the global configuration commands **spanning-tree portfast default** and **spanning-tree portfast bpduguard default**.

Example 10-11 shows truncated output from the **show spanning-tree interface** command for both interfaces. Compare the highlighted lines about PortFast and BPDU Guard to see the different information. On port G1/0/8, the phrase "by default" confirms that G1/0/8 uses the default setting per the global command. The absence of "by default" in the output for port G1/0/7 means those settings come from interface subcommands.

Example 10-11 *Interpreting the Source of PortFast and BPDU Guard Configuration*

```
SW1# show spanning-tree interface g1/0/7 detail | begin portfast
   The port is in the portfast mode
   Link type is point-to-point by default
   Bpdu guard is enabled
   BPDU: sent 387, received 0

SW1# show spanning-tree interface g1/0/8 detail | begin portfast
   The port is in the portfast mode by default
   Link type is point-to-point by default
   Bpdu guard is enabled by default
   BPDU: sent 774, received 0
```

BPDU Filter

In Chapter 9's section about this same topic, also titled "BPDU Filter," you learned about the two different logic branches of this feature. To review:

1. Using a global configuration command, enable BPDU Filter, after which IOS applies BPDU Filter on PortFast ports only. While applied, it monitors for incoming BPDUs. When incoming BPDUs occur, BPDU Filter disables PortFast logic, so that the port then uses normal STP logic on the port.
2. Using an interface subcommand, enable BPDU filter on the port. BPDU Filter discards all outgoing and incoming BPDUs on the port, effectively disabling STP on the port.

This section examines both logic branches in order.

Conditional BPDU Filtering with Global Configuration

To enable the conditional logic of BPDU Filter, you must toggle from the default global setting of **no spanning-tree portfast bpdufilter enable** to the same command without the **no** option: the **spanning-tree portfast bpdufilter enable** global command.

Similar to the effect of the **spanning-tree portfast bpduguard enable** global command, the **spanning-tree portfast bpdufilter enable** global command asks IOS to enable BPDU Filter under these conditions:

- Enable BPDU Filter on ports that currently use PortFast.
- Ignore ports configured to disable BPDU Filter with the **spanning-tree bpdufilter disable** interface subcommand.

Example 10-12 shows a straightforward scenario on switch SW1 port G1/0/23. It shows a classic access port, in VLAN 9, with PortFast enabled with an interface subcommand—along with conditional BPDU Filter enabled globally.

Example 10-12 *BPDU Filter as Global Default, Applied to Port G1/0/23*

```
spanning-tree portfast bpdufilter enable
!
interface GigabitEthernet1/0/23
  switchport mode access
  switchport access vlan 9
  spanning-tree portfast
  ! No BPDU Filter subcommands present
```

You should be ready to think about the configuration and understand the rules IOS applies—and to also see the evidence of the choices in IOS command output. First, to review how IOS interprets and applies the configuration:

- The combination of port G1/0/23 as an access port (from the **switchport mode access** subcommand), with the command to enable PortFast on access ports (the **spanning-tree portfast** subcommand without the **trunk** keyword), enables PortFast.
- The one global command tells IOS to find current PortFast ports (no matter whether access or trunk port) and enable BPDU Filter conditional logic.

Example 10-13 shows evidence of these listed results, while an endpoint connects to the port rather than a rogue switch. Look for the following:

1. The first command, **show spanning-tree**, lists interfaces in the tree for that VLAN. It lists the port type for G1/0/23 as P2p Edge—the word Edge confirms that the port currently uses Portfast.
2. The final command, in the final line, lists a counter of 11 sent BPDUs and 0 received. That confirms the switch sent 11 Hellos before BPDU Filter stopped sending them after 20 seconds. If nothing changes, the received BPDU counter remains at 0 because the attached endpoint device does not send BPDUs to the switch.

3. The phrase at the end of the example "Bpdu filter is enabled by default," at the end of the **show spanning-tree interface** command, reveals that the BPDU Filter configuration uses a global command. This command's output includes the phrase "by default" when the global configuration setting is the reason the feature is enabled.
4. Conversely, that same **show spanning-tree interface** command output reveals that the PortFast configuration uses an interface subcommand. The phrase "by default" does not occur at the end of the line about PortFast, implying the configuration comes from an interface subcommand.

Example 10-13 *Key Status Values When an Endpoint Connects to Port G1/0/23*

```
SW1# show spanning-tree vlan 9 | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/1             Desg FWD 4         128.1    P2p Edge
Gi1/0/2             Desg FWD 4         128.2    P2p Edge
Gi1/0/3             Desg FWD 4         128.3    P2p Edge
Gi1/0/23            Desg FWD 4         128.23   P2p Edge
Te1/1/1             Desg FWD 2         128.25   P2p

SW1# show spanning-tree interface g1/0/23

Vlan                Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
VLAN0009            Desg FWD 4         128.23   P2p Edge

SW1# show spanning-tree interface g1/0/23 detail | begin portfast
   The port is in the portfast mode
   Link type is point-to-point by default
   Bpdu filter is enabled by default
   BPDU: sent 11, received 0
```

As configured using the global command, IOS applies conditional BPDU Filter logic. To see that in action, Example 10-14 begins with the replacement of the attached PC with a rogue switch. The example does not show the related log messages, but know that the interface fails and recovers. The example repeats the same **show** commands as in the previous example, but with these differences:

1. The first command, **show spanning-tree**, lists the port type for G1/0/23 as P2p but without the word Edge. That output change is what confirms IOS no longer applies PortFast logic to this port.
2. The final command, in the final line, lists BPDU counters. In this case, the neighboring switch becomes root and continues to send BPDUs into port G1/0/23. The received BPDU counter will continue to increment over time.
3. The **show spanning-tree interface** command no longer includes the line that mentions the BPDU Filter feature.

Example 10-14 *SW1 Port G1/0/23 Connects to a Rogue Switch*

```
! Someone disconnects the endpoint off SW1's G1/0/23 and attaches a switch:
! G1/0/23 fails and recovers…

SW1# show spanning-tree vlan 9 | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/23            Root FWD 4         128.23   P2p

SW1# show spanning-tree interface g1/0/23 detail | begin portfast
   The port is in the portfast mode
   Link type is point-to-point by default
   BPDU: sent 6, received 138
! Line "bpdu filter is enabled by default" does not appear above.
```

Disabling STP with BPDU Filter Interface Configuration

The other type of BPDU Filter logic always filters all outgoing and incoming BPDUs on a port, in effect disabling STP on the interface. To configure it, simply configure the **spanning-tree bpdufilter enable** interface subcommand.

NOTE That previous statement ought to scare you—one simple command can disable STP on one link. With redundant links in the LAN, that one command, on one port, can create a forwarding loop that makes the entire LAN unusable. That is not an exaggeration. Be very careful before using this command!

Example 10-15 shows a typical configuration. This feature makes sense on links between switches, so it would likely be a trunk port.

Example 10-15 *Disabling STP on a Port Between STP Domains*

```
interface TenGigabitEthernet 1/1/1
  switchport mode trunk
  spanning-tree bpdufilter enable
```

Example 10-16 shows just a few lines of output that are the keys to confirming the feature works. The end of the **show spanning-tree interface** command has a line that confirms the BPDU Filter feature is enabled; however, that line does not end with "by default" as seen near the end of Example 10-14. That tiny difference in the text signals a huge difference in logic! The absence of "by default" means IOS enables BPDU Filter in this case due to an interface subcommand, which means IOS applies the absolute filtering logic of BPDU Filter—disabling STP on the interface.

Example 10-16 *Typical Evidence of BPDU Filter to Disable STP*

```
SW1# show spanning-tree vlan 9 interface te1/1/1 detail | begin Bpdu
   Bpdu filter is enabled
   BPDU: sent 0, received 0
```

To support the claim that IOS applies absolute filtering of all outgoing and incoming BPDUs on the port, note that the counters in the final output line show 0s. Over time, it will continue to show 0s, because BPDU Filter discards all BPDUs on the port.

Root Guard

The most challenging part of working with Root Guard involves analyzing the STP design to decide which ports might benefit from using it. Chapter 9's section with this same title, "Root Guard," discusses the logic, but to summarize, you consider only ports connected to other switches. Then you look for special cases of switch ports that should never receive a superior Hello from a new root switch based on the intended STP design. Root Guard then monitors for incoming superior Hellos, disabling the port when that occurs. Root Guard may not apply to any ports for some networks, but the choice of ports begins by thinking about the STP design, as discussed in Chapter 9.

If ports need to use Root Guard, the implementation and verification take a few short steps. The configuration uses a single option: the **spanning-tree guard root** interface subcommand. There is no global command to change the default, with the default being not to use Root Guard on an interface.

As for the actions taken by Root Guard, Chapter 9 described the big concepts, but the following list provides more detail that you will see in the upcoming CLI examples.

Key Topic

- **Trigger:** Root Guard acts after receiving a superior BPDU in the port.
- **Actions:** Once triggered, Root Guard takes these actions:
  - The actions occur per VLAN, based on the VLAN of the superior Hello.
  - STP places the port in a broken (BRK) state for that VLAN, which discards all traffic (like the discarding and blocking states).
  - STP describes the port as being in a root inconsistent state in the port type information in show commands.
- **Recovery:** When the incoming superior BPDUs cease for a time, STP reverts to its prior STP state without any operator intervention.

Examples 10-17 and 10-18 combine to show an example of Root Guard in action. First, Example 10-17 shows the simple configuration, with the log message that confirms IOS enabled the feature.

Example 10-17 *Configuring Root Guard on Switch SW3 Port G1/0/11*

```
SW3# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)# interface g1/0/11
SW3(config-if)# spanning-tree guard root
SW3(config-if)#
Oct 21 11:02:31.145: %SPANTREE-2-ROOTGUARD_CONFIG_CHANGE: Root guard enabled on port
GigabitEthernet1/0/11.
SW3#
```

To see Root Guard in action, look for log messages, the BRK (broken) port state, and the root inconsistent state. Example 10-18 does just that, as follows:

1. The first two commands confirm port G1/0/11's interface state (connected), STP role (Desg, or designated), STP forwarding state (FWD), and port type (P2P).
2. A log message reveals that Root Guard blocked the port. (Just before that, but not shown, I lowered the priority of the neighboring switch.)
3. The last two commands at the bottom of the example reveal:
   a. No change in in port G1/0/11's interface state (connected) or STP role (designated).
   b. A change to the STP forwarding state BKN, meaning broken, and port type P2p ROOT Inc, meaning Root Inconsistent.

Example 10-18 *Identifying Root Guard Root Inconsistent State*

```
SW3# show interfaces g1/0/11 status

Port      Name               Status       Vlan       Duplex  Speed Type
Gi1/0/11                     connected    trunk      a-full a-1000 10/100/1000BaseTX

SW3# show spanning-tree vlan 9 int g1/0/11

Vlan                Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
VLAN0009            Desg FWD 4         128.11   P2p

SW3#! Neighboring switch priority was lowered so it sends a superior BPDU.

Oct 21 11:03:14.472: %SPANTREE-2-ROOTGUARD_BLOCK: Root guard blocking port
GigabitEthernet1/0/11 on VLAN0009.

SW3# show interfaces g1/0/11 status
Port      Name               Status       Vlan       Duplex  Speed Type
Gi1/0/11                     connected    trunk      a-full a-1000 10/100/1000BaseTX

SW3# show spanning-tree vlan 9 int g1/0/11

Vlan                Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
VLAN0009            Desg BKN*4         128.11   P2p *ROOT_Inc
```

Loop Guard

Chapter 9's section with this same title, "Loop Guard," discusses the compound factors that lead to the specific scenario where you can apply Loop Guard. Understanding those conditions and choosing ports that can effectively use Loop Guard takes some effort. However, the configuration takes only a little effort once chosen, as seen in the following list. You can either enable it directly on the interface or change the global default to enable it on all switch interfaces—but then disable it on selected interfaces as needed. Use these steps.

Key Topic

Step 1. Use the **spanning-tree guard loop** interface subcommand to enable Loop Guard on the interfaces selected to use the feature.

Step 2. To use the global default:

a. Use the **spanning-tree loopguard default** global command to change the Loop Guard default from disabled to enabled on all point-to-point switch interfaces.

b. Use the **no spanning-tree guard loop** interface subcommand to disable Loop Guard on the interfaces selected not to use the feature.

Example 10-19 shows the simple configuration on a single port. The **show spanning-tree interface** command then confirms that Loop Guard is enabled, with the absence of the “by default” phrase implying it was configured using an interface subcommand.

Example 10-19 *Configuring Loop Guard on a Port*

```
SW3# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)# interface g1/0/21
SW3(config-if)# spanning-tree guard loop
SW3(config-if)#

SW3# show spanning-tree vlan 9 int g1/0/21 detail
 Port 21 (GigabitEthernet1/0/21) of VLAN0009 is root forwarding
   Port path cost 4, Port priority 128, Port Identifier 128.21.
   Designated root has priority 24585, address 4488.165a.f200
   Designated bridge has priority 24585, address 4488.165a.f200
   Designated port id is 128.23, designated path cost 0
   Timers: message age 16, forward delay 0, hold 0
   Number of transitions to forwarding state: 7
   Link type is point-to-point by default
   Loop guard is enabled on the port
   BPDU: sent 139, received 165425

! Link becomes unidirectional; switch SW3 G1/0/21 ceases to receive BPDUs
*Feb 23 17:11:19.087: %SPANTREE-2-LOOPGUARD_BLOCK: Loop guard blocking port
GigabitEthernet1/0/23 on VLAN0009.
SW3# show spanning-tree vlan 9 int g1/0/21 detail | include 1/0/21
 Port 21 (GigabitEthernet1/0/21) of VLAN0009 is broken  (Loop Inconsistent)
```

The example ends by showing what happens when a unidirectional link occurs on this port. The end of the example shows a comment to show when the failure occurs and the log message noting the Loop Guard acted for VLAN 9. The final output shows how Loop Guard moved the port to a broken state (Loop Inconsistent) in that VLAN, similar to the Root Inconsistent state used by Root Guard. Loop Guard also recovers automatically when Hellos begin to arrive in the port again.

Configuring Layer 2 EtherChannel

As introduced in Chapter 9, "Spanning Tree Protocol Concepts," two neighboring switches can treat multiple parallel links between each other as a single logical link called an *EtherChannel*. Without EtherChannel, a switch treats each physical port as an independent port, applying MAC learning, forwarding, and STP logic per physical port. With EtherChannel, the switch applies all those same processes to a group of physical ports as one entity: the EtherChannel. Without EtherChannel, with parallel links between two switches, STP/RSTP would block all links except one, but with EtherChannel, the switch can use all the links, load balancing the traffic over the links.

NOTE All references to EtherChannel in this chapter refer to Layer 2 EtherChannels, not to Layer 3 EtherChannels (as discussed in Chapter 18, "IP Routing in the LAN"). CCNA 200-301 exam topics include both Layer 2 and Layer 3 EtherChannels.

EtherChannel might be one of the most challenging switch features to make work. First, the configuration has several options, so you have to remember the details of which options work together. Second, the switches also require a variety of other interface settings to match among all the links in the channel, so you have to know those settings as well.

This section shows how to configure a Layer 2 EtherChannel, first through manual (static) configuration and then by allowing dynamic protocols to create the channel. Cisco recommends using the dynamic method, but the static method can be a little easier to learn initially, so we begin with the static or manual option. This section closes with some information about some common configuration issues that occur with Layer 2 EtherChannels.

Configuring a Manual Layer 2 EtherChannel

To configure a Layer 2 EtherChannel so that all the ports always attempt to be part of the channel, simply add the correct **channel-group** configuration command to each physical interface, on each switch, all with the **on** keyword, and all with the same number. The **on** keyword tells the switches to place a physical interface into an EtherChannel, and the number identifies the PortChannel interface number that the interface should be a part of.

Before getting into the configuration and verification, however, you need to start using three terms as synonyms: **EtherChannel**, **PortChannel**, and **Channel-group**. Oddly, IOS uses the **channel-group** configuration command, but then to display its status, IOS uses the **show etherchannel** command. Then the output of this **show** command refers to neither an "EtherChannel" nor a "Channel-group," instead using "PortChannel." So, pay close attention to these three terms in the example.

To configure an EtherChannel manually, follow these steps:

Key Topic

Config Checklist

Step 1. Add the **channel-group** *number* **mode on** command in interface configuration mode under each physical interface that should be in the channel to add it to the channel.

Step 2. Use the same number for all commands on the same switch, but the channel-group number on the neighboring switch can differ.

Example 10-20 shows a simple example, with two links between switches SW1 and SW2, as shown in Figure 10-10. The example begins with all default interface configuration on the interfaces used for the EtherChannel. The configuration first shows the ports being configured as VLAN trunks and then manually configured to be in channel-group 1 (SW1) and channel-group 2 (SW2).

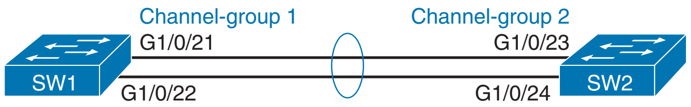


Figure 10-10 *Sample LAN Used in EtherChannel Example*

Example 10-20 *Configuring EtherChannel—Both SW1 and SW2*

```
! First, on switch SW1
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# interface range g1/0/21-22
SW1(config-if-range)# switchport mode trunk
SW1(config-if-range)# channel-group 1 mode on
```

```
! Next, on switch SW2
SW2# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW2(config)# interface range g1/0/23-24
SW2(config-if-range)# switchport mode trunk
SW2(config-if-range)# channel-group 2 mode on
```

Interestingly, IOS reacts to the **channel-group** interface subcommands to create a matching port-channel interface on the switch. Example 10-21 shows an excerpt from the **show running-config** command on switch SW1, listing the new port-channel 1 interface in the configuration, along with the two physical interfaces.

Example 10-21 *Configuration Results from Example 10-20*

```
SW1# show running-config
! Lines omitted for brevity
!
interface Port-channel1
 switchport mode trunk
!
! Lines omitted for brevity
interface GigabitEthernet1/0/21
 switchport mode trunk
 channel-group 1 mode on
!
```

```
interface GigabitEthernet1/0/22
 switchport mode trunk
 channel-group 1 mode on

SW1# show interfaces portchannel 1
Port-channel1 is up, line protocol is up (connected)
  Hardware is EtherChannel, address is 4488.165a.f215 (bia 4488.165a.f215)
  MTU 1500 bytes, BW 2000000 Kbit/sec, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 1000Mb/s, link type is auto, media type is N/A
  input flow-control is on, output flow-control is unsupported
  Members in this channel: Gi1/0/21 Gi1/0/22
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 02:12:51, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
! Interface statistics output removed for brevity
```

The end of Example 10-21 gives more insight into the portchannel interface in the output of the **show interfaces portchannel 1** command. Like all output from the **show interfaces** command, the output lists both an interface and protocol state ("up" and "up" in this case), and the interface bandwidth, noted with the text "BW." However, the output shows the bandwidth as 2,000,000 Kbps, or 2 Gbps, because the portchannel has two active 1-Gbps links. Also, in the final highlighted line, the output lists the currently active interfaces in the portchannel.

The **show etherchannel** command, shown at the top of Example 10-22, lists basic configuration information about the channel per earlier Example 10-20. In this case, it identifies the configured portchannel number (1), with two ports configured to be in the channel. It lists the protocol as a dash (–), meaning that it does not use LACP or PAgP, implying the use of the **channel-group mode on** command. Note that this command does not list status information, only configuration information.

Example 10-22 *Exploring SW1 PortChannel Configuration and Status*

```
SW1# show etherchannel
                 Channel-group listing:
                 ----------------------

Group: 1
----------
Group state = L2
Ports: 2   Maxports = 8
Port-channels: 1 Max Port-channels = 1
Protocol:    -
Minimum Links: 0
```

```
SW1# show etherchannel summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG

Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)          -        Gi1/0/21(P)    Gi1/0/22(P)
```

The **show etherchannel summary** command at the end of Example 10-22 provides status information. The output begins with an extensive status code legend. The lines at the bottom list each PortChannel along with the ports and their status. In the lowest line of output with highlights, note the code P, which means that both of those ports are bundled (working) in the channel. A status of (SU), per the legend, means the channel is in use and acts as a Layer 2 EtherChannel (rather than a Layer 3 EtherChannel).

Configuring Dynamic EtherChannels

Cisco switches also support two different configuration options that use a dynamic protocol to negotiate whether a particular link becomes part of an EtherChannel or not. Basically, the configuration enables a protocol for a particular channel-group number. At that point, the switch can use the protocol to send messages to/from the neighboring switch and discover whether their configuration settings pass all checks. If a given physical link passes, the link is added to the EtherChannel and used; if not, it is placed in a down state, and not used, until the configuration inconsistency can be resolved.

Most Cisco Catalyst switches support the Cisco-proprietary Port Aggregation Protocol (**PAgP**) and the IEEE standard Link Aggregation Control Protocol (**LACP**). Although differences exist between the two, to the depth discussed here, they both accomplish the same task: negotiate so that only links that pass the configuration checks are actually used in an EtherChannel. (Note that the IEEE originally defined LACP in amendment 802.3ad but now defines it in IEEE standard 802.1AX.)

One difference of note is that LACP does support more links in a channel—16—as compared to PaGP's maximum of 8. With LACP, only 8 can be active at one time, with the others waiting to be used should any of the other links fail.

To configure either protocol, a switch uses the **channel-group** configuration commands on each switch, but with a keyword that either means "use this protocol and begin negotiations" or "use this protocol and wait for the other switch to begin negotiations." As shown in Figure 10-11, the **desirable** and **auto** keywords enable PAgP, and the **active** and **passive** keywords enable LACP. With these options, at least one side has to begin the negotiations. In other words, with PAgP, at least one of the two sides must use **desirable**, and with LACP, at least one of the two sides must use **active**.

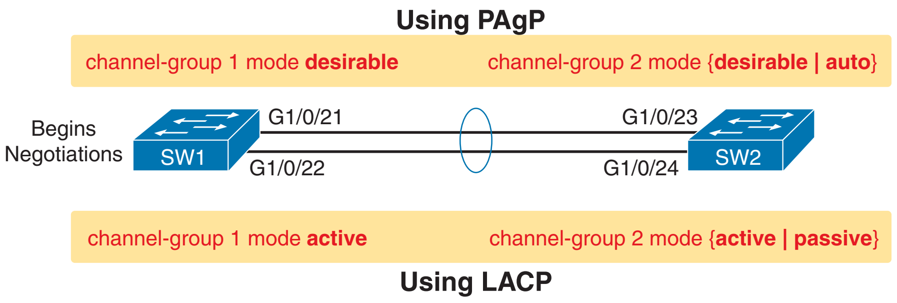

Using PAgP
channel-group 1 mode desirable
channel-group 2 mode {desirable | auto}
Begins Negotiations
SW1
G1/0/21
G1/0/22
G1/0/23
G1/0/24
SW2
channel-group 1 mode active
channel-group 2 mode {active | passive}
Using LACP

Figure 10-11 *Correct EtherChannel Configuration Combinations*

NOTE Do not use the **on** parameter on one end, and either **auto** or **desirable** (or for LACP, **active** or **passive**) on the neighboring switch. The **on** option uses neither PAgP nor LACP, so a configuration that uses **on**, with PAgP or LACP options on the other end, would prevent the EtherChannel from working.

As an example, consider the topology in Figure 10-11, which uses the same switches and ports as Figure 10-10 and the last several examples. However, the next example starts fresh, with the manual EtherChannel no longer configured and no interface commands on any of the interfaces in use. Example 10-23 shows a dynamic configuration with LACP on both switches, with the **channel-group 1 mode active** interface subcommand on SW1 and **channel-group 2 mode passive** on SW2.

Example 10-23 *Configuring an LACP Dynamic EtherChannel—Both SW1 and SW2*

```
! First, on switch SW1
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# interface range g1/0/21-22
SW1(config-if-range)# switchport mode trunk
SW1(config-if-range)# channel-group 1 mode active
```

```
! Next, on switch SW2
SW2# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW2(config)# interface range g1/0/23-24
SW2(config-if-range)# switchport mode trunk
SW2(config-if-range)# channel-group 2 mode passive
```

As with the manual configuration and verification in Examples 10-20 and 10-21, the switch creates a portchannel interface in reaction to the configuration shown in Example 10-23. Example 10-24 confirms the related settings, with group or portchannel 1, two ports in the channel, specifically SW1's G1/0/21 and G1/0/22. Also, note that the output lists the protocol as LACP, because the configuration commands in Example 10-23 use keywords **active** and **passive**, both of which enable LACP.

Example 10-24 *EtherChannel Verification: SW1 with LACP Active Mode*

```
SW1# show etherchannel port-channel
                 Channel-group listing:
                 ----------------------

Group: 1
----------
                 Port-channels in the group:
                 ---------------------------

Port-channel: Po1    (Primary Aggregator)

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

Age of the Port-channel   = 0d:00h:11m:35s
Logical slot/port   = 31/1          Number of ports = 2
HotStandBy port = null
Port state          = Port-channel Ag-Inuse
Protocol            =   LACP
Port security       = Disabled
Fast-switchover     = disabled
Fast-switchover Dampening = disabled

Ports in the Port-channel:

Index   Load   Port        EC state        No of bits
------+------+------+------------------+-----------
  0     00     Gi1/0/21       Active             0
  0     00     Gi1/0/22       Active             0

Time since last port bundled:    0d:00h:02m:17s     Gi1/0/22
Time since last port Un-bundled: 0d:00h:02m:25s     Gi1/0/22
```

Before leaving the core EtherChannel configuration and verification topics, think about EtherChannels and Spanning Tree together. STP/RSTP prefers the better links based on STP/RSTP link costs. An EtherChannel may have more than one working link, and the number of active links changes as links fail and recover. So, by default, IOS calculates the default STP cost for EtherChannel based on the number of active links.

For example, the STP/RSTP default costs prefer 10 Gbps over EtherChannels with 1 Gbps links, and EtherChannels of multiple 1 Gbps links over a single 1 Gbps link, with the following default port costs:

- **Default Cost 4:** EtherChannels with one active 1 Gbps link and any single 1 Gbps link (without an EtherChannel)
- **Default Cost 3:** EtherChannels with 2 to 8 active 1 Gbps links
- **Default Cost 2:** A single 10 Gbps link

Example 10-25 shows the default STP cost on a couple of 1 Gbps interfaces along with interface PortChannel 1, whose configuration resides earlier in Example 10-23.

Example 10-25 *EtherChannel Verification: SW1 with LACP Active Mode*

```
SW1# show spanning-tree vlan 1 | begin Interface
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi1/0/23            Desg FWD 4         128.23   P2p
Gi1/0/24            Desg FWD 4         128.24   P2p
Po1                 Desg FWD 3         128.456  P2p
```

Interface Configuration Consistency with EtherChannels

Even when the **channel-group** commands have all been configured correctly, other configuration settings can prevent a switch from using a physical port in an EtherChannel—even physical ports manually configured to be part of the channel. The next topic examines those reasons.

First, before using a physical port in a dynamic EtherChannel, the switch compares the new physical port's configuration to the existing ports in the channel. That new physical interface's settings must be the same as the existing ports' settings; otherwise, the switch does not add the new link to the list of approved and working interfaces in the channel. That is, the physical interface remains configured as part of the PortChannel, but it is not used as part of the channel, often being placed into some nonworking state.

The list of items the switch checks includes the following:

Key Topic

- Speed
- Duplex
- Operational access or trunking state (all must be access, or all must be trunks)
- If an access port, the access VLAN
- If a trunk port, the allowed VLAN list (per the **switchport trunk allowed** command)
- If a trunk port, the native VLAN

Example 10-26 shows a failure of one link in the EtherChannel due to a purposeful misconfiguration of the native VLAN on SW1's G1/0/21 port. Example 10-26 begins with the configuration of VLAN 21 as the native VLAN on port G1/0/21; port G1/0/22 defaults to native VLAN 1. The log messages following that configuration show the interface failing, with the **show interfaces g1/0/21** command listing a "down (suspended)" protocol state for that interface. The usual **show etherchannel port-channel** command lists only one port as bundled in the channel.

Example 10-26 *Native VLAN Mismatch Removes SW1 G1/0/21 from EtherChannel*

```
SW1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)# interface g1/0/21
SW1(config-if)# switchport trunk native vlan 21
SW1(config-if)# ^Z
SW1#
Jun 17 16:11:34.217: %EC-5-CANNOT_BUNDLE2: Gi1/0/21 is not compatible with Gi1/0/22
and will be suspended (native vlan of Gi1/0/21 is 21, Gi1/0/22 id 1)
Jun 17 16:11:35.220: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/21, changed state to down

SW1# show interfaces gigabitethernet 1/0/21
GigabitEthernet1/0/21 is up, line protocol is down (suspended)
  Hardware is Gigabit Ethernet, address is 4488.165a.f215 (bia 4488.165a.f215)
! lines omitted for brevity

SW1# show etherchannel port-channel
                 Channel-group listing:
                 ----------------------


Group: 1
----------
                 Port-channels in the group:
                 ---------------------------


Port-channel: Po1    (Primary Aggregator)

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

Age of the Port-channel   = 0d:05h:26m:48s
Logical slot/port   = 31/1          Number of ports = 1
HotStandBy port = null
Port state          = Port-channel Ag-Inuse
Protocol            =   LACP
Port security       = Disabled
Fast-switchover     = disabled
Fast-switchover Dampening = disabled

Ports in the Port-channel:
```

```
Index   Load   Port          EC state          No of bits
------+------+------+------------------+-----------
  0      00    Gi1/0/22       Active             0

Time since last port bundled:    0d:00h:05m:44s     Gi1/0/22
Time since last port Un-bundled: 0d:00h:01m:09s     Gi1/0/21
```

The output in Example 10-26 closes with two useful messages at the end of the **show etherchannel port-channel** command output: the interface ID and the timing of the most recent ports bundled and unbundled with the EtherChannel. In this case, it shows the most recent addition of the still-working G1/0/22 interface and the just-suspended G1/0/21 interface.

EtherChannel Load Distribution

When using Layer 2 EtherChannels, a switch's MAC learning process associates MAC addresses with the PortChannel interfaces and not the underlying physical ports. Later, when a switch makes a forwarding decision to send a frame out a PortChannel interface, the switch must do more work: to decide which specific physical port to use to forward the frame. IOS documentation refers to those rules as **EtherChannel load distribution** or *load balancing*. Figure 10-12 shows the main idea.

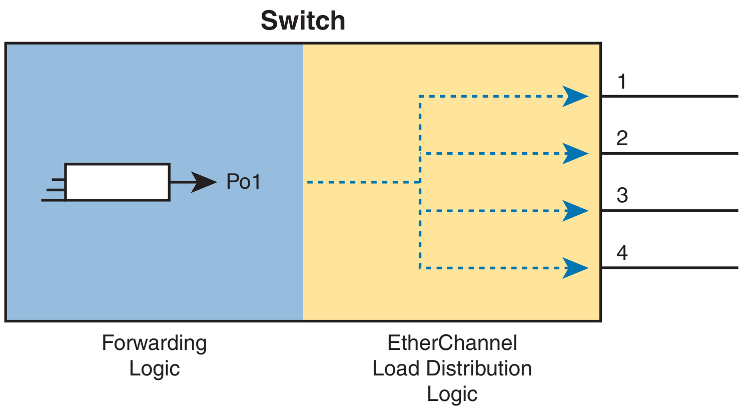


Figure 10-12 *Forwarding Concepts from Outgoing PortChannel to Physical Interfaces*

EtherChannel load distribution makes the choice for each frame based on various numeric values found in the Layer 2, 3, and 4 headers. The process uses one configurable setting as input: the load distribution method as defined with the **port-channel load-balance** *method* global command. The process then performs some match against the fields identified by the configured method.

Table 10-4 lists the most common methods. However, note that some switches may support only MAC-based methods, or only MAC- and IP-based methods, depending on the model and software version.

Table 10-4 EtherChannel Load Distribution Methods

Configuration Keyword	Math Uses...	Layer
src-mac	Source MAC address	2
dst-mac	Destination MAC address	2
src-dst-mac	Both source and destination MAC	2
src-ip	Source IP address	3
dst-ip	Destination IP address	3
src-dst-ip	Both source and destination IP	3
src-port	Source TCP or UDP port	4
dst-port	Destination TCP or UDP port	4
src-dst-port	Both source and destination TCP or UDP port	4

To appreciate why you might want to use different methods, you need to consider the results of how switches make their choice. (The discussion here focuses on the result, and not the logic, because the logic remains internal to the switch, and Cisco does not document how each switch model or IOS version works internally.) However, the various load distribution algorithms do share some common goals:

- To cause all messages in a single application flow to use the same link in the channel, rather than being sent over different links. Doing so means that the switch will not inadvertently reorder the messages sent in that application flow by sending one message over a busy link that has a queue of waiting messages, while immediately sending the next message out an unused link.
- To integrate the load distribution algorithm work into the hardware forwarding ASIC so that load distribution works just as quickly as the work to forward any other frame.
- To use all the active links in the EtherChannel, adjusting to the addition and removal of active links over time.
- Within the constraints of the other goals, balance the traffic across those active links.

In short, the algorithms first intend to avoid message reordering, make use of the switch forwarding ASICs, and use all the active links. However, the algorithm does not attempt to send the exact same number of bits over each link over time. The algorithm does try to balance the traffic, but always within the constraints of the other goals.

Whatever load distribution method you choose, the method identifies fields in the message headers. Any messages in the same application flow will then have the same values in the fields used by the load distribution algorithm and will always be forwarded over the same link. For example, when a user connects to a website, that web server may return thousands of packets to the client. Those thousands of packets should flow over the same link in the EtherChannel.

For instance, with the load distribution method of **src-mac** (meaning source MAC address), all frames with the same MAC address flow over one link. Figure 10-13 shows the idea with pseudo (generic) MAC addresses, with the load distribution sending frames with source MAC 1 over link 1, source MAC 2 over link 2, and source MAC 3 over link 3.

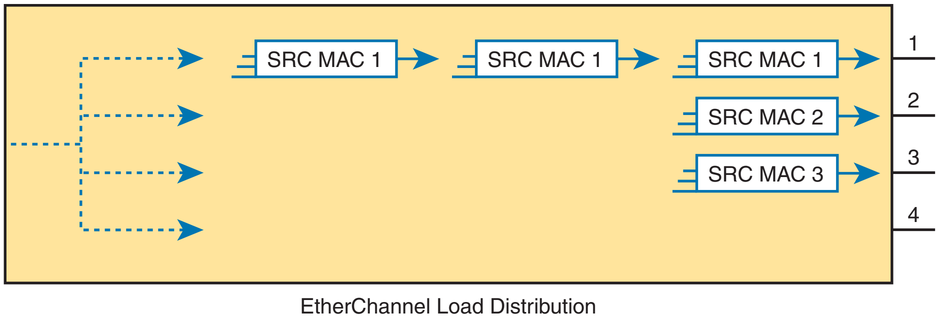


Figure 10-13 *Distributing All Frames with the Same MAC Out the Same Interface*

Cisco provides a variety of load distribution options so that the engineer can examine the flows in the network with the idea of finding which fields have the most variety in their values: source and destination MAC, or IP address, or transport layer port numbers. The more variety in the values in the fields, the better the balancing effects, and the lower the chance of sending disproportionate amounts of traffic over one link.

NOTE The algorithm focuses on the low-order bits in the fields in the headers because the low-order bits typically differ the most in real networks, while the high-order bits do not differ much. By focusing on the lower-order bits, the algorithm achieves better balancing of traffic over the links.

Chapter Review

One key to doing well on the exams is to perform repetitive spaced review sessions. Review this chapter's material using either the tools in the book or interactive tools for the same material found on the book's companion website. Refer to the "Your Study Plan" element for more details. Table 10-5 outlines the key review elements and where you can find them. To better track your study progress, record when you completed these activities in the second column.

Table 10-5 Chapter Review Tracking

Review Element	Review Date(s)	Resource Used
Review key topics		Book, website
Review key terms		Book, website
Answer DIKTA questions		Book, PTP
Review config checklists		Book, website
Review command tables		Book
Review memory tables		Website
Do labs		Blog
Watch video		Website

Review All the Key Topics

Key Topic

Table 10-6 Key Topics for Chapter 10

Key Topic Element	Description	Page Number
Figure 10-1	Typical design choice for which switches should be made to be root	259
Figure 10-2	Conceptual view of load-balancing benefits of PVST+	260
Table 10-2	STP Standards and Configuration Options	261
Figure 10-4	Shows the format of the system ID extension of the STP priority field	262
List	Facts about RPVST+'s methods versus RSTP	266
List	Trigger, actions, and recovery for BPDU Guard	270
List	Trigger, actions, and recovery for Root Guard	278
List	Configuration options with Loop Guard	280
List	Steps to manually configure an EtherChannel	281
List	Items a switch compares in a new physical port's configuration to the existing ports in the channel	287

Key Terms You Should Know

Channel-group, EtherChannel, EtherChannel load distribution, LACP, PAgP, PortChannel, PVST+, Rapid PVST+, system ID extension

Command References

Tables 10-7 and 10-8 list configuration and verification commands used in this chapter. As an easy review exercise, cover the left column in a table, read the right column, and try to recall the command without looking. Then repeat the exercise, covering the right column, and try to recall what the command does.

Table 10-7 Chapter 10 Configuration Command Reference

Command	Description
spanning-tree mode {**pvst** | **rapid-pvst** | **mst**}	Global configuration command to set the STP mode.
spanning-tree [**vlan** *vlan-number*] **root primary**	Global configuration command that changes this switch to the root switch. The switch's priority is changed to the lower of either 24,576 or 4096 less than the priority of the current root bridge when the command was issued.
spanning-tree [**vlan** *vlan-number*] **root secondary**	Global configuration command that sets this switch's STP base priority to 28,672.
spanning-tree vlan *vlan-id* **priority** *priority*	Global configuration command that changes the bridge priority of this switch for the specified VLAN.
spanning-tree [**vlan** *vlan-number*] **cost** *cost*	Interface subcommand that changes the STP cost to the configured value.

Command	Description
spanning-tree [**vlan** *vlan-number*] **port-priority** *priority*	Interface subcommand that changes the STP port priority in that VLAN (0 to 240, in increments of 16).
spanning-tree portfast	Interface subcommand that enables PortFast if the port is also an access port.
spanning-tree portfast trunk	Interface subcommand that enables PortFast if the port is also a trunk port.
spanning-tree bpduguard enable	Interface subcommand that enables BPDU Guard on the interface under all conditions.
spanning-tree portfast disable	Interface subcommand that reverses the **spanning-tree portfast** command.
spanning-tree bpduguard disable	Interface subcommand that reverses the **spanning-tree bpduguard enable** command.
spanning-tree portfast enable	Global command that changes the default interface setting to the same logic as if the **spanning-tree portfast** interface subcommand were configured.
spanning-tree portfast bpduguard default	Global command that changes the default interface setting to enable BPDU Guard if the port is also actively using PortFast.
spanning-tree bpdufilter enable	Interface subcommand that enables BPDU Filter on the interface under all conditions, disabling STP on the interface.
spanning-tree portfast bpdufilter default	Global command that directs IOS to enable BPDU Filter conditional logic, which toggles away from using PortFast as needed if the port is also actively using PortFast.
[**no**] **spanning-tree guard root**	Interface subcommand to enable or disable (with the **no** option) Root Guard.
[**no**] **spanning-tree guard loop**	Interface subcommand to enable or disable (with the **no** option) Loop Guard.
spanning-tree loopguard default	Global command to change the default setting on interfaces to enable Loop Guard.
channel-group *channel-group-number* **mode** {**auto** | **desirable** | **active** | **passive** | **on**}	Interface subcommand that enables EtherChannel on the interface.

Table 10-8 Chapter 10 EXEC Command Reference

Command	Description
show spanning-tree	Lists details about the state of STP on the switch, including the state of each port.
show spanning-tree vlan *vlan-id*	Lists STP information for the specified VLAN.
show spanning-tree vlan *vlan-id* **interface** *interface-id* [**detail**]	Lists STP information for the specified VLAN about the specific interface.

Command	Description
show etherchannel [*channel-group-number*] {**brief** | **detail** | **port** | **port-channel** | **summary**}	Lists information about the state of EtherChannels on this switch.
show interfaces portchannel *number*	Lists information typical of the **show interfaces** command and also lists the interfaces included in the EtherChannel.
show etherchannel	Displays configuration settings for each EtherChannel.
show etherchannel [*number*] {**summary** | **portchannel** | **detail**}	Displays status information about all EtherChannels, or the one specific EtherChannel. The final parameter suggests the briefest option (**summary**) to the most detailed (**detail**).

This page intentionally left blank

Part III Review

Keep track of your part review progress with the checklist shown in Table P3-1. Details on each task follow the table.

Table P3-1 Part III Part Review Checklist

Activity	1st Date Completed	2nd Date Completed
Repeat All DIKTA Questions		
Answer Part Review Questions		
Review Key Topics		
Do Labs		
Review Appendices		
Watch Video		
Use Per-Chapter Interactive Review		

Repeat All DIKTA Questions

For this task, answer the “Do I Know This Already?” questions again for the chapters in this part of the book, using the PTP software.

Answer Part Review Questions

For this task, answer the Part Review questions for this part of the book, using the PTP software.

Review Key Topics

Review all key topics in all chapters in this part, either by browsing the chapters or by using the Key Topics application on the companion website.

Do Labs

Depending on your chosen lab tool, here are some suggestions for what to do in the labs:

Pearson Network Simulator: If you use the full Pearson CCNA simulator, focus more on the configuration scenario and troubleshooting scenario labs associated with the topics in this part of the book. These types of labs include a larger set of topics and work well as Part Review activities. (See the Introduction for some details about how to find which labs are about topics in this part of the book.) Note that the Sim Lite that comes with this book also has a couple of labs about VLANs.

Blog: Config Labs: The author’s blog includes a series of configuration-focused labs that you can do on paper or with Cisco Packet Tracer in about 15 minutes. To find them, open https://www.certskills.com and look under the Labs menu item.

Other: If using other lab tools, as a few suggestions: make sure to experiment heavily with VLAN configuration and VLAN trunking configuration.

Dig Deeper with Appendices on the Companion Website

The chapters in Part III of the book recommended the following appendices for extra reading. If you care to read further, consider

- **Appendix L, "LAN Troubleshooting":** An appendix from the previous edition of the book. It includes topics about VLANs, trunks, and STP and how to troubleshoot each.

Watch Video

The companion website includes a variety of common mistake and Q&A videos organized by part and chapter. Use these videos to challenge your thinking, dig deeper, review topics, and better prepare for the exam. Make sure to bookmark a link to the companion website and use the videos for review whenever you have a few extra minutes.

Use Per-Chapter Interactive Review

Using the companion website, browse through the interactive review elements, like memory tables and key term flashcards, to review the content from each chapter.