This section begins with a short discussion about VLSM design, to drive home the ideas behind VLSM overlaps. It then gets into an operational and troubleshooting approach to the topic, by looking at existing designs and trying to find any existing overlaps.

Designing Subnetting Plans with VLSM

When creating a subnetting plan using VLSM, you have to be much more careful in choosing what subnets to use. First, whatever masks you use in a VLSM design, each subnet ID must be a valid subnet ID given the mask that you use for that subnet.

For example, consider a subnet plan for Class B network 172.16.0.0. To create a subnet with a /24 mask, the subnet ID must be a subnet ID that you could choose if you subnetted the whole Class B network with that same mask. Appendix L, "Subnet Design," discusses how to find those subnets in depth, but with a Class B network and a /24 mask, the possible subnet IDs should be easy to calculate by now: 172.16.0.0 (the zero subnet), then 172.16.1.0, 172.16.2.0, 172.16.3.0, 172.16.4.0, and so on, up through 172.16.255.0.

NOTE Subnet IDs must always follow this important binary rule as noted back in Chapter 14, "Analyzing Existing Subnets": In binary, each subnet ID has a host field of all binary 0s. If you use the math and processes to find all subnet IDs per Appendix L, all those subnet IDs happen to have binary 0s in the host fields.

N

Now expand your thinking about subnet IDs to a VLSM design. To begin, you would decide that you need some subnets with one mask, other subnets with another mask, and so on, to meet the requirements for different sizes of different subnets. For instance, imagine you start with a brand-new VLSM design, with Class B network 172.16.0.0. You plan to have some subnets with /22 masks, some with /23, and some with /24. You might develop then a planning diagram, or at least draw the ideas, with something like Figure N-2.

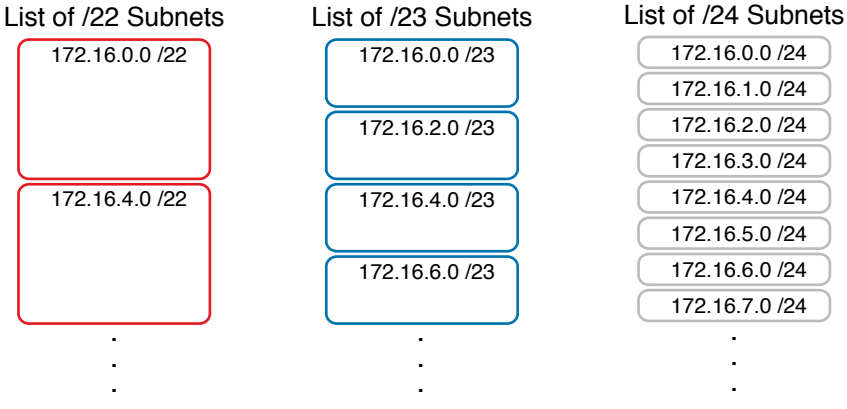

Figure N-2 *Possible Subnet IDs of Network 172.16.0.0, with /22, /23, and /24 Masks*

The drawing shows the first few subnet IDs available with each mask, but you cannot use all subnets from all three lists in a design. As soon as you choose to use one subnet from any column, you remove some subnets from the other lists because subnets cannot overlap. Overlapping subnets are subnets whose range of addresses include some of the same addresses.

As an example, Figure N-3 shows the same list of the first few possible /22, /23, and /24 subnets of Class B network 172.16.0.0. However, it shows a check mark beside two subnets that have been allocated for use; that is, on paper, the person making the subnetting plan has decided to use these two subnets somewhere in the network. The subnets with a dark gray shading and an *X* in them can no longer be used because they have some overlapping addresses with the subnets that have check marks (172.16.3.0/24 and 172.16.4.0/22).

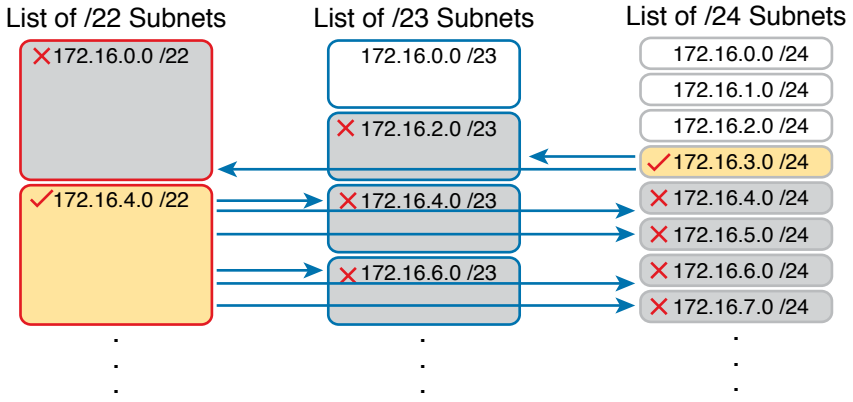

Figure N-3 *Selecting Two Subnets Disallows Other Subnets in Different Columns*

Just to complete the example, first look at subnet 172.16.4.0 on the lower left. That subnet includes addresses from the subnet ID of 172.16.4.0 through the subnet broadcast address of 172.16.7.255. As you can see just by looking at the subnet IDs to the right, all the subnets referenced with the arrowed lines are within that same range of addresses.

Now look to the upper right of the figure, to subnet 172.16.3.0/24. The subnet has a range of 172.16.3.0–172.16.3.255 including the subnet ID and subnet broadcast address. That subnet overlaps with the two subnets referenced to the left. For instance, subnet 172.16.0.0/22 includes the range from 172.16.0.0–172.16.3.255. But because there is some overlap, once the design has allocated the 172.16.3.0/24 subnet, the 172.16.2.0/23 and 172.16.0.0/22 subnets could not be used without causing problems, because:

A subnetting design, whether using VLSM or not, should not allow subnets whose address ranges overlap. If overlapping subnets are implemented, routing problems occur and some hosts simply cannot communicate outside their subnets.

These address overlaps are easier to see when not using VLSM. When not using VLSM, overlapped subnets have identical subnet IDs, so to find overlaps, you just have to look at the subnet IDs. With VLSM, overlapped subnets may not have the same subnet ID, as was the case in this most recent example with the subnets across the top of Figure N-3. To find these overlaps, you have to look at the entire range of addresses in each subnet, from subnet ID to subnet broadcast address, and compare the range to the other subnets in the design.

An Example of Finding a VLSM Overlap

For example, imagine that a practice question for the CCENT exam shows Figure N-4. It uses a single Class B network (172.16.0.0), with VLSM, because it uses three different masks: /23, /24, and /30.

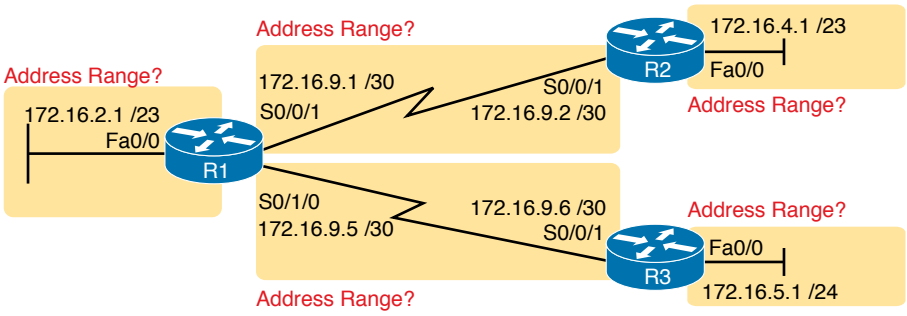

Figure N-4 *VLSM Design with Possible Overlap*

Now imagine that the exam question shows you the figure, and either directly or indirectly asks whether overlapping subnets exist. This type of question might simply tell you that some hosts cannot ping each other, or it might not even mention that the root cause could be that some of the subnets overlap. To answer such a question, you could follow this simple but possibly laborious process:

N

Key Topic

Step 1. Calculate the subnet ID and subnet broadcast address of each subnet, which gives you the range of addresses in that subnet.

Step 2. List the subnet IDs in numerical order (along with their subnet broadcast addresses).

Step 3. Scan the list from top to bottom, comparing each pair of adjacent entries, to see whether their range of addresses overlaps.

For example, Table N-2 completes the first two steps based on Figure N-4, listing the subnet IDs and subnet broadcast addresses, in numerical order based on the subnet IDs.

Table N-2 Subnet IDs and Broadcast Addresses, in Numerical Order, from Figure N-4

Subnet	Subnet Number	Broadcast Address
R1 LAN	172.16.2.0	172.16.3.255
R2 LAN	172.16.4.0	172.16.5.255
R3 LAN	172.16.5.0	172.16.5.255
R1-R2 serial	172.16.9.0	172.16.9.3
R1-R3 serial	172.16.9.4	172.16.9.7

The VLSM design is invalid in this case because of the overlap between R2's LAN subnet and R3's LAN subnet. As for the process, Step 3 states the somewhat obvious step of comparing the address ranges to see whether any overlaps occur. Note that, in this case, none of the subnet numbers are identical, but two entries (highlighted) do overlap. The design is invalid because of the overlap, and one of these two subnets would need to be changed.

As far as the three-step process works, note that if two adjacent entries in the list overlap, compare three entries at the next step. The two subnets already marked as overlapped can overlap with the next subnet in the list. For example, the three subnets in the following list overlap in that the first subnet overlaps with the second and third subnets in the list. If you

followed the process shown here, you would have first noticed the overlap between the first two subnets in the list, so you would then also need to check the next subnet in the list to find out if it overlapped.

10.1.0.0/16 (subnet ID 10.1.0.0, broadcast 10.1.255.255)

10.1.200.0/24 (subnet ID 10.1.200.0, broadcast 10.1.200.255)

10.1.250.0/24 (subnet ID 10.1.250.0, broadcast 10.1.250.255)

Practice Finding VLSM Overlaps

As typical of anything to with applying IP addressing and subnetting, practice helps. To that end, Table N-3 lists three practice problems. Just start with the five IP addresses listed in a single column, and then follow the three-step process outlined in the previous section to find any VLSM overlaps. The answers can be found near the end of this chapter, in the section "Answers to Earlier Practice Problems."

Table N-3 VLSM Overlap Practice Problems

Problem 1	Problem 2	Problem 3
10.1.34.9/22	172.16.126.151/22	192.168.1.253/30
10.1.29.101/23	172.16.122.57/27	192.168.1.113/28
10.1.23.254/22	172.16.122.33/30	192.168.1.245/29
10.1.17.1/21	172.16.122.1/30	192.168.1.125/30
10.1.1.1/20	172.16.128.151/20	192.168.1.122/30

Adding a New Subnet to an Existing VLSM Design

The task described in this section happens frequently in real networks: choosing new subnets to add to an existing design. In real life, you can use IP Address Management (IPAM) tools that help you choose a new subnet so that you do not cause an overlap. However, for the CCNA exam, you need to be ready to do the mental process and math of choosing a subnet that does not create an overlapped VLSM subnet condition. In other words, you need to pick a new subnet and not make a mistake!

For example, consider the internetwork shown earlier in Figure N-2, with classful network 172.16.0.0. An exam question might suggest that a new subnet, with a /23 prefix length, needs to be added to the design. The question might also say, "Pick the numerically lowest subnet number that can be used for the new subnet." In other words, if both 172.16.4.0 and 172.16.6.0 would work, use 172.16.4.0.

So, you really have a couple of tasks: To find all the subnet IDs that could be used, rule out the ones that would cause an overlap, and then check to see whether the question guides you to pick either the numerically lowest (or highest) subnet ID. This list outlines the specific steps:

Step 1. Pick the subnet mask (prefix length) for the new subnet, based on the design requirements (if not already listed as part of the question).

Step 2. Calculate all possible subnet numbers of the classful network using the mask from Step 1, along with the subnet broadcast addresses.

Step 3. Make a list of existing subnet IDs and matching subnet broadcast addresses.

Step 4. Compare the existing subnets to the candidate new subnets to rule out overlapping new subnets.

Step 5. Choose the new subnet ID from the remaining subnets identified at Step 4, paying attention to whether the question asks for the numerically lowest or numerically highest subnet ID.

An Example of Adding a New VLSM Subnet

For example, Figure N-5 shows an existing internetwork that uses VLSM. (The figure uses the same IP addresses as shown in Figure N-4, but with R3's LAN IP address changed to fix the VLSM overlap shown in Figure N-4.) In this case, you need to add a new subnet to support 300 hosts. Imagine that the question tells you to use the smallest subnet (least number of hosts) to meet that requirement. You use some math and logic you learned earlier in your study to choose mask /23, which gives you 9 host bits, for $2^9 - 2 = 510$ hosts in the subnet.

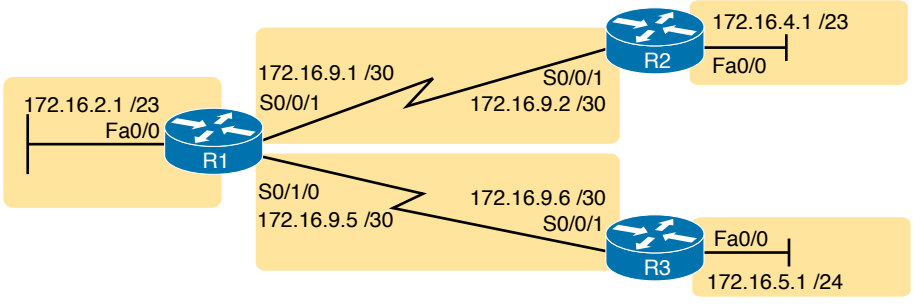

Figure N-5 *Internetwork to Which You Need to Add a /23 Subnet, Network 172.16.0.0*

At this point, just follow the steps listed before Figure N-5. For Step 1, you have already been given the mask (/23). For Step 2, you need to list all the subnet numbers and broadcast addresses of 172.16.0.0, assuming the /23 mask. You will not use all these subnets, but you need the list for comparison to the existing subnets. Table N-4 shows the results, at least for the first five possible /23 subnets.

Table N-4 First Five Possible /23 Subnets

Subnet	Subnet Number	Subnet Broadcast Address
First (zero)	172.16.0.0	172.16.1.255
Second	172.16.2.0	172.16.3.255
Third	172.16.4.0	172.16.5.255
Fourth	172.16.6.0	172.16.7.255
Fifth	172.16.8.0	172.16.9.255

Next, at Step 3, list the existing subnet numbers and broadcast addresses, as shown earlier in Figure N-5. To do so, do the usual math to take an IP address/mask to then find the subnet ID and subnet broadcast address. Table N-5 summarizes that information, including the locations, subnet numbers, and subnet broadcast addresses.

Table N-5 Existing Subnet IDs and Broadcast Addresses from Figure N-5

Subnet	Subnet Number	Subnet Broadcast Address
R1 LAN	172.16.2.0	172.16.3.255
R2 LAN	172.16.4.0	172.16.5.255
R3 LAN	172.16.6.0	172.16.6.255
R1-R2 serial	172.16.9.0	172.16.9.3
R1-R3 serial	172.16.9.4	172.16.9.7

At this point, you have all the information you need to look for the overlap at Step 4. Simply compare the range of numbers for the subnets in the previous two tables. Which of the possible new /23 subnets (Table N-4) overlap with the existing subnets (Table N-5)? In this case, the second through fifth subnets in Table N-4 overlap, so rule those out as candidates to be used. (Table N-4 denotes those subnets with gray highlights.)

Step 5 has more to do with the exam than with real network design, but it is still worth listing as a separate step. Multiple-choice questions sometimes need to force you into a single answer, and asking for the numerically lowest or highest subnet does that. This particular example asks for the numerically lowest subnet number, which in this case is 172.16.0.0/23.

NOTE The answer, 172.16.0.0/23, happens to be a zero subnet. For the exam, the zero subnet should be avoided if (a) the question implies the use of classful routing protocols or (b) the routers are configured with the **no ip subnet-zero** global configuration command. Otherwise, assume that the zero subnet can be used.

Answers to Earlier Practice Problems

Answers to Practice Finding VLSM Overlaps

This section lists the answers to the three practice problems in the section "Practice Finding VLSM Overlaps," as listed earlier in Table N-3. Note that the tables that list details of the answer reordered the subnets as part of the process.

In Problem 1, the second and third subnet IDs listed in Table N-6 happen to overlap. The second subnet's range completely includes the range of addresses in the third subnet.

Table N-6 VLSM Overlap Problem 1 Answers (Overlaps Highlighted)

Reference	Original Address and Mask	Subnet ID	Broadcast Address
1	10.1.1.1/20	10.1.0.0	10.1.15.255
2	10.1.17.1/21	10.1.16.0	10.1.23.255
3	10.1.23.254/22	10.1.20.0	10.1.23.255
4	10.1.29.101/23	10.1.28.0	10.1.29.255
5	10.1.34.9/22	10.1.32.0	10.1.35.255

In Problem 2, again the second and third subnet IDs (listed in Table N-7) happen to overlap, and again, the second subnet's range completely includes the range of addresses in the third subnet. Also, the second and third subnet IDs are the same value, so the overlap is more obvious.

Table N-7 VLSM Overlap Problem 2 Answers (Overlaps Highlighted)

Reference	Original Address and Mask	Subnet ID	Broadcast Address
1	172.16.122.1/30	172.16.122.0	172.16.122.3
2	172.16.122.57/27	172.16.122.32	172.16.122.63
3	172.16.122.33/30	172.16.122.32	172.16.122.35
4	172.16.126.151/22	172.16.124.0	172.16.127.255
5	172.16.128.151/20	172.16.128.0	172.16.143.255

In Problem 3, three subnets overlap. Subnet 1's range completely includes the range of addresses in the second and third subnets, as shown in Table N-8. Note that the second and third subnets do not overlap with each other, so for the process in this book to find all the overlaps, after you find that the first two subnets overlap, you should compare the next entry in the table (3) with both of the two known-to-overlap entries (1 and 2).

Table N-8 VLSM Overlap Problem 3 Answers (Overlaps Highlighted)

Reference	Original Address and Mask	Subnet ID	Broadcast Address
1	192.168.1.113/28	192.168.1.112	192.168.1.127
2	192.168.1.122/30	192.168.1.120	192.168.1.123
3	192.168.1.125/30	192.168.1.124	192.168.1.127
4	192.168.1.245/29	192.168.1.240	192.168.1.247
5	192.168.1.253/30	192.168.1.252	192.168.1.255

This page intentionally left blank

APPENDIX O

Spanning Tree Protocol Implementation

NOTE This appendix contains an entire chapter that was published as a chapter in one of the past editions of this book or a related book. The author includes this appendix with the current edition as extra reading for anyone interested in learning more. However, note that the content in this appendix has not been edited since it was published in the earlier edition, so references to exams and exam topics, and to other chapters, will be outdated. This appendix was previously published as Chapter 3 of the book *CCNA ICND2 200-105 Official Cert Guide*, published in 2016.

Cisco IOS–based LAN switches enable Spanning Tree Protocol (STP) by default on all interfaces in every VLAN. However, network engineers who work with medium-size to large-size Ethernet LANs usually want to configure at least some STP settings. First and foremost, Cisco IOS switches traditionally default to use STP rather than Rapid STP (RSTP), and the simple upgrade to RSTP improves convergence. For most LANs with more than a few switches, the network engineer will likely want to influence the choices made by STP, whether using traditional STP or RSTP—choices such as which switch becomes root, with predictability about which switch ports will block/discard when all ports are physically working. The configuration can also be set so that when links or switches fail, the engineer can predict the STP topology in those cases, as well.

This chapter discusses configuration and verification of STP. The first major section weaves a story of how to change different settings, per VLAN, with the **show** commands that reveal the current STP status affected by each configuration command. Those settings impact both STP and RSTP, but the examples use switches that use traditional 802.1D STP rather than RSTP. The second major section shows how to configure the optional STP features PortFast, BPDU Guard, and EtherChannel (specifically Layer 2 EtherChannel). The final major section of this chapter looks at the simple (one command) configuration to enable RSTP, and the differences and similarities in **show** command output that occur when using RSTP versus STP.

Foundation Topics

Implementing STP

Cisco IOS switches usually use STP (IEEE 802.1D) by default rather than RSTP, and with effective default settings. You can buy some Cisco switches and connect them with Ethernet cables in a redundant topology, and STP will ensure that frames do not loop. And you never even have to think about changing any settings!

Although STP works without any configuration, most medium-size to large-size campus LANs benefit from some STP configuration. With all defaults, the switches choose the root based on the lowest burned-in MAC address on the switches because they all default to use the same STP priority. As a better option, configure the switches so that the root is predictable.

For instance, Figure O-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. For instance, the configuration could make D1 be the root by having a lower priority, with D2 configured with the next lower priority, so it becomes root if D1 fails.

Key Topic

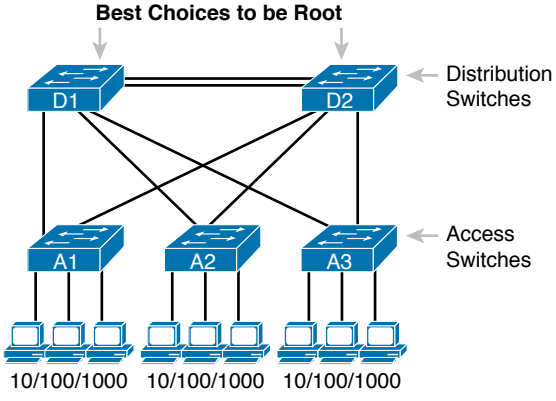

Figure O-1 *Typical Configuration Choice: Making Distribution Switch Be Root*

This first section of the chapter examines a variety of topics that somehow relate to STP configuration. It begins with a look at STP configuration options, as a way to link the concepts of Chapter 2 to the configuration choices in this chapter. Following that, this section introduces some **show** commands for the purpose of verifying the default STP settings before changing any configuration.

Setting the STP Mode

The IEEE first standardized STP as the IEEE 802.1D standard, first published back in 1990. To put some perspective on that date, Cisco sold no LAN switches at the time, and virtual LANs did not exist yet. Instead of multiple VLANs in a LAN, there was just one broadcast domain, and one instance of STP. However, the addition of VLANs and the introduction of LAN switches into the market have created a need to add to and extend STP.

Today, Cisco IOS–based LAN switches allow you to use one of three STP configuration modes that reflect that history. The first two sections of this chapter use the mode called Per-VLAN Spanning Tree Plus (PVST+, or sometimes PVSTP), a Cisco-proprietary improvement of 802.1D STP. The *per-VLAN* part of the name gives away the main feature: PVST+ creates a different STP topology per VLAN, whereas 802.1D actually did not. PVST+ also introduced PortFast. Cisco switches often use PVST+ as the default STP mode per a default global command of **spanning-tree mode pvst**.

Over time, Cisco added RSTP support as well, with two STP modes that happen to use RSTP. One mode basically takes PVST+ and upgrades it to use RSTP logic as well, with a mode called *Rapid PVST+*, enabled with the global command **spanning-tree mode rapid-pvst**. Cisco IOS–based switches support a third mode, called Multiple Spanning Tree (MST) (or Multiple Instance of Spanning Tree), enabled with the **spanning-tree mode mst** command.

Connecting STP Concepts to STP Configuration Options

STP uses two types of numbers for most of its decisions: the BID and STP port costs. Focusing on those two types of numbers, consider this summary of what STP does behind the scenes:

- Uses the BID to elect the root switch, electing the switch with the numerically lowest BID
- Uses the total STP cost in each path to the root, when each nonroot switch chooses its own root port (RP)
- Uses each switch's root cost, which is in turn based on STP port costs, when switches decide which switch port becomes the designated port (DP) on each LAN segment

Unsurprisingly, Cisco switches let you configure part of a switch's BID and the STP port cost, which in turn influences the choices each switch makes with STP.

Per-VLAN Configuration Settings

Beyond supporting the configuration of the BID and STP port costs, Cisco switches support configuring both settings per VLAN. By default, Cisco switches use IEEE 802.1D, not RSTP (802.1w), with a Cisco-proprietary feature called Per-VLAN Spanning Tree Plus (PVST+). PVST+ (often abbreviated as simply PVST today) creates a different instance of STP for each VLAN. So, before looking at the tunable STP parameters, you need to have a basic understanding of PVST+, because the configuration settings can differ for each instance of STP.

PVST+ gives engineers a load-balancing tool with STP. By changing some STP configuration parameters differently for different VLANs, the engineer could cause switches to pick different RPs and DPs in different VLANs. As a result, some traffic in some VLANs can be forwarded over one trunk, and traffic for other VLANs can be forwarded over a different trunk.

Figure O-2 shows the basic idea, with SW3 forwarding odd-numbered VLAN traffic over the left trunk (Gi0/1) and even-numbered VLANs over the right trunk (Gi0/2).

Key Topic

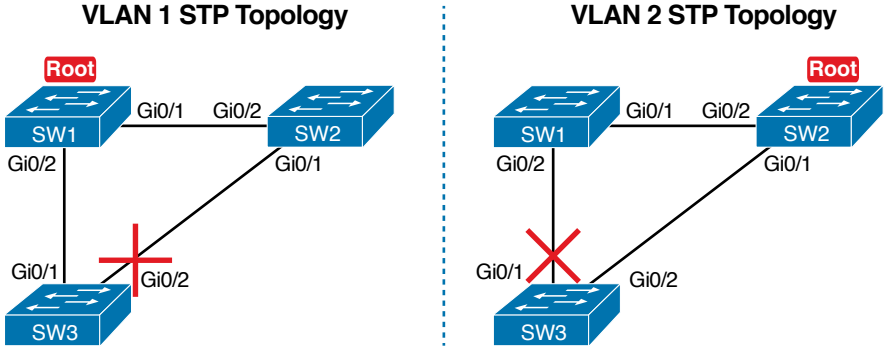

Figure O-2 *Load Balancing with PVST+*

The next few pages look specifically at how to change the BID and STP port cost settings, per VLAN, when using the default PVST+ mode.

The Bridge ID and System ID Extension

Originally, a switch's BID was formed by combining the switch's 2-byte priority and its 6-byte MAC address. Later, the IEEE changed the rules, splitting the original priority field into two separate fields, as shown in Figure O-3: a 4-bit priority field and a 12-bit subfield called the *system ID extension* (which represents the VLAN ID).

Key Topic

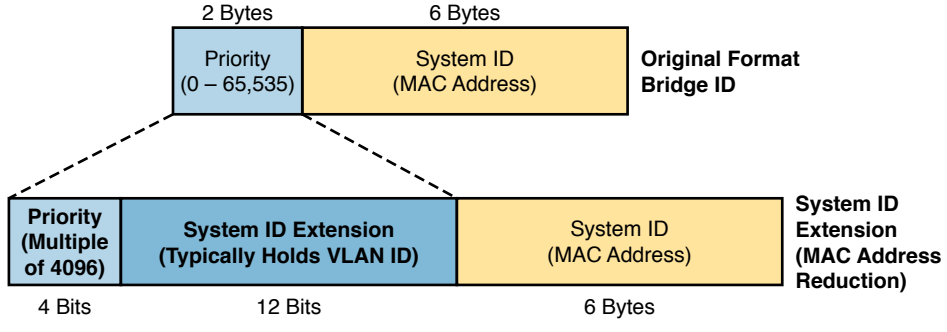

Figure O-3 *STP System ID Extension*

Cisco switches let you configure the BID, but only the priority part. 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. The only part configurable by the network engineer is the 4-bit priority field.

Configuring the number to put in the priority field, however, is one of the strangest things to configure on a Cisco router or switch. As shown at the top of Figure O-3, the priority field was originally a 16-bit number, which represented a decimal number from 0 to 65,535. Because of that history, the current configuration command (**spanning-tree vlan** *vlan-id* **priority** *x*) requires a decimal number between 0 and 65,535. But not just any number in that range will suffice—it must be a multiple of 4096: 0, 4096, 8192, 12288, and so on, up through 61,440.

The switch still sets the first 4 bits of the BID based on the configured value. As it turns out, of the 16 allowed multiples of 4096, from 0 through 61,440, each has a different binary value in their first 4 bits: 0000, 0001, 0010, and so on, up through 1111. The switch sets the true 4-bit priority based on the first 4 bits of the configured value.

Although the history and configuration might make the BID priority idea seem a bit convoluted, having an extra 12-bit field in the BID works well in practice because it can be used to identify the VLAN ID. VLAN IDs range from 1 to 4094, requiring 12 bits. Cisco switches place the VLAN ID into the system ID extension field, so each switch has a unique BID per VLAN.

For example, a switch configured with VLANs 1 through 4, with a default base priority of 32,768, has a default STP priority of 32,769 in VLAN 1, 32,770 in VLAN 2, 32,771 in VLAN 3, and so on. So, you can view the 16-bit priority as a base priority (as configured in the **spanning-tree vlan** *vlan-id* **priority** *x* command) plus the VLAN ID.

NOTE Cisco switches must use the system ID extension version of the bridge ID; it cannot be disabled.

Per-VLAN Port Costs

Each switch interface defaults its per-VLAN STP cost based on IEEE recommendations. On interfaces that support multiple speeds, Cisco switches base the cost on the current actual speed. So, if an interface negotiates to use a lower speed, the default STP cost reflects that lower speed. If the interface negotiates to use a different speed, the switch dynamically changes the STP port cost as well.

Alternatively, you can configure a switch's STP port cost with the **spanning-tree** [**vlan** *vlan-id*] **cost** *cost* interface subcommand. You see this command most often on trunks because setting the cost on trunks has an impact on the switch's root cost, whereas setting STP costs on access ports does not.

For the command itself, it can include the VLAN ID, or not. The command only needs a **vlan** parameter on trunk ports to set the cost per VLAN. On a trunk, if the command omits the VLAN parameter, it sets the STP cost for all VLANs whose cost is not set by a **spanning-tree vlan** *x* **cost** command for that VLAN.

STP Configuration Option Summary

Table O-1 summarizes the default settings for both the BID and the port costs and lists the optional configuration commands covered in this chapter.

Key Topic

Table O-1 STP Defaults and Configuration Options

Setting	Default	Command(s) to Change Default
BID priority	Base: 32,768	**spanning-tree vlan** *vlan-id* **root** {**primary** | **secondary**} **spanning-tree vlan** *vlan-id* **priority** *priority*
Interface cost	100 for 10 Mbps 19 for 100 Mbps 4 for 1 Gbps 2 for 10 Gbps	**spanning-tree vlan** *vlan-id* **cost** *cost*
PortFast	Not enabled	**spanning-tree portfast**
BPDU Guard	Not enabled	**spanning-tree bpduguard enable**

Next, the configuration section shows how to examine the operation of STP in a simple network, along with how to change these optional settings.

Verifying STP Operation

Before taking a look at how to change the configuration, first consider a few STP verification commands. Looking at these commands first will help reinforce the default STP settings. In particular, the examples in this section use the network shown in Figure O-4.

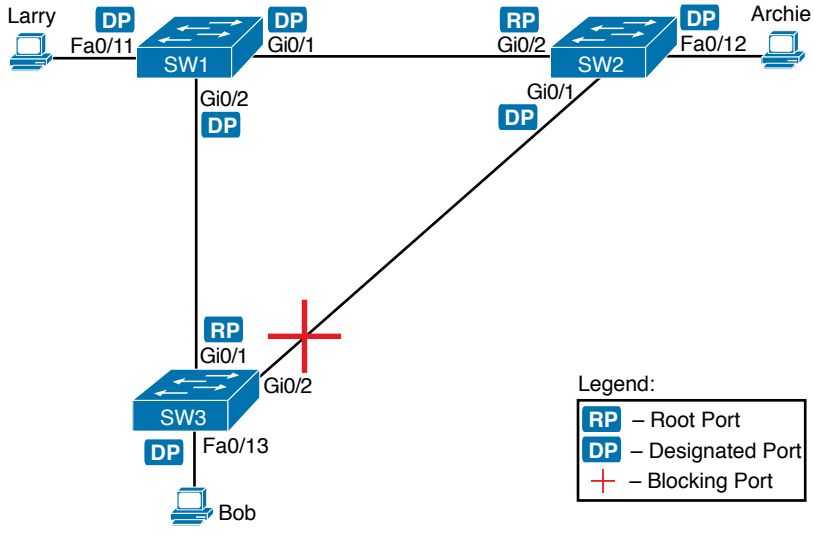

Figure O-4 *Sample LAN for STP Configuration and Verification Examples*

Example O-1 begins the discussion with a useful command for STP: the **show spanning-tree vlan 10** command. This command identifies the root switch and lists settings on the local switch. Example O-1 lists the output of this command on both SW1 and SW2, as explained following the example.

Example O-1 *STP Status with Default STP Parameters on SW1 and SW2*

```
SW1# show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID     Priority    32778
              Address     1833.9d7b.0e80
              This bridge is the root
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     1833.9d7b.0e80
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Fa0/11              Desg FWD 19        128.11   P2p Edge
Gi0/1               Desg FWD 4         128.25   P2p
Gi0/2               Desg FWD 4         128.26   P2p
```

```
SW2# show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID     Priority    32778
              Address     1833.9d7b.0e80
              Cost        4
              Port        26 (GigabitEthernet0/2)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     1833.9d7b.1380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Fa0/12              Desg FWD 19        128.12   P2p
Gi0/1               Desg FWD 4         128.25   P2p
Gi0/2               Root FWD 4         128.26   P2p
```

Example O-1 begins with the output of the **show spanning-tree vlan 10** command on SW1. This command first lists three major groups of messages: one group of messages about the root switch, followed by another group about the local switch, and ending with interface role and status information. In this case, SW1 lists its own BID as the root, with

even a specific statement that "This bridge is the root," confirming that SW1 is now the root of the VLAN 10 STP topology.

Next, compare the highlighted lines of the same command on SW2 in the lower half of the example. SW2 lists SW1's BID details as the root; in other words, SW2 agrees that SW1 has won the root election. SW2 does not list the phrase "This bridge is the root." SW2 then lists its own (different) BID details in the lines after the details about the root's BID.

The output also confirms a few default values. First, each switch lists the priority part of the BID as a separate number: 32778. This value comes from the default priority of 32768, plus VLAN 10, for a total of 32778. The output also shows the interface cost for some Fast Ethernet and Gigabit Ethernet interfaces, defaulting to 19 and 4, respectively.

Finally, the bottom of the output from the **show spanning-tree** command lists each interface in the VLAN, including trunks, with the STP port role and port state listed. For instance, on switch SW1, the output lists three interfaces, with a role of Desg for designated port (DP) and a state of FWD for forwarding. SW2 lists three interfaces, two DPs, and one root port, so all three are in an FWD or forwarding state.

Example O-1 shows a lot of good STP information, but two other commands, shown in Example O-2, work better for listing BID information in a shorter form. The first, **show spanning-tree root**, lists the root's BID for each VLAN. This command also lists other details, like the local switch's root cost and root port. The other command, **show spanning-tree vlan 10 bridge**, breaks out the BID into its component parts. In this example, it shows SW2's priority as the default of 32768, the VLAN ID of 10, and the MAC address.

Example O-2 *Listing Root Switch and Local Switch BIDs on Switch SW2*

```
SW2# show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                Root ID             Cost    Time  Age Dly  Root Port
---------------- ------------------- --------- ----- --- ---  ------------
VLAN0001         32769 1833.9d5d.c900        23    2   20  15  Gi0/1
VLAN0010         32778 1833.9d7b.0e80         4    2   20  15  Gi0/2
VLAN0020         32788 1833.9d7b.0e80         4    2   20  15  Gi0/2
VLAN0030         32798 1833.9d7b.0e80         4    2   20  15  Gi0/2
VLAN0040         32808 1833.9d7b.0e80         4    2   20  15  Gi0/2

SW2# show spanning-tree vlan 10 bridge

                                               Hello  Max  Fwd
Vlan                        Bridge ID           Time  Age  Dly  Protocol
---------------- --------------------------------- -----  ---  ---  --------
VLAN0010         32778 (32768,  10) 1833.9d7b.1380    2    20   15  ieee
```

Note that both the commands in Example O-2 have a VLAN option: **show spanning-tree** [**vlan** *x*] **root** and **show spanning-tree** [**vlan** *x*] **bridge**. Without the VLAN listed, each command lists one line per VLAN; with the VLAN, the output lists the same information, but just for that one VLAN.

Configuring STP Port Costs

Changing the STP port costs requires a simple interface subcommand: **spanning-tree [vlan** *x*] **cost** *x*. To show how it works, consider the following example, which changes what happens in the network shown in Figure O-4.

Back in Figure O-4, with default settings, SW1 became root, and SW3 blocked on its G0/2 interface. A brief scan of the figure, based on the default STP cost of 4 for Gigabit interfaces, shows that SW3 should have found a cost 4 path and a cost 8 path to reach the root, as shown in Figure O-5.

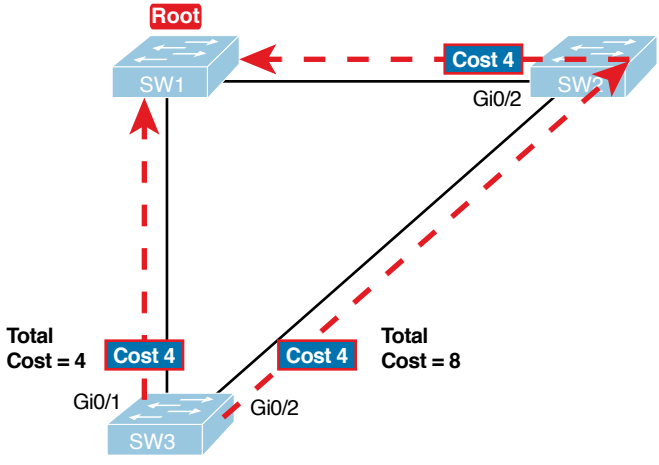

Figure O-5 *Analysis of SW3's Current Root Cost of 4 with Defaults*

To show the effects of changing the port cost, the next example shows a change to SW3's configuration, setting its G0/1 port cost higher so that the better path to the root goes out SW3's G0/2 port instead. Example O-3 also shows several other interesting effects.

Example O-3 *Manipulating STP Port Cost and Watching the Transition to Forwarding State*

```
SW3# debug spanning-tree events
Spanning Tree event debugging is on
SW3# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)# interface gigabitethernet0/1
SW3(config-if)# spanning-tree vlan 10 cost 30
SW3(config-if)# ^Z
SW3#
*Mar 11 06:28:00.860: STP: VLAN0010 new root port Gi0/2, cost 8
*Mar 11 06:28:00.860: STP: VLAN0010 Gi0/2 -> listening
*Mar 11 06:28:00.860: STP: VLAN0010 sent Topology Change Notice on Gi0/2
*Mar 11 06:28:00.860: STP[10]: Generating TC trap for port GigabitEthernet0/1
*Mar 11 06:28:00.860: STP: VLAN0010 Gi0/1 -> blocking
*Mar 11 06:28:15.867: STP: VLAN0010 Gi0/2 -> learning
*Mar 11 06:28:30.874: STP[10]: Generating TC trap for port GigabitEthernet0/2
*Mar 11 06:28:30.874: STP: VLAN0010 sent Topology Change Notice on Gi0/2
*Mar 11 06:28:30.874: STP: VLAN0010 Gi0/2 -> forwarding
```

This example starts with the **debug spanning-tree events** command on SW3. This command tells the switch to issue debug log messages whenever STP performs changes to an interface's role or state. These messages show up in the example as a result of the configuration.

Next, the example shows the configuration to change SW3's port cost, in VLAN 10, to 30, with the **spanning-tree vlan 10 cost 30** interface subcommand. Based on the figure, the root cost through SW3's G0/1 will now be 30 instead of 4. As a result, SW3's best cost to reach the root is cost 8, with SW3's G0/2 as its root port.

The debug messages tell us what STP on SW3 is thinking behind the scenes, with time-stamps. Note that the first five debug messages, displayed immediately after the user exited configuration mode in this case, all happen at the same time (down to the same millisec-ond). Notably, G0/1, which had been forwarding, immediately moves to a blocking state. Interface G0/2, which had been blocking, does not go to a forwarding state, instead moving to a listening state (at least, according to this message).

Now look for the debug message that lists G0/2 transitioning to learning state, and then the next one that shows it finally reaching forwarding state. How long between the messages? In each case, the message's timestamps show that 15 seconds passed. In this experiment, the switches used a default setting of forward delay (15 seconds). So, these debug messages confirm the steps that STP takes to transition an interface from blocking to forwarding state.

If you did not happen to enable a debug when configuring the cost, using **show** commands later can confirm the same choice by SW3, to now use its G0/2 port as its RP. Example O-4 shows the new STP port cost setting on SW3, along with the new root port and root cost, using the **show spanning-tree vlan 10** command. Note that G0/2 is now listed as the root port. The top of the output lists SW3's root cost as 8, matching the analysis shown in Figure O-5.

Example O-4 *New STP Status and Settings on SW3*

```
SW3# show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID     Priority    32778
              Address     1833.9d7b.0e80
              Cost        8
              Port        26 (GigabitEthernet0/2)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     f47f.35cb.d780
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Fa0/23              Desg FWD 19        128.23   P2p
Gi0/1               Altn BLK 30        128.25   P2p
Gi0/2               Root FWD 4         128.26   P2p
```

Configuring Priority to Influence the Root Election

The other big STP configuration option is to influence the root election by changing the priority of a switch. The priority can be set explicitly with the **spanning-tree vlan** *vlan-id* **priority** *value* global configuration command, which sets the base priority of the switch. (This is the command that requires a parameter of a multiple of 4096.)

However, Cisco gives us a better configuration option than configuring a specific priority value. In most designs, the network engineers pick two switches to be root: one to be root if all switches are up, and another to take over if the first switch fails. Switch IOS supports this idea with the **spanning-tree vlan** *vlan-id* **root primary** and **spanning-tree vlan** *vlan-id* **root secondary** commands.

The **spanning-tree vlan** *vlan-id* **root primary** command tells the switch to set its priority low enough to become root right now. The switch looks at the current root in that VLAN, and at the root's priority. Then the local switch chooses a priority value that causes the local switch to take over as root.

Remembering that Cisco switches use a default base priority of 32,768, this command chooses the base priority as follows:

Key Topic

- If the current root has a base priority higher than 24,576, the local switch uses a base priority of 24,576.
- If the current root's base priority is 24,576 or lower, the local switch sets its base priority to the highest multiple of 4096 that still results in the local switch becoming root.

O

For the switch intended to take over as the root if the first switch fails, use the **spanning-tree vlan** *vlan-id* **root secondary** command. This command is much like the **spanning-tree vlan** *vlan-id* **root primary** command, but with a priority value worse than the primary switch but better than all the other switches. This command sets the switch's base priority to 28,672 regardless of the current root's current priority value.

For example, in Figures O-4 and O-5, SW1 was the root switch, and as shown in various commands, all three switches defaulted to use a base priority of 32,768. Example O-5 shows a configuration that makes SW2 the primary root, and SW1 the secondary, just to show the role move from one to the other. These commands result in SW2 having a base priority of 24,576, and SW1 having a base priority of 28,672.

Example O-5 *Making SW2 Become Root Primary, and SW1 Root Secondary*

```
! First, on SW2:
SW2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW2(config)# spanning-tree vlan 10 root primary
SW2(config)# ^Z
```

```
! Next, SW1 is configured to back-up SW1
SW1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)# spanning-tree vlan 10 root secondary
SW1(config)# ^Z
SW1#
```

```
! The next command shows the local switch's BID (SW1)
SW1# show spanning-tree vlan 10 bridge

                                                Hello  Max  Fwd
Vlan                         Bridge ID           Time  Age  Dly  Protocol
---------------- --------------------------------- -----  ---  ---  --------
VLAN0010         28682 (28672,  10) 1833.9d7b.0e80    2    20   15  ieee

! The next command shows the root's BID (SW2)
SW1# show spanning-tree vlan 10 root

                                       Root    Hello Max Fwd
Vlan                 Root ID           Cost    Time  Age Dly  Root Port
---------------- ------------------- --------- ----- --- ---  ------------
VLAN0010         24586 1833.9d7b.1380         4    2   20  15  Gi0/1
```

The output of the two **show** commands clearly points out the resulting priority values on each switch. First, the **show spanning-tree bridge** command lists the local switch's BID information, while the **show spanning-tree root** command lists the root's BID, plus the local switch's root cost and root port (assuming it is not the root switch). So, SW1 lists its own BID, with priority 28,682 (base 28,672, with VLAN 10) with the **show spanning-tree bridge** command. Still on SW1, the output lists the root's priority as 24,586 in VLAN 10, implied as base 24,576 plus 10 for VLAN 10, with the **show spanning-tree root** command.

Note that alternatively you could have configured the priority settings specifically. SW1 could have used the **spanning-tree vlan 10 priority 28672** command, with SW2 using the **spanning-tree vlan 10 priority 24576** command. In this particular case, both options would result in the same STP operation.

Implementing Optional STP Features

This just-completed first major section of the chapter showed examples that used PVST+ only, assuming a default global command of **spanning-tree mode pvst**. At the same time, all the configuration commands shown in that first section, commands that influence STP operation, would influence both traditional STP and RSTP operation.

This section, the second of three major sections in this chapter, now moves on to discuss some useful but optional features that make both STP and RSTP work even better.

Configuring PortFast and BPDU Guard

You can easily configure the PortFast and BPDU Guard features on any interface, but with two different configuration options. One option works best when you want to enable these features only on a few ports, and the other works best when you want to enable these features on most every access port.

First, to enable the features on just one port at a time, use the **spanning-tree portfast** and the **spanning-tree bpduguard enable** interface subcommands. Example O-6 shows an

example of the process, with SW3's F0/4 interface enabling both features. (Also, note the long warning message IOS lists when enabling PortFast; using PortFast on a port connected to other switches can indeed cause serious problems.)

Example O-6 *Enabling PortFast and BPDU Guard on One Interface*

```
SW3# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)# interface fastEthernet 0/4
SW3(config-if)# spanning-tree portfast
%Warning: portfast should only be enabled on ports connected to a single
 host. Connecting hubs, concentrators, switches, bridges, etc... to this
 interface  when portfast is enabled, can cause temporary bridging loops.
 Use with CAUTION

%Portfast has been configured on FastEthernet0/4 but will only
 have effect when the interface is in a non-trunking mode.

SW3(config-if)# spanning-tree bpduguard ?
  disable  Disable BPDU guard for this interface
  enable   Enable BPDU guard for this interface

SW3(config-if)# spanning-tree bpduguard enable
SW3(config-if)# ^Z
SW3#
```

Example O-7 shows some brief information about the interface configuration of both PortFast and BPDU Guard. Of course, the **show running-config** command (not shown) would confirm the configuration commands from Example O-6. The **show spanning-tree interface fastethernet0/4 portfast** command in Example O-7 lists the PortFast status of the interface; note that the status value of *enabled* is displayed only if PortFast is configured and the interface is up. The **show spanning-tree interface detail** command then shows a line near the end of the output that states that PortFast and BPDU Guard are enabled. Note that this command would not list those two highlighted lines of output if these two features were not enabled.

Example O-7 *Verifying PortFast and BPDU Guard Configuration*

```
SW3# show spanning-tree interface fastethernet0/4 portfast
VLAN0104            enabled

SW11# show spanning-tree interface F0/4 detail
 Port 4 (FastEthernet0/4) of VLAN0001 is designated forwarding
   Port path cost 19, Port priority 128, Port Identifier 128.4.
   Designated root has priority 32769, address bcc4.938b.a180
   Designated bridge has priority 32769, address bcc4.938b.e500
   Designated port id is 128.4, designated path cost 19
   Timers: message age 0, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
```

```
  The port is in the portfast mode
  Link type is point-to-point by default
  Bpdu guard is enabled
  BPDU: sent 1721, received 0
```

PortFast and BPDU Guard are disabled by default on all interfaces, and to use them, each interface requires interface subcommands like those in Example O-6. Alternately, for both features, you can enable the feature globally. Then, for interfaces for which the feature should be disabled, you can use another interface subcommand to disable the feature.

The ability to change the global default for these features reduces the number of interface subcommands required. For instance, on an access layer switch with 48 access ports and two uplinks, 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 the features globally, and then disable them on the uplink ports.

Table O-2 summarizes the commands to enable and disable both PortFast and BPDU Guard, both globally and per interface. For instance, the global command **spanning-tree portfast default** changes the default so that all interfaces use PortFast, unless a port also has the **spanning-tree portfast disable** interface subcommand configured.

Table O-2 Enabling and Disabling PortFast and BPDU Guard, Globally and Per Interface

Action	Globally	One Interface
Disable PortFast	**no spanning-tree portfast default**	**spanning-tree portfast disable**
Enable PortFast	**spanning-tree portfast default**	**spanning-tree portfast**
Disable BPDU Guard	**no spanning-tree portfast bpduguard default**	**spanning-tree bpduguard disable**
Enable BPDU Guard	**spanning-tree portfast bpduguard default**	**spanning-tree bpduguard enable**

Example O-8 shows another new command, **show spanning-tree summary**. This command shows the current global settings for several STP parameters, including the PortFast and BPDU Guard features. This output was gathered on a switch that had enabled both PortFast and BPDU Guard globally.

Example O-8 *Displaying Status of Global Settings for PortFast and BPDU Guard*

```
SW1# show spanning-tree summary
Switch is in pvst mode
Root bridge for: none
EtherChannel misconfig guard is enabled
Extended system ID           is enabled
Portfast Default             is enabled
PortFast BPDU Guard Default  is enabled
Portfast BPDU Filter Default is disabled
Loopguard Default            is disabled
UplinkFast                   is disabled
BackboneFast                 is disabled
```

```
Configured Pathcost method used is short

Name                   Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
VLAN0001                      3         0        0          2          5
---------------------- -------- --------- -------- ---------- ----------
1 vlan                        3         0        0          2          5
```

Configuring EtherChannel

Two neighboring switches can treat multiple parallel links between each other as a single logical link called an *EtherChannel*. STP operates on the EtherChannel, instead of the individual physical links, so that STP either forwards or blocks on the entire logical EtherChannel for a given VLAN. As a result, a switch in a forwarding state can then load balance traffic over all the physical links in the EtherChannel. Without EtherChannel, only one of the parallel links between two switches would be allowed to forward traffic, with the rest of the links blocked by STP.

NOTE All references to EtherChannel in this Chapter refer to Layer 2 EtherChannels, and not to Layer 3 EtherChannels.

EtherChannel may 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 focuses on the correct EtherChannel configuration.

Configuring a Manual EtherChannel

The simplest way to configure an EtherChannel is to add the correct **channel-group** configuration command to each physical interface, on each switch, all with the **on** keyword. The **on** keyword tells the switches to place a physical interface into an EtherChannel.

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 the term "PortChannel." So, pay close attention to these three terms in the example.

To configure an EtherChannel manually, follow these steps:

Key Topic

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 O-9 shows a simple example, with two links between switches SW1 and SW2, as shown in Figure O-6. The configuration shows SW1's two interfaces placed into channel-group 1, with two **show** commands to follow.

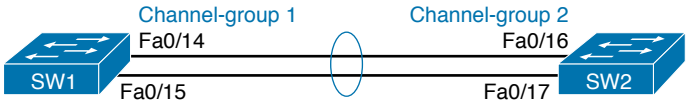

Figure O-6 *Sample LAN Used in EtherChannel Example*

Example O-9 *Configuring and Monitoring EtherChannel*

```
SW1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)# interface fa 0/14
SW1(config-if)# channel-group 1 mode on
SW1(config)# interface fa 0/15
SW1(config-if)# channel-group 1 mode on
SW1(config-if)# ^Z

SW1# show spanning-tree vlan 3

VLAN0003
  Spanning tree enabled protocol ieee
  Root ID     Priority    28675
              Address     0019.e859.5380
              Cost        12
              Port        72 (Port-channel1)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    28675  (priority 28672 sys-id-ext 3)
              Address     0019.e86a.6f80
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Po1              Root FWD 12        128.64   P2p Peer(STP)

SW1# show etherchannel 1 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


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

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)           -        Fa0/14(P)   Fa0/15(P)
```

Take a few moments to look at the output in the two **show** commands in the example, as well. First, the **show spanning-tree** command lists Po1, short for PortChannel1, as an interface. This interface exists because of the **channel-group** commands using the **1** parameter. STP no longer operates on physical interfaces F0/14 and F0/15, instead operating on the PortChannel1 interface, so only that interface is listed in the output.

Next, note the output of the **show etherchannel 1 summary** command. It lists as a heading "Port-channel," with Po1 below it. It also lists both F0/14 and F0/15 in the list of ports, with a (P) beside each. Per the legend, the *P* means that the ports are bundled in the port channel, which is a code that means these ports have passed all the configuration checks and are valid to be included in the channel.

NOTE Cisco uses the term *EtherChannel* to refer to the concepts discussed in this section. To refer to the item configured in the switch, Cisco instead uses the term *port channel*, with the command keyword **port-channel**. For the purposes of understanding the technology, you may treat these terms as synonyms. However, it helps to pay close attention to the use of the terms *port channel* and *EtherChannel* as you work through the examples in this section, because IOS uses both.

Configuring Dynamic EtherChannels

Cisco switches support two different protocols that allow the switches to negotiate whether a particular link becomes part of an EtherChannel or not. Basically, the configuration enables the 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.

Cisco switches support the Cisco-proprietary Port Aggregation Protocol (PAgP) and the IEEE standard Link Aggregation Control Protocol (LACP), based on IEEE standard 802.3ad. 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.

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 O-7, 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**.

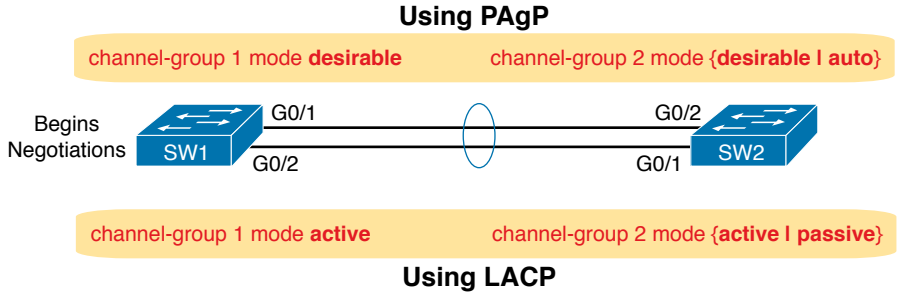

Figure O-7 *Correct EtherChannel Configuration Combinations*

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.

For example, in the design shown in Figure O-7, imagine both physical interfaces on both switches were configured with the **channel-group 2 mode desirable** interface subcommand. As a result, the two switches would negotiate and create an EtherChannel. Example O-10 shows the verification of that configuration, with the command **show etherchannel 2 port-channel**. This command confirms the protocol in use (PAgP, because the **desirable** keyword was configured), and the list of interfaces in the channel.

Example O-10 *EtherChannel Verification: PAgP Desirable Mode*

```
SW1# show etherchannel 2 port-channel
                Port-channels in the group:
                ---------------------------


Port-channel: Po2
------------

Age of the Port-channel   = 0d:00h:04m:04s
Logical slot/port   = 16/1          Number of ports = 2
GC                  = 0x00020001      HotStandBy port = null
Port state          = Port-channel Ag-Inuse
Protocol            =   PAgP
Port security       = Disabled
```

```
Ports in the Port-channel:

Index   Load   Port     EC state        No of bits
------+------+------+------------------+-----------
  0     00     Gi0/1    Desirable-Sl       0
  0     00     Gi0/2    Desirable-Sl       0

Time since last port bundled:    0d:00h:03m:57s    Gi0/2
```

Implementing RSTP

All you have to do to migrate from STP to RSTP is to configure the **spanning-tree mode rapid-pvst** global command on all the switches. However, for exam preparation, it helps to work through the various **show** commands, particularly to prepare for Simlet questions. Those questions can ask you to interpret **show** command output without allowing you to look at the configuration, and the output of **show** commands when using STP versus RSTP is very similar.

This third and final major section of this chapter focuses on pointing out the similarities and differences between STP and RSTP as seen in Catalyst switch configuration and verification commands. This section explains the configuration and verification of RSTP, with emphasis on how to identify RSTP features.

Identifying the STP Mode on a Catalyst Switch

Cisco Catalyst switches operate in some STP mode as defined by the **spanning-tree mode** global configuration command. Based on this command's setting, the switch is using either 802.1D STP or 802.1w RSTP, as noted in Table O-3.

Key Topic

Table O-3 Cisco Catalyst STP Configuration Modes

Parameter on spanning-tree mode Command	Uses STP or RSTP?	Protocol Listed in Command Output	Description
pvst	STP	ieee	Default; Per-VLAN Spanning Tree instance
rapid-pvst	RSTP	rstp	Like PVST, but uses RSTP rules instead of STP for each STP instance
mst	RSTP	mst	Creates multiple RSTP instances but does not require one instance per each VLAN

To determine whether a Cisco Catalyst switch uses RSTP, you can look for two types of information. First, you can look at the configuration, as noted in the left column of Table O-3. Also, some **show** commands list the STP protocol as a reference to the configuration of the **spanning-tree mode** global configuration command. A protocol of **rstp** or **mst** refers to one of the modes that uses RSTP, and a protocol of **ieee** refers to the mode that happens to use STP.

Before looking at an example of the output, review the topology in Figure O-8. The remaining RSTP examples in this chapter use this topology. In the RSTP examples in this chapter, SW1 will become root, and SW3 will block on one port (G0/2), as shown.

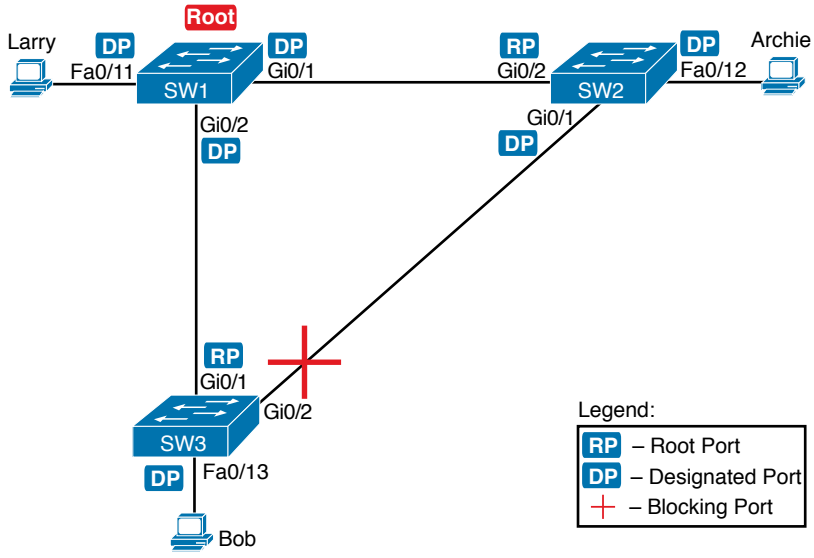

Figure O-8 *Network Topology for STP and RSTP Examples*

The first example focuses on VLAN 10, with all switches using 802.1D STP and the default setting of **spanning-tree mode pvst**. This setting creates an instance of STP per VLAN (which is the per-VLAN part of the name) and uses 802.1D STP. Each switch places the port connected to the PC into VLAN 10 and enables both PortFast and BPDU Guard. Example O-11 shows a sample configuration from switch SW3, with identical interface subcommands configured on SW1's F0/11 and SW2's F0/12 ports, respectively.

Example O-11 *Sample Configuration from Switch SW3*

```
SW3# show running-config interface Fastethernet 0/13

Building configuration...

Current configuration : 117 bytes
!
interface FastEthernet0/13
 switchport access vlan 10
 spanning-tree portfast
 spanning-tree bpduguard enable
end
```

At this point, the three switches use 802.1D STP because all use the default PVST mode. Example O-12 shows the evidence of STP's work, with only subtle and indirect clues that STP happens to be in use.

Example O-12 *Output That Confirms the Use of 802.1D STP on Switch SW3*

```
SW3# show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol ieee
  Root ID     Priority    32778
              Address     1833.9d7b.0e80
              Cost        4
              Port        25 (GigabitEthernet0/1)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     f47f.35cb.d780
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Fa0/13              Desg FWD 19        128.13   P2p Edge
Gi0/1               Root FWD 4         128.25   P2p
Gi0/2               Altn BLK 4         128.26   P2p


SW3# show spanning-tree vlan 10 bridge

                                                  Hello  Max  Fwd
Vlan                         Bridge ID             Time  Age  Dly  Protocol
---------------- --------------------------------- -----  ---  ---  --------
VLAN0010         32778 (32768,  10) f47f.35cb.d780    2    20   15  ieee
```

The highlighted parts of the example note the references to the STP protocol as ieee, which implies that STP is in use. The term *ieee* is a reference to the original IEEE 802.1D STP standard.

To migrate this small network to use RSTP, configure the **spanning-tree mode rapid-pvst** command. This continues the use of per-VLAN spanning-tree instances, but it applies RSTP logic to each STP instance. Example O-13 shows the output of the same two commands from Example O-12 after configuring the **spanning-tree mode rapid-pvst** command on all three switches.

Example O-13 *Output That Confirms the Use of 802.1w RSTP on Switch SW3*

```
SW3# show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol rstp
  Root ID     Priority    32778
              Address     1833.9d7b.0e80
```

```
              Cost        4
              Port        25 (GigabitEthernet0/1)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32778  (priority 32768 sys-id-ext 10)
              Address     f47f.35cb.d780
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Fa0/13              Desg FWD 19        128.13   P2p Edge
Gi0/1               Root FWD 4         128.25   P2p
Gi0/2               Altn BLK 4         128.26   P2p


SW3# show spanning-tree vlan 10 bridge

                                                  Hello  Max  Fwd
Vlan                         Bridge ID             Time  Age  Dly  Protocol
---------------- --------------------------------- -----  ---  ---  --------
VLAN0010         32778 (32768,  10) f47f.35cb.d780    2    20   15  rstp
```

Pay close attention to the differences between the 802.1D STP output in Example O-12 and the 802.1w RSTP output in Example O-13. Literally, the only difference is rstp instead of ieee in one place in the output of each of the two commands listed. In this case, rstp refers to the configuration of the **spanning-tree mode rapid-pvst** global config command, which implied the use of RSTP.

RSTP Port Roles

RSTP adds two port roles to STP: the alternate port and the backup port. Example O-14 repeats an excerpt from the **show spanning-tree vlan 10** command on switch SW3 to show an example of the alternate port role. SW3 (as shown earlier in Figure O-8) is not the root switch, with G0/1 as its root port and G0/2 as an alternate port.

Example O-14 *Output Confirming SW3's Root Port and Alternate Port Roles*

```
SW3# show spanning-tree vlan 10
! Lines omitted for brevity
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Fa0/13              Desg FWD 19        128.13   P2p Edge
Gi0/1               Root FWD 4         128.25   P2p
Gi0/2               Altn BLK 4         128.26   P2p
```

The good news is that the output clearly lists which port is the root port (Gi0/1) and which port is the alternate root port (Gi0/2). The only trick is to know that Altn is a shortened version of the word *alternate*.

Pay close attention to this short description of an oddity about the STP and RSTP output on Catalyst switches! Cisco Catalyst switches often show the alternate and backup ports in output even when using STP and not RSTP. The alternate and backup port concepts are RSTP concepts. The switches only converge faster using these concepts when using RSTP. But **show** command output, when using STP and not RSTP, happens to identify what would be the alternate and backup ports if RSTP were used.

Key Topic

Why might you care about such trivia? Seeing output that lists an RSTP alternate port does not confirm that the switch is using RSTP. So, do not make that assumption on the exam. To confirm that a switch uses RSTP, you must look at the configuration of the **spanning-tree mode** command, or look for the protocol as summarized back in Table O-3.

For instance, just compare the output of Example O-12 and Example O-14. Example O-12 shows output for this same SW3, with the same parameters, except that all switches used PVST mode, meaning all the switches used STP. Example O-12's output (based on STP) lists SW3's G0/2 as Altn, meaning alternate, even though the alternate port concept is not an STP concept, but an RSTP concept.

RSTP Port States

RSTP added one new port state compared to STP, discarding, using it as a replacement for the STP port states of disabled and blocking. You might think that after you configure a switch to use RSTP rather than STP, instead of seeing ports in a blocking state, you would now see the discarding state. However, the Cisco Catalyst switch output basically ignores the new term *discarding*, continuing to use the old term *blocking* instead.

For example, scan back to the most recent RSTP example (Example O-14), to the line for SW3's port G0/2. Then look for the column with heading STS, which refers to the status or state. The output shows G0/2 is listed as BLK, or blocking. In theory, because SW3 uses RSTP, the port state ought to be discarding, but the switch IOS continues to use the older notation of BLK for blocking.

Just as one more bit of evidence, the command **show spanning-tree vlan 10 interface gigabitethernet0/2 state** lists the STP or RSTP port state with the state fully spelled out. Example O-15 shows this command, taken from SW3, for interface G0/2. Note the fully spelled-out *blocking* term instead of the RSTP term *discarding*.

Example O-15 *SW3, an RSTP Switch, Continues to Use the Old Blocking Term*

```
SW3# show spanning-tree vlan 10 interface gigabitEthernet 0/2 state
VLAN0010            blocking
```

RSTP Port Types

Cisco Catalyst switches determine the RSTP port type based on two port settings: the current duplex (full or half) and whether the PortFast feature is enabled. First, full duplex tells the switch to use port type point-to-point, with half duplex telling the switch to use port type shared. Enabling PortFast tells the switch to treat the port as an edge port. Table O-4 summarizes the combinations.

Table O-4 RSTP Port Types

Type	Current Duplex Status	Is Spanning-Tree PortFast Configured?
Point-to-point	Full	No
Point-to-point edge	Full	Yes
Shared	Half	No
Shared edge[1]	Half	Yes

[1] Cisco recommends against using this combination, to avoid causing loops.

You can easily find the RSTP port types in the output of several commands, including the same **show spanning-tree** command in Example O-16. Example O-16 lists output from switch SW2, with a hub added off SW2's F0/18 port (not shown in Figure O-8). The hub was added so that the output in Example O-16 lists a shared port (noted as Shr) to go along with the point-to-point ports (noted as P2p).

Example O-16 *RSTP Port Types*

```
SW2# show spanning-tree vlan 10

VLAN0010
  Spanning tree enabled protocol rstp
  Root ID     Priority    32778
              Address     1833.9d7b.0e80
              Cost        4
              Port        26 (GigabitEthernet0/2)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32778  (priority 32768 sys-id-ext 10)
              Address     1833.9d7b.1380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Fa0/12              Desg FWD 19        128.12   P2p Edge
Fa0/18              Desg FWD 19        128.18   Shr
Gi0/1               Desg FWD 4         128.25   P2p
Gi0/2               Root FWD 4         128.26   P2p
```

For exam prep, again note an odd fact about the highlighted output in Example O-16: The port type details appear in the output when using both STP and RSTP. For example, refer to Example O-12 again, which shows output from SW3 when using STP (when configured for PVST mode). The Type column also identifies point-to-point and edge interfaces.

Command References

Tables O-5 and O-6 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 O-5 Appendix O 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.
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).
channel-group *channel-group-number* **mode** {**auto** | **desirable** | **active** | **passive** | **on**}	Interface subcommand that enables EtherChannel on the interface.
spanning-tree portfast	Interface subcommand that enables PortFast on the interface.
spanning-tree bpduguard enable	Interface subcommand that enables BPDU Guard on an interface.
spanning-tree portfast default	Global command that changes the switch default for PortFast on access interfaces from disabled to enabled.
spanning-tree portfast bpduguard default	Global command that changes the switch default for BPDU Guard on access interfaces from disabled to enabled.
no spanning-tree portfast default	Global command that changes the global setting for PortFast to disabled.
no spanning-tree portfast bpduguard default	Global command that changes the global setting for BPDU Guard to disabled.
spanning-tree portfast disable	Interface subcommand that disables PortFast on the interface.
spanning-tree bpduguard disable	Interface subcommand that disables BPDU Guard on an interface.

Table O-6 Appendix O 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 interface *interface-id*	Lists STP information only for the specified port.
show spanning-tree vlan *vlan-id*	Lists STP information for the specified VLAN.
show spanning-tree [**vlan** *vlan-id*] **root**	Lists information about each VLAN's root or for just the specified VLAN.
show spanning-tree [**vlan** *vlan-id*] **bridge**	Lists STP information about the local switch for each VLAN or for just the specified VLAN.
show spanning-tree summary	Lists global STP settings for a switch, including the default PortFast and BPDU Guard settings, and the VLANs for which this switch is the root switch.
debug spanning-tree events	Causes the switch to provide informational messages about changes in the STP topology.
show spanning-tree interface *type number* **portfast**	Lists a one-line status message about PortFast on the listed interface.
show etherchannel [*channel-group-number*] {**brief** | **detail** | **port** | **port-channel** | **summary**}	Lists information about the state of EtherChannels on this switch.

APPENDIX P

LAN Troubleshooting

NOTE This appendix contains an entire chapter that was published as a chapter in one of the past editions of this book or a related book. The author includes this appendix with the current edition as extra reading for anyone interested in learning more. However, note that the content in this appendix has not been edited since it was published in the earlier edition, so references to exams and exam topics, and to other chapters, will be outdated. This appendix was previously published as Chapter 4 of the book *CCNA Routing and Switching ICND2 200-105 Official Cert Guide*, published in 2016.

This chapter discusses the LAN topics discussed in depth in the first three chapters, plus a few prerequisite topics, from a troubleshooting perspective.

Troubleshooting for any networking topic requires a slightly different mindset as compared to thinking about configuration and verification. When thinking about configuration and verification, it helps to think about basic designs, learn how to configure the feature correctly, and learn how to verify the correct configuration is indeed working correctly. However, to learn how to troubleshoot, you need to think about symptoms when the design is incorrect, or if the configuration does not match the good design. What symptoms occur when you make one type of mistake or another? This chapter looks at the common types of mistakes, and works through how to look at the status with **show** commands to find those mistakes.

This chapter breaks the material into four major sections. The first section tackles the largest topic, STP troubleshooting. STP is not likely to fail as a protocol; instead, STP may not be operating as designed, so the task is to find how STP is currently working and discover how to then make the configuration implement the correct design. The second major section then moves on to Layer 2 EtherChannels, which have a variety of small potential problems that can prevent the dynamic formation of an EtherChannel.

The third major section of the chapter focuses on the data plane forwarding of Ethernet frames on LAN switches, in light of VLANs, trunks, STP, and EtherChannels. That same section reviews the Layer 2 forwarding logic of a switch in light of these features. The fourth and final major section then examines VLAN and trunking issues, and how those issues impact switch forwarding.

Note that a few of the subtopics listed within the exam topics at the beginning of this chapter are not discussed in this chapter. This chapter does not discuss VTP beyond its basic features or Layer 3 EtherChannels.

Foundation Topics

Troubleshooting STP

STP questions tend to intimidate many test takers. STP uses many rules, with tiebreakers in case one rule ends with a tie. Without much experience with STP, people tend to distrust their own answers. Also, even those of us with networking jobs already probably do not troubleshoot STP very often, because STP works well. Often, troubleshooting STP is not about STP failing to do its job but rather about STP working differently than designed, with a different root switch, or different root ports (RP), and so on. Seldom does STP troubleshooting begin with a case in which STP has failed to prevent a loop.

This section reviews the rules for STP, while emphasizing some important troubleshooting points. In particular, this section takes a closer look at the tiebreakers that STP uses to make decisions. It also makes some practical suggestions about how to go about answering exam questions such as "which switch is the root switch?"

Determining the Root Switch

Determining the STP root switch is easy if you know all the switches' BIDs: Just pick the lowest value. If the question lists the priority and MAC address separately, as is common in some **show** command output, pick the switch with the lowest priority, or in the case of a tie, pick the lower MAC address value.

And just to be extra clear, STP does not have nor need a tiebreaker for electing the root switch. The BID uses a switch universal MAC address as the last 48 bits of the BID. These MAC addresses are unique in the universe, so there should never be identical BIDs or the need for a tiebreaker.

For the exam, a question that asks about the root switch might not be so simple as listing a bunch of BIDs and asking you which one is "best." A more likely question is a simulator (sim) question in which you have to do any **show** commands you like or a multiple choice question that lists the output from only one or two commands. Then you have to apply the STP algorithm to figure out the rest.

When faced with an exam question using a simulator, or just the output in an exhibit, use a simple strategy of ruling out switches, as follows:

Step 1. Begin with a list or diagram of switches, and consider all as possible root switches.

Step 2. Rule out any switches that have an RP (**show spanning-tree**, **show spanning-tree root**), because root switches do not have an RP.

Step 3. Always try **show spanning-tree**, because it identifies the local switch as root directly: "This switch is the root" on the fifth line of output.

Step 4. Always try **show spanning-tree root**, because it identifies the local switch as root indirectly: The RP column is empty if the local switch is the root.

Step 5. When using a sim, rather than try switches randomly, chase the RPs. For example, if starting with SW1, and SW1's G0/1 is an RP, next try the switch on the other end of SW1's G0/1 port.

Step 6. When using a sim, use **show spanning-tree vlan** *x* on a few switches and record the root switch, RP, and designated port (DP). This strategy can quickly show you most STP facts.

The one step in this list that most people ignore is the idea of ruling out switches that have an RP. Root switches do not have an RP, so any switch with an RP can be ruled out as not being the root switch for that VLAN. Example P-1 shows two commands on switch SW2 in some LAN that confirms that SW2 has an RP and is therefore not the root switch.

Example P-1 *Ruling Out Switches as Root Based on Having a Root Port*

```
SW2# show spanning-tree vlan 20 root

                                        Root Hello Max Fwd
Vlan                Root ID             Cost  Time Age Dly  Root Port
---------------- ------------------- --------- ----- --- ---  ------------
VLAN0020         32788 1833.9d7b.0e80          4    2   20  15  Gi0/2

SW2# show spanning-tree vlan 20

VLAN0020
  Spanning tree enabled protocol ieee
  Root ID     Priority    32788
              Address     1833.9d7b.0e80
              Cost        4
              Port        26 (GigabitEthernet0/2)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32788  (priority 32768 sys-id-ext 20)
              Address     1833.9d7b.1380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  15  sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Gi0/1               Desg FWD 4         128.25   P2p
Gi0/2               Root FWD 4         128.26   P2p
```

Both commands identify SW2's G0/2 port as its RP, so if you follow the suggestions, the next switch to try in a sim question would be the switch on the other end of SW2's G0/2 interface.

Determining the Root Port on Nonroot Switches

Determining the RP of a switch when **show** command output is available is relatively easy. As shown recently in Example P-1, both **show spanning-tree** and **show spanning-tree root** list the root port of the local switch, assuming it is not the root switch. The challenge comes more when an exam question makes you think through how the switches choose the RP based on the root cost of each path to the root switch, with some tiebreakers as necessary.

As a review, each nonroot switch has one, and only one, RP for a VLAN. To choose its RP, a switch listens for incoming Hello bridge protocol data units (BPDU). For each received Hello, the switch adds the cost listed in the hello BPDU to the cost of the incoming interface (the interface on which the Hello was received). That total is the root cost over that path. The lowest root cost wins, and the local switch uses its local port that is part of the least root cost path as its root port.

Most humans can analyze what STP chooses by using a network diagram and a slightly different algorithm. Instead of thinking about Hello messages and so on, approach the question as this: the sum of all outgoing port costs between the nonroot switch and the root. Repeating a familiar example, with a twist, Figure P-1 shows the calculation of the root cost. Note that SW3's Gi0/1 port has yet again had its cost configured to a different value.

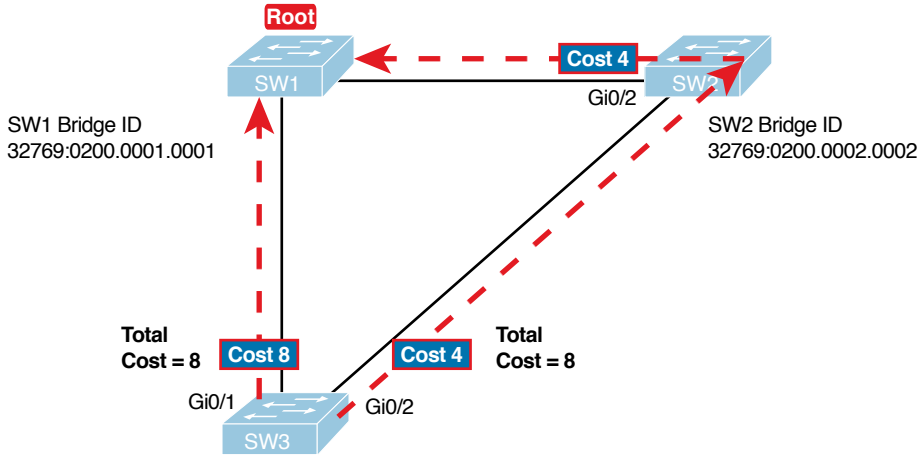

Figure P-1 *SW3's Root Cost Calculation Ends in a Tie*

STP Tiebreakers When Choosing the Root Port

Figure P-1 shows the easier process of adding the STP costs of the outgoing interfaces over each from SW3, a nonroot, to SW1, the root. It also shows a tie (on purpose), to talk about the tiebreakers.

When a switch chooses its root port, the first choice is to choose the local port that is part of the least root cost path. When those costs tie, the switch picks the port connected to the neighbor with the lowest BID. This tiebreaker usually breaks the tie, but not always. So, for completeness, the three tiebreakers are, in the order a switch uses them, 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.

(Note that the switch only considers the root paths that tie when thinking about these tiebreakers.)

For example, Figure P-1 shows that SW3 is not root and that its two paths to reach the root tie with their root costs of 8. The first tiebreaker is the lowest neighbor's BID. SW1's BID value is lower than SW2's, so SW3 chooses its G0/1 interface as its RP in this case.

The last two RP tiebreakers come into play only when two switches connect to each other with multiple links, as shown in Figure P-2. In that case, a switch receives Hellos on more than one port from the same neighboring switch, so the BIDs tie.

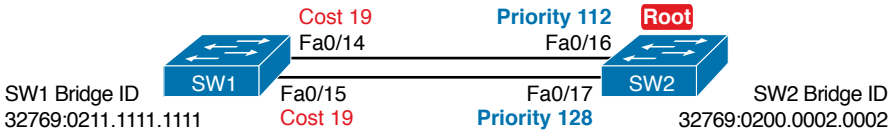

Figure P-2 *Topology Required for the Last Two Tiebreakers for Root Port*

In this particular example, SW2 becomes root, and SW1 needs to choose its RP. SW1's port costs tie, at 19 each, so SW1's root cost over each path will tie at 19. SW2 sends Hellos over each link to SW1, so SW1 cannot break the tie based on SW1's neighbor BID because both list SW2's BID. So, SW1 has to turn to the other two tiebreakers.

NOTE In real life, most engineers would put these two links into an EtherChannel.

The next tiebreaker is a configurable option: the neighboring switch's port priority on each neighboring switch interface. Cisco switch ports default to a setting of 128, with a range of values from 0 through 255, with lower being better (as usual). In this example, the network engineer has set SW2's F0/16 interface with the **spanning-tree vlan 10 port-priority 112** command. SW1 learns that the neighbor has a port priority of 112 on the top link and 128 on the bottom, so SW1 uses its top (F0/14) interface as the root port.

If the port priority ties, which it often does due to the default values, STP relies on an internal port numbering on the neighbor. Cisco switches assign an internal integer to identify each interface on the switch. The nonroot looks for the neighbor's lowest internal port number (as listed in the Hello messages) and chooses its RP based on the lower number.

Cisco switches use an obvious numbering, with Fa0/1 having the lowest number, then Fa0/2, then Fa0/3, and so on. So, in Figure P-2, SW2's Fa0/16 would have a lower internal port number than Fa0/17; SW1 would learn those numbers in the Hello; and SW1 would use its Fa0/14 port as its RP.

Suggestions for Attacking Root Port Problems on the Exam

Exam questions that make you think about the RP can be easy if you know where to look and the output of a few key commands is available. However, the more conceptual the question, the more you have to calculate the root cost over each path, correlate that to different **show** commands, and put the ideas together. The following list makes a few suggestions about how to approach STP problems on the exam:

Key Topic

1. If available, look at the **show spanning-tree** and **show spanning-tree root** commands. Both commands list the root port and the root cost (see Example P-1).
2. The **show spanning-tree** command lists cost in two places: the root cost at the top, in the section about the root switch; and the interface cost, at the bottom, in the per-interface section. Be careful, though; the cost at the bottom is the interface cost, not the root cost!

P

3. For problems where you have to calculate a switch's root cost:
   a. Memorize the default cost values: 100 for 10 Mbps, 19 for 100 Mbps, 4 for 1 Gbps, and 2 for 10 Gbps.
   b. Look for any evidence of the **spanning-tree cost** configuration command on an interface, because it overrides the default cost. Do not assume default costs are used.
   c. When you know a default cost is used, if you can, check the current actual speed as well. Cisco switches choose STP cost defaults based on the current speed, not the maximum speed.

Determining the Designated Port on Each LAN Segment

Each LAN segment has a single switch that acts as the designated port (DP) on that segment. On segments that connect a switch to a device that does not even use STP—for example, segments connecting a switch to a PC or a router—the switch always wins, because it is the only device sending a Hello onto the link. However, links with two switches require a little more work to discover which should be the DP. By definition:

Key Topic

Step 1. For switches connected to the same LAN segment, the switch with the lowest cost to reach the root, as advertised in the Hello they send onto the link, becomes the DP on that link.

Step 2. In case of a tie, among the switches that tied on cost, the switch with the lowest BID becomes the DP.

For example, consider Figure P-3. This figure notes the root, RPs, and DPs and each switch's least cost to reach the root over its respective RP.

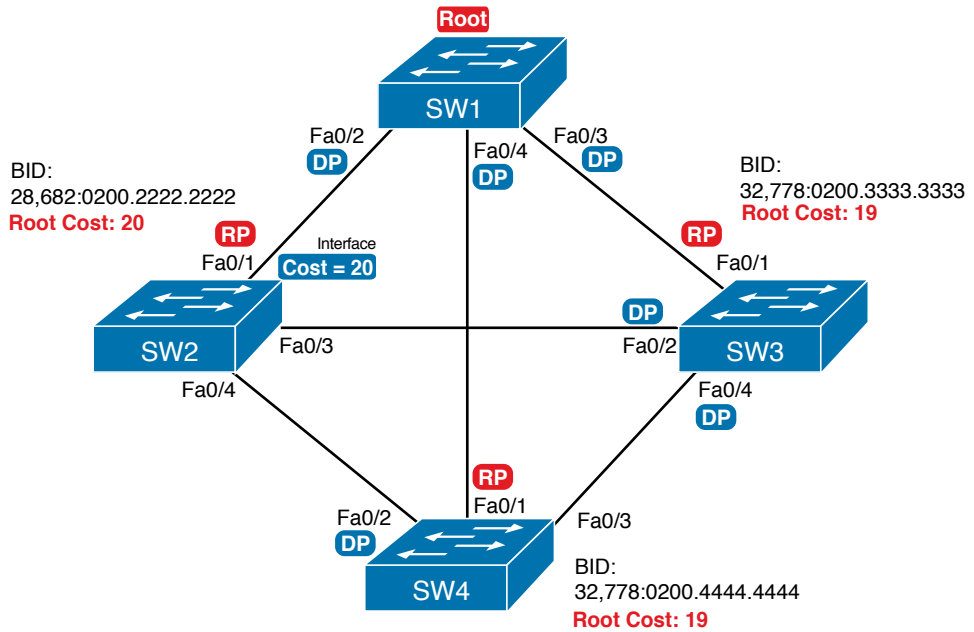

Figure P-3 *Picking the DPs*

Focus on the segments that connect the nonroot switches for a moment:

SW2–SW4 segment: SW4 wins because of its root cost of 19, compared to SW2's root cost of 20.

SW2–SW3 segment: SW3 wins because of its root cost of 19, compared to SW2's root cost of 20.

SW3–SW4 segment: SW3 and SW4 tie on root cost, both with root cost 19. SW3 wins due to its better (lower) BID value.

Interestingly, SW2 loses and does not become DP on the links to SW3 and SW4 even though SW2 has the better (lower) BID value. The DP tiebreaker does use the lowest BID, but the first DP criteria is the lowest root cost, and SW2's root cost happens to be higher than SW3's and SW4's.

NOTE A single switch can connect two or more interfaces to the same collision domain, and compete to become DP, if hubs are used. In such cases, two different switch ports on the same switch tie, the DP choice uses the same two final tiebreakers as used with the RP selection: the lowest interface STP priority, and if that ties, the lowest internal interface number.

Suggestions for Attacking Designated Port Problems on the Exam

As with exam questions asking about the RP, exam questions that make you think about the DP can be easy if you know where to look and the output of a few key commands is available. However, the more conceptual the question, the more you have to think about the criteria for choosing the DP: first the root cost of the competing switches, and then the better BID if they tie based on root cost.

The following list gives some tips to keep in mind when digging into a given DP issue. Some of this list repeats the suggestions for finding the RP, but to be complete, this list includes each idea as well.

Key Topic

1. If available, look at the **show spanning-tree** commands, at the list of interfaces at the end of the output. Then, look for the Role column, and look for Desg, to identify any DPs.
2. Identify the root cost of a switch directly by using the **show spanning-tree** command. But be careful! This command lists the cost in two places, and only the mention at the top, in the section about the root, lists the root cost.
3. For problems where you have to calculate a switch's root cost, do the following:
   a. Memorize the default cost values: 100 for 10 Mbps, 19 for 100 Mbps, 4 for 1 Gbps, and 2 for 10 Gbps.
   b. Look for any evidence of the **spanning-tree cost** configuration command on an interface, because it overrides the default cost. Do not assume default costs are used.
   c. When you know a default cost is used, if you can, check the current actual speed as well. Cisco switches choose STP cost defaults based on the current speed, not the maximum speed.

STP Convergence

STP puts each RP and DP into a forwarding state, and ports that are neither RP nor DP into a blocking state. Those states may remain as is for days, weeks, or months. But at some point, some switch or link will fail, a link may change speeds (changing the STP cost), or the STP configuration may change. Any of these events can cause switches to repeat their STP algorithm, which may in turn change their own RP and any ports that are DPs.

When STP converges based on some change, not all the ports have to change their state. For instance, a port that was forwarding, if it still needs to forward, just keeps on forwarding. Ports that were blocking that still need to block keep on blocking. But when a port needs to change state, something has to happen, based on the following rules:

Key Topic

- For interfaces that stay in the same STP state, nothing needs to change.
- For interfaces that need to move from a forwarding state to a blocking state, the switch immediately changes the state to blocking.
- For interfaces that need to move from a blocking state to a forwarding state, the switch first moves the interface to listening state, then learning state, each for the time specified by the forward delay timer (default 15 seconds). Only then is the interface placed into forwarding state.

Because the transition from blocking to forwarding does require some extra steps, you should be ready to respond to conceptual questions about the transition.

Troubleshooting Layer 2 EtherChannel

EtherChannels can prove particularly challenging to troubleshoot for a couple of reasons. First, you have to be careful to match the correct configuration, and there are many more incorrect configuration combinations than there are correct combinations. Second, many interface settings must match on the physical links, both on the local switch and on the neighboring switch, before a switch will add the physical link to the channel. This second major section in the chapter works through both sets of issues.

Incorrect Options on the channel-group Command

The rules for the small set of working configuration options on the **channel-group** command can be summarized as follows, for a single EtherChannel:

Key Topic

1. On the local switch, all the **channel-group** commands for all the physical interfaces must use the same channel-group number.
2. The channel-group number can be different on the neighboring switches.
3. If using the **on** keyword, you must use it on the corresponding interfaces of both switches.
4. If you use the **desirable** keyword on one switch, the switch uses PAgP; the other switch must use either **desirable** or **auto**.
5. If you use the **active** keyword on one switch, the switch uses LACP; the other switch must use either **active** or **passive**.

These rules summarize the correct configuration options, but the options actually leave many more incorrect choices. The following list shows some incorrect configurations that the switches allow, even though they would result in the EtherChannel not working. The list

compares the configuration on one switch to another based on the physical interface configuration. Each lists the reasons why the configuration is incorrect.

- Configuring the **on** keyword on one switch, and **desirable**, **auto**, **active**, or **passive** on the other switch. The **on** keyword does not enable PAgP, and does not enable LACP, and the other options rely on PAgP or LACP.
- Configuring the **auto** keyword on both switches. Both use PAgP, but both wait on the other switch to begin negotiations.
- Configuring the **passive** keyword on both switches. Both use LACP, but both wait on the other switch to begin negotiations.
- Configuring the **active** keyword on one switch and either **desirable** or **auto** on the other switch. The **active** keyword uses LACP, whereas the other keywords use PAgP.
- Configuring the **desirable** keyword on one switch and either **active** or **passive** on the other switch. The **desirable** keyword uses PAgP, whereas the other keywords use LACP.

Example P-2 shows an example that matches the last item in the list. In this case, SW1's two ports (F0/14 and F0/15) have been configured with the **desirable** keyword, and SW2's matching F0/16 and F0/17 have been configured with the **active** keyword. The example lists some telling status information about the failure, with notes following the example.

Example P-2 *Incorrect Configuration Using Mismatched PortChannel Protocols*

```
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


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

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------

1      Po1(SD)          PAgP      Fa0/14(I)    Fa0/15(I)

SW1# show interfaces status | include Po|14|15
Port      Name               Status       Vlan       Duplex  Speed Type
Fa0/14                       connected    301        a-full  a-100 10/100BaseTX
Fa0/15                       connected    301        a-full  a-100 10/100BaseTX
Po1                          notconnect   unassigned   auto   auto
```

P

Start at the top, in the legend of the **show etherchannel summary** command. The *D* code letter means that the channel itself is down, with *S* meaning that the channel is a Layer 2 EtherChannel. Code *I* means that the physical interface is working independently from the PortChannel (described as "stand-alone"). Then, the bottom of that command's output highlights PortChannel 1 (Po1) as Layer 2 EtherChannel in a down state (SD), with F0/14 and F0/15 as stand-alone interfaces (I).

Interestingly, because the problem is a configuration mistake, the two physical interfaces still operate independently, as if the PortChannel did not exist. The last command in the example shows that while the PortChannel 1 interface is down, the two physical interfaces are in a connected state.

NOTE As a suggestion for attacking EtherChannel problems on the exam, rather than memorizing all the incorrect configuration options, concentrate on the list of correct configuration options. Then look for any differences between a given question's configuration as compared to the known correct configurations and work from there.

Configuration Checks Before Adding Interfaces to EtherChannels

Even when the **channel-group** commands have all been configured correctly, other configuration settings can cause problems as well. This last topic examines those configuration settings and their impact.

First, a local switch checks each new physical interface that is configured to be part of an EtherChannel, comparing each new link to the existing links. That new physical interface's settings must be the same as the existing links' 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
- STP interface settings

In addition, switches check the settings on the neighboring switch. To do so, the switches either use PAgP or LACP (if already in use), or use Cisco Discovery Protocol (CDP) if using manual configuration. The neighbor must match on all parameters in this list except the STP settings.

As an example, SW1 and SW2 again use two links in one EtherChannel. Before configuring the EtherChannel, SW1's F0/15 was given a different STP port cost than F0/14. Example P-3 picks up the story just after configuring the correct **channel-group** commands, when the switch is deciding whether to use F0/14 and F0/15 in this EtherChannel.

Example P-3 *Local Interfaces Fail in EtherChannel Because of Mismatched STP Cost*

```
*Mar  1 23:18:56.132: %PM-P-ERR_DISABLE: channel-misconfig (STP) error detected on
  Po1, putting Fa0/14 in err-disable state
*Mar  1 23:18:56.132: %PM-P-ERR_DISABLE: channel-misconfig (STP) error detected on
  Po1, putting Fa0/15 in err-disable state
*Mar  1 23:18:56.132: %PM-P-ERR_DISABLE: channel-misconfig (STP) error detected on
  Po1, putting Po1 in err-disable state
*Mar  1 23:18:58.120: %LINK-3-UPDOWN: Interface FastEthernet0/14, changed state to
  down
*Mar  1 23:18:58.137: %LINK-3-UPDOWN: Interface Port-channel1, changed state to down
*Mar  1 23:18:58.137: %LINK-3-UPDOWN: Interface FastEthernet0/15, changed state to
  down

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


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

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SD)           -        Fa0/14(D)   Fa0/15(D)
```

The messages at the top of the example specifically state what the switch does when determining whether the interface settings match. In this case, SW1 detects the different STP costs. SW1 does not use F0/14, does not use F0/15, and even places them into an err-disabled state. The switch also puts the PortChannel into err-disabled state. As a result, the PortChannel is not operational, and the physical interfaces are also not operational.

To solve this problem, you must reconfigure the physical interfaces to use the same STP settings. In addition, the PortChannel and physical interfaces must be **shutdown**, and then **no shutdown**, to recover from the err-disabled state. (Note that when a switch applies the **shutdown** and **no shutdown** commands to a PortChannel, it applies those same commands to the physical interfaces, as well; so, just do the **shutdown/no shutdown** on the PortChannel interface.)

Analyzing the Switch Data Plane Forwarding

STP and EtherChannel both have an impact on what a switch's forwarding logic can use. STP limits which interfaces the data plane even considers using by placing some ports in a

blocking state (STP) or discarding state (RSTP), which in turn tells the data plane to simply not use that port. EtherChannel gives the data plane new ports to use in the switch's MAC address table—EtherChannels—while telling the data plane to not use the underlying physical interfaces in an EtherChannel in the MAC table.

This (short) third major section of the chapter explores the impact of STP and EtherChannel on data plane logic and a switch's MAC address table.

Predicting STP Impact on MAC Tables

Consider the small LAN shown in Figure P-4. The LAN has only three switches, with redundancy, just big enough to make the point for this next example. The LAN supports two VLANs, 1 and 2, and the engineer has configured STP such that SW3 blocks on a different port in each of the two VLANs. As a result, VLAN 1 traffic would flow from SW3 to SW1 next, and in VLAN 2, traffic would flow from SW3 to SW2 next instead.

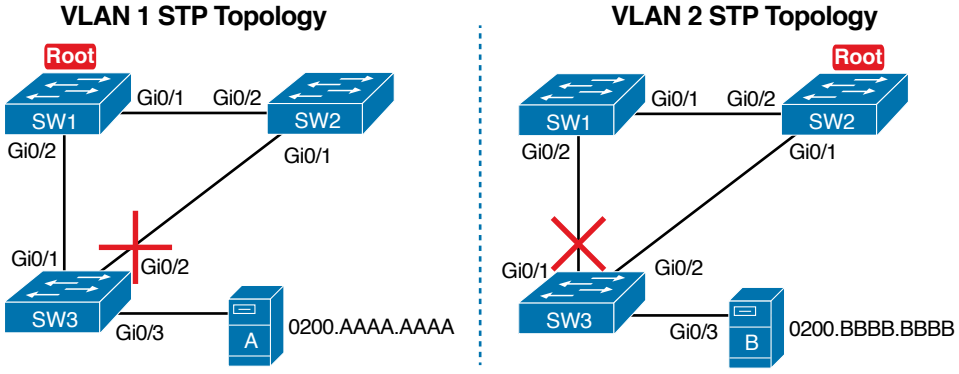

Figure P-4 *Two Different STP Topologies for Same Physical LAN, Two Different VLANs*

Looking at diagrams like those in Figure P-4 makes the forwarding path obvious. Although the figure shows the traffic path, that path is determined by switch MAC learning, which is then impacted by the ports on which STP has set a blocking or discarding state.

For example, consider VLAN 1's STP topology in Figure P-4. Remember, STP blocks on a port on one switch, not on both ends of the link. So, in the case of VLAN 1, SW3's G0/2 port blocks, but SW2's G0/1 does not. Even so, by blocking on a port on one end of the link, that act effectively stops any MAC learning from happening by either device on the link. That is, SW3 learns no MAC addresses on its G0/2 port, and SW2 learns no MAC addresses on its G0/1 port, for these reasons:

- **SW2 learns no MAC addresses on G0/1:** On the blocking (SW3) end of the SW3–SW2 trunk, SW3 will not send frames out that link to SW2, so SW2 will never receive frames from which to learn MAC addresses on SW2's G0/1.
- **SW3 learns no MAC addresses on G0/2:** On the not blocking (SW2) end of the SW3–SW2 trunk, SW2 will flood frames out that port. SW3 receives those frames, but because SW3 blocks, SW3 ignores those received frames and does not learn their MAC addresses.

Given that discussion, can you predict the MAC table entries on each of the three switches for the MAC addresses of servers A and B in Figure P-4? On switch SW2, the entry for

server A, in VLAN 1, should refer to SW2's G0/2 port, pointing to SW1 next, matching the figure. But SW2's entry for server B, in VLAN 2, references SW2's G0/1 port, again matching the figure. Example P-4 shows the MAC tables on SW1 and SW2 as a confirmation.

Example P-4 *Examining SW1 and SW2 Dynamic MAC Address Table Entries*

```
SW1# show mac address-table dynamic
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  1     0200.AAAA.AAAA    DYNAMIC     Gi0/2
  2     0200.BBBB.BBBB    DYNAMIC     Gi0/1
```

```
SW2# show mac address-table dynamic
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  1     0200.AAAA.AAAA    DYNAMIC     Gi0/2
  2     0200.BBBB.BBBB    DYNAMIC     Gi0/1
```

Predicting EtherChannel Impact on MAC Tables

Most designs use multiple links between switches, with those links configured to be part of an EtherChannel. What does that do to the MAC forwarding logic? In short, the switch uses the PortChannel interfaces, and not the physical interfaces bundled into the EtherChannel, in the MAC address table. Specifically:

MAC learning: Frames received in a physical interface that is part of a PortChannel are considered to arrive on the PortChannel interface. So, MAC learning adds the PortChannel interface rather than the physical interface to the MAC address table.

MAC forwarding: The forwarding process will find a PortChannel port as an outgoing interface when matching the MAC address table. Then the switch must take the additional step to choose the outgoing physical interface, based on the load-balancing preferences configured for that PortChannel.

For example, consider Figure P-5, which updates previous Figure P-4 with two-link PortChannels between each pair of switches. With VLAN 1 blocking again on switch SW3, but this time on SW3's PortChannel3 interface, what MAC table entries would you expect to see in each switch? Similarly, what MAC table entries would you expect to see for VLAN 2, with SW3 blocking on its PortChannel2 interface?

The logic of which entries exist on which ports mirrors the logic with the earlier example surrounding Figure P-4. In this case, the interfaces just happen to be PortChannel interfaces. Example P-5 shows the same command from the same two switches as Example P-4: **show mac address-table dynamic** from both SW1 and SW2. (Note that to save length, the MAC table output shows only the entries for the two servers in Figure P-5.)

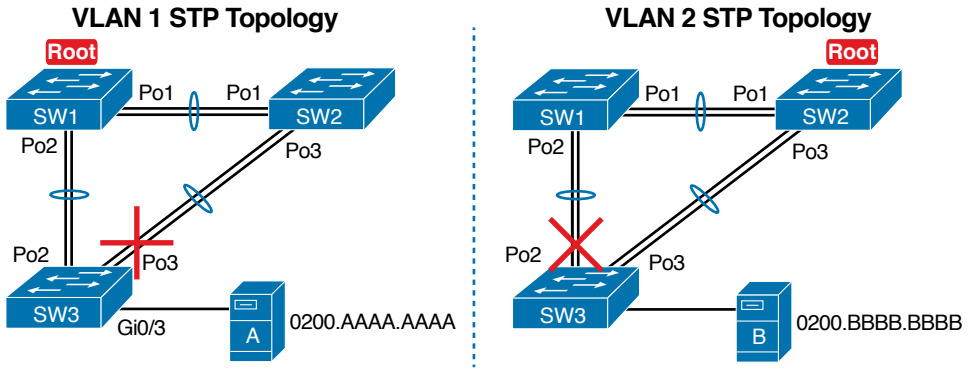

Figure P-5 *VLAN Topology with PortChannels Between Switches*

Example P-5 *SW1 and SW2 MAC Tables with PortChannel Ports Listed*

```
SW1# show mac address-table dynamic
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  1     0200.AAAA.AAAA    DYNAMIC     Po2
  2     0200.BBBB.BBBB    DYNAMIC     Po1
```

```
SW2# show mac address-table dynamic
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  1     0200.AAAA.AAAA    DYNAMIC     Po1
  2     0200.BBBB.BBBB    DYNAMIC     Po3
```

Switches use one of many load-balancing options to then choose the physical interface to use after matching MAC table entries like those shown in Example P-5. By default, Cisco Layer 2 switches often default to use a balancing method based on the source MAC address. In particular, the switch looks at the low-order bits of the source MAC address (which are on the far right of the MAC address in written form). This approach increases the chances that the balancing will be spread somewhat evenly based on the source MAC addresses in use.

Choosing the VLAN of Incoming Frames

To wrap up the analysis of switch data plane forwarding, this section mostly reviews topics already discussed, but it serves to emphasize some important points. The topic is simply this: How does a switch know which VLAN a frame is a part of as the frame enters a switch? You have seen all the information needed to answer this question already, but take the time to review.

First, some interfaces trunk, and in those cases, the frame arrives with a VLAN ID listed in the incoming trunking header. In other cases, the frame does not arrive with a trunking

header, and the switch must look at local configuration. But because the switch will match both the destination MAC address and the frame VLAN ID when matching the MAC address table, knowing how the switch determines the VLAN ID is important.

The following list reviews and summarizes the key points of how a switch determines the VLAN ID to associate with an incoming frame:

Key Topic

Step 1. If the port is an access port, associate the frame with the configured access VLAN (**switchport access vlan** *vlan_id*).

Step 2. If the port is a voice port, or has both an IP Phone and PC (or other data device) connected to the phone:

A. Associate the frames from the data device with the configured access VLAN (as configured with the **switchport access vlan** *vlan_id* command).

B. Associate the frames from the phone with the VLAN ID in the 802.1Q header (as configured with the **switchport voice vlan** *vlan_id* command).

Step 3. If the port is a trunk, determine the frame's tagged VLAN, or if there is no tag, use that incoming interface's native VLAN ID (**switchport trunk native** *vlan_id*).

P

Troubleshooting VLANs and VLAN Trunks

A switch's data plane forwarding processes depend in part on VLANs and VLAN trunking. Before a switch can forward frames in a particular VLAN, the switch must know about a VLAN and the VLAN must be active. And before a switch can forward a frame over a VLAN trunk, the trunk must currently allow that VLAN to pass over the trunk.

This final major section in this chapter focuses on VLAN and VLAN trunking issues, specifically issues that impact the frame switching process. The issues are as follows:

Key Topic

Step 1. Identify all access interfaces and their assigned access VLANs and reassign into the correct VLANs if incorrect.

Step 2. Determine whether the VLANs both exist (either configured or learned with the VLAN Trunking Protocol [VTP]) and are active on each switch. If not, configure and activate the VLANs to resolve problems as needed.

Step 3. Check the allowed VLAN lists, on the switches on both ends of the trunk, and ensure that the lists of allowed VLANs are the same.

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

Step 5. Check the allowed VLANs on each trunk, to make sure that the trunk has not administratively removed a VLAN from being supported on a trunk.

Access VLAN Configuration Incorrect

To ensure that each access interface has been assigned to the correct VLAN, engineers simply need to determine which switch interfaces are access interfaces instead of trunk

interfaces, determine the assigned access VLANs on each interface, and compare the information to the documentation. The **show** commands listed in Table P-1 can be particularly helpful in this process.

Key Topic

Table P-1 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 *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, because they list all the known VLANs and the access interfaces assigned to each VLAN. Be aware, however, that these two commands do not list operational trunks. The output does list all other interfaces (those not currently trunking), no matter whether the interface is in a working or nonworking state.

If the **show vlan** and **show interface switchport** commands are not available in a particular exam question, the **show mac address-table** command can also help identify the access VLAN. This command lists the MAC address table, with each entry including a MAC address, interface, and VLAN ID. If the exam question implies that a switch interface connects to a single device, you should only see one MAC table entry that lists that particular access interface; the VLAN ID listed for that same entry identifies the access VLAN. (You cannot make such assumptions for trunking interfaces.)

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 it is disabled (shut down). This section 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. For this discussion, consider that the only way for a switch to know about a VLAN is to have a **vlan** command configured on the local switch.

Next, the **show vlan** command always lists all VLANs known to the switch, but the **show running-config** command does not. Switches configured as VTP servers and clients do not list the **vlan** commands in the running-config file nor the startup-config file; on these

switches, you must use the **show vlan** command. Switches configured to use VTP transparent mode, or that disable VTP, list the **vlan** configuration commands in the configuration files. (Use the **show vtp status** command to learn the current VTP mode of a switch.)

After you determine that a VLAN does not exist on a switch, the problem might be that the VLAN simply needs to be configured.

Even for existing VLANs, you must also verify whether the VLAN is active. The **show vlan** command should list 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 that *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 P-6 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).

Example P-6 *Enabling and Disabling VLANs on a Switch*

```
SW2# 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
                                                Fa0/14, Fa0/15, Fa0/16, Fa0/17
                                                Fa0/18, Fa0/19, Fa0/20, Fa0/21
                                                Fa0/22, Fa0/23, Fa0/24, Gi0/1
10   VLAN0010                         act/lshut Fa0/13
20   VLAN0020                         active
30   VLAN0030                         act/lshut
40   VLAN0040                         active
SW2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW2(config)# no shutdown vlan 10
SW2(config)# shutdown vlan 20
SW2(config)# vlan 30
SW2(config-vlan)# no shutdown
SW2(config-vlan)# vlan 40
SW2(config-vlan)# shutdown
SW2(config-vlan)#
```

P

Mismatched Trunking Operational States

Trunking can be configured correctly so that both switches forward frames for the same set of VLANs. However, trunks can also be misconfigured, with a couple of different results. In some cases, both switches conclude that their interfaces do not trunk. In other cases, one switch believes that its interface is correctly trunking, while the other switch does not.

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.

With this particular incorrect configuration, the **show interfaces switchport** command on both switches confirms both the administrative state (auto) and the fact that both switches operate as "static access" ports. Example P-7 highlights those parts of the output from this command.

Example P-7 *Operational Trunking State*

```
SW2# show interfaces gigabit0/2 switchport
Name: Gi0/2
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 results in one switch with an operational state of "trunk," while the other switch has an operational state of "static access." When this combination of events happens, the interface works a little. The status on each end will be up/up or connected. Traffic in the native VLAN will actually cross the link successfully. However, traffic in all the rest of the VLANs will not cross the link.

Figure P-6 shows the incorrect configuration along with which side trunks and which does not. The side that trunks (SW1 in this case) enables trunking always, using the command **switchport mode trunk**. However, this command does not disable Dynamic Trunking Protocol (DTP) negotiations. To cause this particular problem, SW1 also disables DTP negotiation using the **switchport nonegotiate** command. SW2's configuration also helps create the problem, by using a trunking option that relies on DTP. Because SW1 has disabled DTP, SW2's DTP negotiations fail, and SW2 does not trunk.

In this case, SW1 treats its G0/1 interface as a trunk, and SW2 treats its G0/2 interface as an access port (not a trunk). As shown in the figure 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 SW2 treats its G0/2 port as an access port. So, SW2 discards any 802.1Q frames received on that port.

To deal with the possibility of this problem, always check the trunk's operational state on both sides of the trunk. The best commands to check trunking-related facts are **show interfaces trunk** and **show interfaces switchport**.

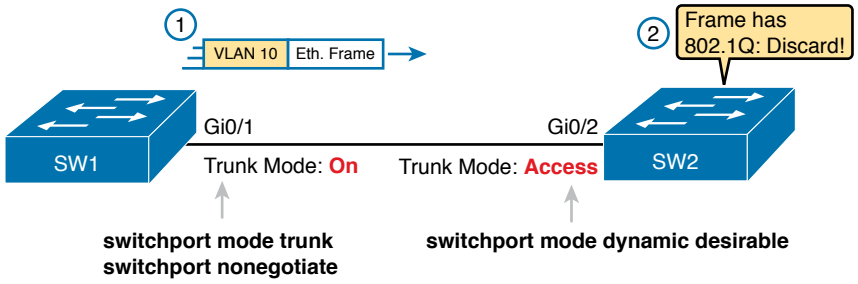

Figure P-6 *Mismatched Trunking Operational States*

NOTE Frankly, in real life, just avoid this kind of configuration. However, the switches do not prevent you from making these types of mistakes, so you need to be ready.

Mismatched Supported VLAN List on Trunks

VLAN trunks on Cisco switches can forward traffic for all defined and active VLANs. However, a particular trunk may not forward traffic for a defined and active VLAN for a variety of other reasons. You should know 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 trunk** command, which only lists information 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 exists and is active on the local switch (as seen in the **show vlan** command).
- 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 has not been VTP-pruned from the trunk. (You can ignore this feature for the purposes of this book; it is mentioned here only because the **show** command mentions it.)
- The trunk is in an STP forwarding state in that VLAN (as also seen in the **show spanning-tree vlan** *vlan-id* command).

Example P-8 shows a sample of the command output from the **show interfaces trunk** command, with the final section of the command output shaded. In this case, the trunk only forwards traffic in VLANs 1 and 4.

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

```
SW1# show interfaces trunk

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

Port        Vlans allowed on trunk
Gi0/1       1-2,P-4094

Port        Vlans allowed and active in management domain
Gi0/1       1,4

Port        Vlans in spanning tree forwarding state and not pruned
Gi0/1       1,4
```

The absence of a VLAN in this last part of the command's output does not necessarily mean that a problem has occurred. In fact, a VLAN might be legitimately excluded from a trunk for any of the reasons in the list just before Example P-8. However, for a given exam question, it can be useful to know why traffic for a VLAN will not be forwarded over a trunk, and the details inside the output identify the specific reasons.

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 P-2 summarizes the headings that precede each list and the reasons why a switch chooses to include or not include a VLAN in each list.

Key Topic

Table P-2 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

Closing with a brief mention of one other trunking topic, you should also check a trunk's native VLAN configuration at this step. 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 accidentally cause frames to leave one VLAN and enter another.

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 page intentionally left blank

APPENDIX Q

Troubleshooting IPv4 Routing Protocols

AUTHOR NOTE This appendix contains an entire chapter that was published as a chapter in one of the past editions of this book or a related book. The author includes this appendix with the current edition as extra reading for anyone interested in learning more. However, note that the content in this appendix has not been edited since it was published in the earlier edition, so references to exams and exam topics, and to other chapters, will be outdated. This appendix was previously published as Chapter 11 of the book *CCNA Routing and Switching ICND2 200-105 Official Cert Guide*, published in 2016.

To troubleshoot a possible IPv4 routing protocol problem, first focus on interfaces, and then on neighbors. The routing protocol configuration identifies the interfaces on which the router should use the routing protocol. After identifying those interfaces, a network engineer can look at the neighbors each router finds on each interface, searching for neighbors that should exist but do not.

This chapter focuses on issues related to these two main branches of logic: on which interfaces should a router enable the routing protocol, and which neighbor relationships should each router create. This chapter's troubleshooting discussions emphasize how to find incorrect configuration problems by using only **show** and **debug** commands.

This chapter first briefly introduces a few broad concepts related to troubleshooting problems with routing protocols. The next major section examines problems related to which interfaces on which a router enables the routing protocol, with the final major section focusing of routing protocol neighbor relationships. Note that the entire chapter moves back and forth between discussing both Enhanced Interior Gateway Routing Protocol (EIGRP) and Open Shortest Path First Version 2 (OSPFv2).

Foundation Topics

Perspectives on Troubleshooting Routing Protocol Problems

Because a routing protocol's job is to fill a router's routing table with the currently best routes, it makes sense that troubleshooting potential problems with routing protocols could begin with the IP routing table. Given basic information about an internetwork, including the routers, their IP addresses and masks, and the routing protocol, you could calculate the subnet numbers that should be in the router's routing table and list the likely next-hop routers for each route. For example, Figure Q-1 shows an internetwork with six subnets. Router R1's routing table should list all six subnets, with three connected routes, two routes learned from R2 (172.16.4.0/24 and 172.16.5.0/24), and one route learned from R3 (172.16.6.0/24).

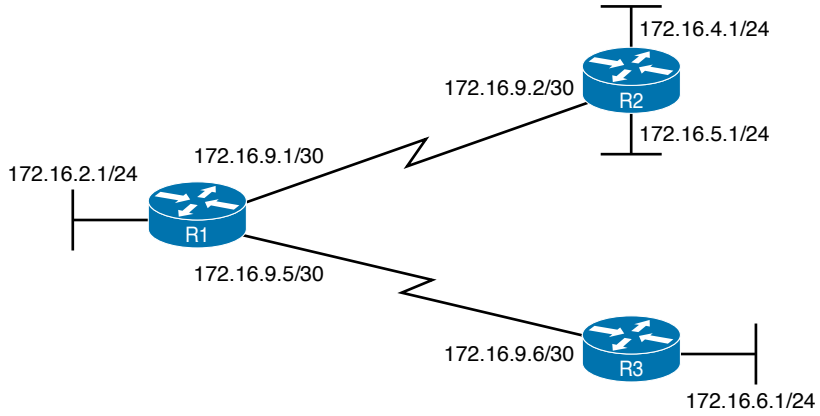

Figure Q-1 *Internetwork with Six Subnets*

So, one possible troubleshooting process is to analyze the internetwork, look at the routing table, and look for missing routes. If one or more expected routes are missing, the next step would be to determine whether that router has learned any routes from the expected next-hop (neighbor) router. The next steps to isolate the problem differ greatly if a router is having problems forming a neighbor relationship with another router, versus having a working neighbor relationship but not being able to learn all routes.

For example, suppose that R1 in Figure Q-1 has learned a route for subnet 172.16.4.0/24 in Figure Q-1 but not for subnet 172.16.5.0/24. In this case, it is clear that R1 has a working neighbor relationship with R2. In these cases, the root cause of this problem might still be related to the routing protocol, or it might not. For example, the problem may be that R2's lower LAN interface is down. However, if R1 did not have a route for both 172.16.4.0/24 and 172.16.5.0/24, R1's neighbor relationship with R2 could be the problem.

Troubleshooting routing protocol problems in real internetworks can be very complex—much more complex than even the most difficult CCNA R&S exam questions. Defining a generic troubleshooting process with which to attack both simple and complex routing protocol problems would require a lot of space and be counterproductive for preparing for

the CCNA exam. This chapter instead offers a straightforward process for attacking routing protocol problems—specifically, problems similar to the depth and complexity of the CCNA exam.

If an exam question appears to be related to a problem with a routing protocol, you can quickly identify some common configuration errors with the following process—even if the question does not list the configuration. The process has three main tasks:

Step 1. Examine the internetwork design to determine on which interfaces the routing protocol should be enabled and which routers are expected to become neighbors.

Step 2. Verify whether the routing protocol is enabled on each interface (as per Step 1). If it isn't, determine the root cause and fix the problem.

Step 3. Verify that each router has formed all expected neighbor relationships. If it hasn't, find the root cause and fix the problem.

For instance, as noted with asterisks in Figure Q-2, each router should enable the routing protocol on each of the interfaces shown in the figure. Also, routing protocol neighbor relationships should form between R1 and R2, and R1 and R3, but not between R2 and R3.

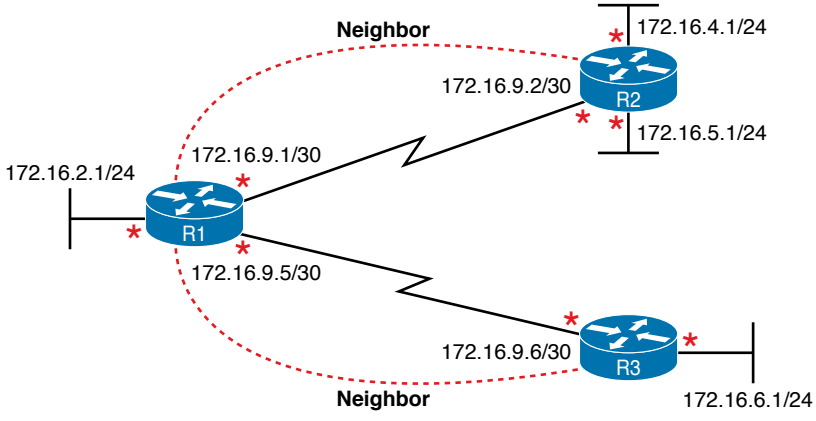

Figure Q-2 *Routing Protocol Interfaces and Neighbor Relationships*

While the concepts outlined in Figure Q-2 should be somewhat obvious by now, this chapter discusses how some of the most common configuration mistakes can impact the interfaces used by a routing protocol and whether a routing protocol creates neighbor relationships.

Interfaces Enabled with a Routing Protocol

This section examines the second major troubleshooting step outlined in the previous section of the chapter: how to verify the interfaces on which the routing protocol has been enabled. Both EIGRP and OSPF configuration enable the routing protocol on an interface by using the **network** router subcommand. For any interfaces matched by the **network** commands, the routing protocol tries the following two actions:

Key Topic

- Attempt to find potential neighbors on the subnet connected to the interface
- Advertise the subnet connected to that interface

At the same time, the **passive-interface** router subcommand can be configured so that the router does not attempt to find neighbors on the interface (the first action just listed), but still advertises the connected subnet (the second action).

Three **show** commands are all that is needed to know exactly which interfaces have been enabled with EIGRP and which interfaces are passive. In particular, the **show ip eigrp interfaces** command lists all EIGRP-enabled interfaces that are not passive interfaces. The **show ip protocols** command essentially lists the contents of the configured **network** commands for each routing protocol and a separate list of the passive interfaces. Comparing these two commands identifies all EIGRP-enabled interfaces and those that are passive.

For OSPF, the command works slightly differently, with the **show ip ospf interface brief** command listing all OSPF-enabled interfaces (including passive interfaces). Using this command, along with the list of passive interfaces listed by the **show ip protocols** command, again identifies all fully enabled OSPF interfaces as well as all passive interfaces.

Table Q-1 summarizes the commands that identify the interfaces on which OSPFv2 and EIGRP are enabled for easier reference.

Key Topic

Table Q-1 Key Commands to Find Routing Protocol-Enabled Interfaces

Command	Key Information	Lists Passive Interfaces?
show ip eigrp interfaces	Lists the interfaces on which EIGRP is enabled (based on the **network** commands), *excluding* passive interfaces.	No
show ip ospf interface brief	Lists the interfaces on which the OSPFv2 is enabled (based on the **network** router subcommands or **ip ospf** interface subcommands), *including* passive interfaces.	Yes
show ip protocols	Lists the contents of the **network** configuration commands for each routing process, and lists enabled but passive interfaces.	Yes

NOTE All the commands in Table Q-1 list the interfaces regardless of interface status, in effect telling you the results of the **network** and **passive-interface** configuration commands.

So, for the major troubleshooting step covered in this section, the task is to use the commands in Table Q-1 and analyze the output. First, an EIGRP example will be shown, followed by an OSPF example.

EIGRP Interface Troubleshooting

This section shows a few examples of the commands in the context of Figure Q-3, which is used in all the examples in this chapter.

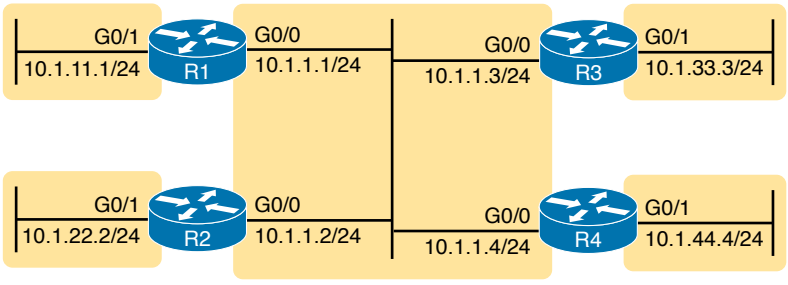

Figure Q-3 *Internetwork for EIGRP/OSPF Troubleshooting Examples*

This example includes four routers, with the following scenario in this case:

- R1 and R2 are configured correctly on both LAN interfaces.
- R3 is mistakenly not enabled with EIGRP on its G0/1 interface.
- R4 meant to use a **passive-interface G0/1** command because no other routers are off R4's G0/1 LAN. However, R4 has instead configured a **passive-interface G0/0** command.

This example begins by showing the working details between Routers R1 and R2, and then moves on to discuss the issues related to R3 and R4.

Examining Working EIGRP Interfaces

Examples Q-1 and Q-2 list configuration and **show** commands for R1 and R2, respectively. Each lists the related configuration, the **show ip eigrp interfaces** and **show ip protocols** command, and the EIGRP-learned routes on each router.

Example Q-1 *EIGRP Interfaces Problem: R1 Commands*

```
R1# show running-config
! only pertinent lines shown
router eigrp 99
 network 10.0.0.0
!
R1# show ip eigrp interfaces
EIGRP-IPv4 Interfaces for AS(99)
                   Xmit Queue   PeerQ        Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Gi0/0          2        0/0        0/0          2       0/0           50           0
Gi0/1          0        0/0        0/0          0       0/0            0           0

R1# show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "eigrp 99"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
```

```
  EIGRP-IPv4 Protocol for AS(99)
    Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
    NSF-aware route hold timer is 240
    Router-ID: 1.1.1.1
    Topology : 0 (base)
      Active Timer: 3 min
      Distance: internal 90 external 170
      Maximum path: 4
      Maximum hopcount 100
      Maximum metric variance 1

  Automatic Summarization: disabled
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.1.1.2              90      09:55:51
    10.1.1.3              90      00:02:00
  Distance: internal 90 external 170

R1# show ip route eigrp
! Legend omitted for brevity

      10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
D        10.1.22.0/24 [90/30720] via 10.1.1.2, 00:00:40, GigabitEthernet0/0
```

Example Q-2 *EIGRP Interfaces Problem: R2 Commands*

```
R2# show running-config
! only pertinent lines shown
router eigrp 99
 network 10.1.0.0 0.0.255.255

R2# show ip eigrp interfaces
EIGRP-IPv4 Interfaces for AS(99)
                   Xmit Queue   PeerQ        Mean   Pacing Time   Multicast    Pending
Interface   Peers  Un/Reliable  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Gi0/0           2        0/0       0/0           1       0/1           50           0
Gi0/1           0        0/0       0/0           0       0/0            0           0

R2# show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "eigrp 99"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
```

```
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP-IPv4 Protocol for AS(99)
    Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
    NSF-aware route hold timer is 240
    Router-ID: 2.2.2.2
    Topology : 0 (base)
      Active Timer: 3 min
      Distance: internal 90 external 170
      Maximum path: 4
      Maximum hopcount 100
      Maximum metric variance 1

  Automatic Summarization: disabled
  Maximum path: 4
  Routing for Networks:
    10.1.0.0/16
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.1.1.3              90      00:02:30
    10.1.1.1              90      09:56:20
  Distance: internal 90 external 170

R2# show ip route eigrp
! Legend omitted for brevity
      10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
D        10.1.11.0/24 [90/30720] via 10.1.1.1, 00:03:25, GigabitEthernet0/0
```

The **show ip eigrp interfaces** command output on both R1 and R2 shows how both R1 and R2 have configured EIGRP using process ID 99, and that EIGRP has been enabled on both G0/0 and G0/1 on both these routers. This command lists only interfaces on which EIGRP has been enabled, excluding passive interfaces.

The highlighted parts of the **show ip protocols** command output on each router are particularly interesting. These sections show the parameters of the configured **network** commands. The **show ip protocols** command lists a separate line under the header "Routing for Networks," one for each configured **network** command. Example Q-1's output suggests R1 has a **network 10.0.0.0** configuration command (as shown at the beginning of the example), and Example Q-2's "10.1.0.0/16" suggests R2 has a **network 10.1.0.0 0.0.255.255** command.

Examining the Problems with EIGRP Interfaces

The next few pages now look at the problems caused by the configuration on Routers R3 and R4.

First, Example Q-2 gives brief insight into the current problem caused by R3. The end of R2's **show ip protocols** command (Example Q-2) lists two routing information sources: 10.1.1.1 (R1) and 10.1.1.3 (R3). However, R2 has learned only one EIGRP route (10.1.11.0/24), as shown in the **show ip route eigrp** command output. When working properly, R2 should learn three EIGRP routes—one for each of the other LAN subnets shown in Figure Q-3.

Example Q-3 shows the root cause on R3. First, R3's **show ip eigrp interfaces** command lists G0/0, but not G0/1, so a problem might exist with how EIGRP has been configured on G0/1. The configuration at the top of the example lists the root cause: an incorrect **network** command, which does not enable EIGRP on R3's G0/1 interface.

Example Q-3 *EIGRP Problems on R3*

```
R3# show running-config
! lines omitted for brevity
router eigrp 99
 network 10.1.1.3 0.0.0.0
 network 10.1.13.3 0.0.0.0
 auto-summary

R3# show ip eigrp interfaces
EIGRP-IPv4 Interfaces for AS(99)
                   Xmit Queue   PeerQ        Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Gi0/0          2        0/0         0/0          1       0/1           50           0

R3# show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "eigrp 99"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP-IPv4 Protocol for AS(99)
    Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
    NSF-aware route hold timer is 240
    Router-ID: 3.3.3.3
    Topology : 0 (base)
      Active Timer: 3 min
      Distance: internal 90 external 170
      Maximum path: 4
      Maximum hopcount 100
      Maximum metric variance 1

  Automatic Summarization: disabled
  Maximum path: 4
  Routing for Networks:
    10.1.1.3/32
    10.1.13.3/32
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.1.1.2              90      00:05:14
    10.1.1.1              90      00:05:14
  Distance: internal 90 external 170
```

The root cause of R3's problem is that R3 has a **network 10.1.13.3 0.0.0.0** configuration command, which does not match R3's 10.1.33.3 G0/1 IP address. If the configuration was not available in the exam question, the **show ip protocols** command could be used to essentially see the same configuration details. In this case, the **show ip protocols** command on R3 lists the text "10.1.13.3/32" as a reference to the contents of the incorrect **network** command's parameters, with "/32" translating to a wildcard mask of 32 binary 0s, or decimal 0.0.0.0.

R3's incorrect configuration means that two actions do not happen on R3's G0/1 interface. First, R3 does not try to find neighbors on its G0/1 interface, which is not a big deal in this case. However, R3 also does not advertise subnet 10.1.33.0/24, the connected subnet off R3's G0/1 interface.

Moving on to R4's problem, Example Q-4 shows why R1 and R2 do not learn R4's 10.1.44.0/24 subnet. In this case, on R4, the engineer could have correctly used a **passive-interface gigabitethernet0/1** router subcommand because no other routers should exist off R4's G0/1 interface. However, the engineer mistakenly made R4's G0/0 interface passive.

Example Q-4 *EIGRP Problems on R4*

```
R4# show running-config
! lines omitted for brevity
router eigrp 99
 passive-interface GigabitEthernet0/0
 network 10.0.0.0
 auto-summary

R4# show ip eigrp interfaces
EIGRP-IPv4 Interfaces for AS(99)
                   Xmit Queue   PeerQ        Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Gi0/1          0        0/0          0/0         0       0/1            0           0

R4# show ip protocols | begin Routing for Networks
  Routing for Networks:
    10.0.0.0
  Passive Interface(s):
    GigabitEthernet0/0
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: internal 90 external 170
```

NOTE The last command on the example, **show ip protocols | begin Routing for Networks**, lists the command output, but starting with the line with the literal case-sensitive string **Routing for Networks**. You can use this feature with any output from a command when you prefer to view only later lines of the command's output.

To find this mistake without the configuration, Example Q-4 lists two useful commands. R4's **show ip eigrp interfaces** command omits the (G0/0) passive interface, which means that R4 will not attempt to find EIGRP neighbors off that interface. Also, the highlighted part of R4's **show ip protocols** command output lists G0/0 as a passive interface, which again means that R4 does not even attempt to become neighbors with others off its G0/0 interface.

OSPF Interface Troubleshooting

OSPF has the same basic requirements as EIGRP for interfaces, with a few exceptions. First, EIGRP routers need to use the same autonomous system number (ASN) as their neighboring routers, as configured in the **router eigrp** *asn* global configuration command. OSPF routers can use any process ID on the **router ospf** *process-id* command, with no need to match their neighbors. Second, OSPF requires that the interfaces connected to the same subnet be assigned to the same OSPF area, whereas EIGRP has no concept of areas.

Example Q-5 shows a mostly working OSPF internetwork, again based on Figure Q-3. The problem in this case relates to the area design, as shown in Figure Q-4, the revised version of Figure Q-3. All subnets should be placed into area 0. However, the engineer made a configuration mistake on R2, putting both its interfaces into area 1. As a result, R2's G0/0 interface breaks the OSPF design rule of being in the same subnet as R1, R3, and R4, but not being in the same OSPF area.

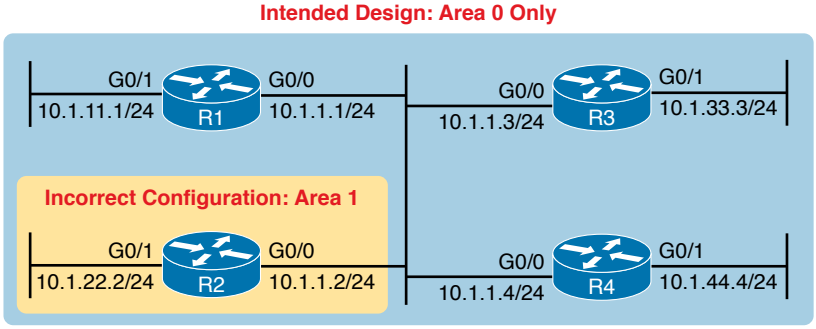

Figure Q-4 *Intended Area Design Using Only Area 0, with R2 Breaking the Design*

Example Q-5 begins to break down the problem by looking at the status of OSPF on the router interfaces of R1 and R2, using the **show ip ospf interface brief** command.

Example Q-5 **show ip interface brief** *on R1 and R2*

```
R1> show ip ospf interface brief
Interface    PID   Area            IP Address/Mask    Cost  State Nbrs F/C
Gi0/1        1     0               10.1.11.1/24       1     DR    0/0
Gi0/0        1     0               10.1.1.1/24        1     DROTH 2/2

! The following command is from R2
R2> show ip ospf interface brief
Interface    PID   Area            IP Address/Mask    Cost  State Nbrs F/C
Gi0/1        2     1               10.1.22.2/24       1     WAIT  0/0
Gi0/0        2     1               10.1.1.2/24        1     WAIT  0/0
```

From a general perspective, the **show ip ospf interface brief** command lists output similar to the **show ip eigrp interface** command, with one line for each enabled interface. The **show ip ospf interface** command, not shown in the example, lists detailed OSPF information for each interface.

Specific to this problem, the output in Example Q-5 shows that R1 and R2 both have OSPF enabled on both LAN interfaces. However, this command also lists the area number for each interface, with R2 having both LAN interfaces in area 1. Also, these commands repeat the IP address and mask of the interfaces, so together, you can see that R1's 10.1.1.1/24 address is in the same subnet as R2's 10.1.1.2/24 address, putting these two routers in the same subnet but in different OSPF areas.

Example Q-6 shows another way to look at the problem, with the **show ip protocols** commands on both R1 and R2. Because this command lists the OSPF **network** commands in shorthand form, it can point toward a possible configuration error, even if the configuration is not available.

Example Q-6 *Finding OSPF Configuration Errors with* **show ip protocols** *R1 and R2*

```
R1> show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 1.1.1.1
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    10.0.0.0 0.255.255.255 area 0
  Routing Information Sources:
    Gateway         Distance      Last Update
    2.2.2.2              110      00:14:32
    3.3.3.3              110      00:14:32
    10.1.44.4            110      00:14:42
  Distance: (default is 110)

R1> show ip route ospf
! Legend omitted for brevity

      10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
O        10.1.33.0/24 [110/2] via 10.1.1.3, 00:15:32, GigabitEthernet0/0
O        10.1.44.0/24 [110/2] via 10.1.1.4, 00:15:42, GigabitEthernet0/0
```

```
! Now moving to Router R2

R2> show ip protocols
*** IP Routing is NSF aware ***
```

```
Routing Protocol is "ospf 2"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 2.2.2.2
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    10.0.0.0 0.255.255.255 area 1
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: (default is 110)

R2>
Nov 15 12:16:39.377: %OSPF-4-ERRRCV: Received invalid packet: mismatched area
ID, from backbone area must be virtual-link but not found from 10.1.1.1,
GigabitEthernet0/0
```

Interestingly, a closer look at R2's **show ip protocols** command output, particularly the highlighted portion, points out the configuration error. As usual, the section with the heading "Routing for Networks:" points to a shorthand version of the configuration. In this case, the highlighted phrase "10.0.0.0 0.255.255.255 area 1" is actually the exact syntax of the one **network** command on Router R2, minus the word *network*, or **network 10.0.0.0 0.255.255.255 area 1**. Because Figure Q-4 shows the design should put all interfaces in area 0, reconfiguring this command to instead be **network 10.0.0.0 0.255.255.255 area 0** would solve this particular problem.

The end of the example also shows an unsolicited log message generated by Router R2, notifying the console user that this router has received a Hello from a router in a different area.

As you check the interfaces, you could also check several other details. It makes sense to go ahead and check the interface IP addresses, masks, and interface status values by using the **show interfaces** and **show ip interface brief** commands. In particular, it is helpful to note which interfaces are up/up, because a router will send no packets (including routing protocol packets) out interfaces that are not in an up/up state.

Neighbor Relationships

This final major section of the chapter examines the large number of facts that each router must check with each potential neighbor before the two routers become neighbors.

At a very basic level, routing protocols can easily create neighbor relationships using a Hello protocol. First, the routing protocol must be enabled on an interface. In addition, the interface may not be configured as a passive interface, because that stops the routing protocol from sending the Hello messages.

Beyond this basic process, the routing protocols actually check several other parameters to find out whether the routers should become neighbors. Both OSPF and EIGRP use Hello messages, and these messages each list information used to perform some basic verification

checks. For example, as just shown in earlier Example Q-5, an OSPF router should not become neighbors with another router in another area because all routers on a common subnet should be in the same OSPF area by design.

After an EIGRP or OSPF router hears a Hello from a new neighbor, the routing protocol examines the information in the Hello, and compares that information with the local router's own settings. If the settings match, great. If not, the routers do not become neighbors. Because there is no formal term for all these items that a routing protocol considers, this book just calls them *neighbor requirements*.

Table Q-2 lists the neighbor requirements for both EIGRP and OSPF. Following the table, the next few pages examine some of these settings for both EIGRP and OSPF, again using examples based on Figure Q-3.

NOTE Even though it is important to study and remember the items in this table, when reading this chapter the first time, just keep reading. When later reviewing the chapter or part, make sure you remember the details in the table.

Key Topic

Table Q-2 Neighbor Requirements for EIGRP and OSPF

Requirement	EIGRP	OSPF
Interfaces must be in an up/up state.	Yes	Yes
Interfaces must be in the same subnet.	Yes	Yes
Access control lists (ACL) must not filter routing protocol messages.	Yes	Yes
Must pass routing protocol neighbor authentication (if configured).	Yes	Yes
Must use the same ASN/PID on the **router** configuration command.	Yes	No
Hello and hold/dead timers must match.	No	Yes
Router IDs (RID) must be unique.	No[1]	Yes
K-values must match.	Yes	N/A
Must be in the same area.	N/A	Yes

[1] Having duplicate EIGRP RIDs does not prevent routers from becoming neighbors, but it can cause problems when external EIGRP routes are added to the routing table.

Q

Unlike most of the neighbor requirements listed in Table Q-2, the first three requirements have very little to do with the routing protocols themselves. The two routers must be able to send packets to each other over the physical network to which they are both connected. To do that, the router interfaces must be up/up, and they must be in the same subnet. In addition, the routers must not be using an ACL that filters the routing protocol traffic.

For instance, OSPF sends many messages to the well-known multicast IP addresses 224.0.0.5 and 224.0.0.6, whereas EIGRP uses 224.0.0.10. An ACL command like **access-list 101 deny ip any host 224.0.0.10**, in an inbound ACL on a router interface, would filter incoming EIGRP packets. Or, an ACL command like **access-list 102 deny ospf any any** could filter all OSPF traffic. Even more difficult to notice is an ACL that has lots of **permit** commands that match different TCP and UDP port numbers, but does not match the routing protocol explicitly, so the routing protocol packets match the implicit deny any at the end of the ACL. So, take extra care to watch for ACLs, especially when it seems like all the routing protocol configuration looks good.

In practice, before examining the rest of the details of why two routers do not become neighbors, confirm that the two routers can ping each other on the local subnet. If the ping fails, investigate all the Layer 1, 2, and 3 issues that could prevent the ping from working (such as an interface not being up/up).

Now, on to the specific discussions about EIGRP and OSPF. Because the details differ slightly between the two routing protocols, this section first examines EIGRP, followed by OSPF.

NOTE This section assumes that the routing protocol has actually been enabled on each required interface, as covered earlier in this chapter in the "Interfaces Enabled with a Routing Protocol" section.

EIGRP Neighbor Verification Checks

Any two EIGRP routers that connect to the same data link, and whose interfaces have been enabled for EIGRP and are not passive, will at least consider becoming neighbors. To quickly and definitively know which potential neighbors have passed all the neighbor requirements for EIGRP, just look at the output of the **show ip eigrp neighbors** command. This command lists only neighbors that have passed all the neighbor verification checks.

Example Q-7 shows an example of the **show ip eigrp neighbors** command, with the four routers from Figure Q-3 again. In this case, all the routers have been configured correctly, so each has a neighbor relationship with the other three routers on the same LAN subnet.

Example Q-7 *R1* **show ip eigrp neighbors** *Command with All Problems Fixed*

```
R1# show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(99)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
1   10.1.1.3                Gi0/0                    13 00:00:20    1   100  0  31
2   10.1.1.4                Gi0/0                    13 00:00:43   80   480  0  10
0   10.1.1.2                Gi0/0                    13 00:13:52    1   100  0  20
```

If the **show ip eigrp neighbors** command does not list one or more expected neighbors, the first problem isolation step should be to find out if the two routers can ping each other's IP addresses on the same subnet. If that works, start looking at the list of neighbor verification checks, as relisted for EIGRP here in Table Q-3. Table Q-3 summarizes the EIGRP neighbor requirements, while noting the best commands with which to determine which requirement is the root cause of the problem.

Key Topic

Table Q-3 EIGRP Neighbor Requirements and the Best **show/debug** Commands

Requirement	Best Commands to Isolate the Problem
Must be in the same subnet.	**show interfaces**, **show ip interface**
Must use the same ASN on the **router** configuration command.	**show ip eigrp interfaces**, **show ip protocols**
Must pass EIGRP neighbor authentication.	**debug eigrp packets**
K-values must match.	**show ip protocols**

Of the four rows of requirements listed in Table Q-3, the first two have already been discussed in this chapter, and do not need further discussion.

For EIGRP authentication (the third item in the table), EIGRP supports the capability for routers to trust routers as EIGRP neighbors only if the routers share the same security key (password); if that check fails, the neighbor relationship fails. By default, routers do not attempt EIGRP authentication, which allows the routers to form EIGRP neighbor relationships. If one router uses authentication, and the other does not, they will not become neighbors. If both use authentication, they must use the same authentication key to become neighbors.

The last item in the table, EIGRP K-values, refers to the EIGRP metric components and the metric calculation. These K-values are variables that basically enable or disable the use of the different components in the EIGRP composite metric. Cisco recommends leaving these values at their default settings, using only bandwidth and delay in the metric calculation. The K-value settings must match before two routers will become neighbors; you can check the K-values on both routers with the **show ip protocols** command.

EIGRP Neighbor Troubleshooting Example

Example Q-8 shows three problems that can cause EIGRP routers to fail to become neighbors. This example uses the usual design for this chapter, as repeated in Figure Q-5. The figure shows the same routers, and same interfaces, but with the following problems:

- R2 has been configured with IP address 10.1.2.2/24 in a different subnet than R1, R3, and R4.
- R3 has been configured to use ASN 199 with the **router eigrp 199** command instead of ASN 99, as used on the other three routers.
- R4 has been configured to use message digest 5 (MD5) authentication, whereas the other routers use no authentication.

R1 can actually detect two of the problems using local commands and messages, as shown in Example Q-8. R1 generates an unsolicited log message for the mismatched subnet problem, and a **debug** command on R1 can reveal the authentication failure. The example shows some running commentary inside the example.

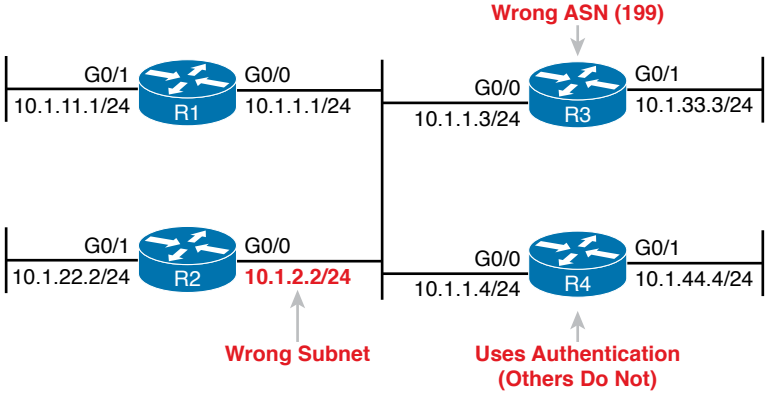

Figure Q-5 *Summary of Problems That Prevent EIGRP Neighbors on the Central LAN*

Example Q-8 *Common Problems Preventing the Formation of EIGRP Neighbors (R1)*

```
! First, R1 has no neighbor relationships yet. R1 uses ASN (process) 99.
R1# show ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(99)

R1#
! Next, R1 generates a log message, which shows up at the console, stating
! that the router with IP address 10.1.2.2 is not on the same subnet as R1.
!
*Nov 15 16:19:14.740: %DUAL-6-NBRINFO: EIGRP-IPv4 99: Neighbor 10.1.2.2
(GigabitEthernet0/0) is blocked: not on common subnet (10.1.1.1/24)

! Next, R1 enables a debug that shows messages for each packet received from R4,
! which uses the wrong password (authentication key string)
!
R1# debug eigrp packets
EIGRP Packets debugging is on
    (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK, STUB, SIAQUERY,
     SIAREPLY)
R1#

*Nov 15 16:20:30.865: EIGRP: Gi0/0: ignored packet from 10.1.1.4, opcode = 5
  (authentication off or key-chain missing)
```

Example Q-8 shows some evidence of the mismatched subnet with R2, and the invalid authentication problem with R4. Even without knowing the details, it is easy to imagine that if one router's EIGRP process uses authentication with a defined password, and the other does not, that authentication will fail. The result? Neighbor relationships do not form.

Example Q-8 shows details about two of the problems, but not any details about the incorrect ASN configured on R3. Example Q-9 shows those details by listing excerpts from two **show** commands on R3, both of which identify the ASN configured on that router. By using these same commands on all the routers, you could note that R1, R2, and R4 use ASN 99, whereas R3 uses 199, as shown in Example Q-9.

Example Q-9 *Displaying the Incorrect ASN (199) on R3*

```
R3# show ip protocols
Routing Protocol is "eigrp 199"
!
! The first line of output from show ip eigrp interfaces lists ASN 199
!
R3# show ip eigrp interfaces
EIGRP-IPv4 Interfaces for AS(199)
                        Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface        Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Gi0/0              0        0/0         0       0/1            0           0
Gi0/1              0        0/0         0       0/1            0           0
```

OSPF Neighbor Troubleshooting

Similar to EIGRP, a router's **show ip ospf neighbor** command lists all the neighboring routers that have met all the requirements to become an OSPF neighbor as listed in Table Q-2. So, the first step in troubleshooting OSPF neighbors is to look at the list of neighbors.

Example Q-10 lists the output of a **show ip ospf neighbor** command on Router R2, from Figure Q-4. All four routers sit on the same LAN subnet, in area 0, with correct configurations, so all four routers form a valid OSPF neighbor relationship.

Example Q-10 *Normal Working* **show ip ospf neighbors** *Command on Router R2*

```
R2# show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           1   FULL/BDR        00:00:37    10.1.1.1        GigabitEthernet0/0
3.3.3.3           1   2WAY/DROTHER    00:00:37    10.1.1.3        GigabitEthernet0/0
4.4.4.4           1   FULL/DR         00:00:31    10.1.1.4        GigabitEthernet0/0
```

First, note that the neighbor IDs, listed in the first column, identify neighbors by their router ID (RID). For this example network, all four routers use an easily guessed RID. Further to the right, the Address column lists the interface IP address used by that neighbor on the common subnet.

A brief review of OSPF neighbor states can help you understand a few of the subtleties of the output in the example. A router's listed status for each of its OSPF neighbors—the neighbor's state—should settle into either a 2-way or full state under normal operation. For neighbors that do not need to directly exchange their databases, typically two non-designated router (DR) routers on a LAN, the routers should settle into a 2-way neighbor state. In most cases, two neighboring routers need to directly exchange their full link-state databases (LSDB) with each other. As soon as that process has been completed, the two routers settle into a full neighbor state.

In Example Q-10, Router R4 is the DR, and R1 is the backup DR (BDR), so R2 and R3 (as non-DRs) do not need to directly exchange routes. Therefore, R2's neighbor state for R3 (RID 3.3.3.3) in Example Q-10 is listed as 2-way.

NOTE Notably, OSPF neighbors do not have to use the same process ID on the **router ospf** *process-id* command to become neighbors. In Example Q-10, all four routers use different PIDs.

If the **show ip ospf neighbor** command does not list one or more expected neighbors, you should confirm, even before moving on to look at OSPF neighbor requirements, that the two routers can ping each other on the local subnet. But if the two neighboring routers can ping each other, and the two routers still do not become OSPF neighbors, the next step is to examine each of the OSPF neighbor requirements. Table Q-4 summarizes the requirements, listing the most useful commands with which to find the answers.

Key Topic

Table Q-4 OSPF Neighbor Requirements and the Best **show/debug** Commands

Requirement	Best show Command	Best debug Command
Must be in the same subnet.	**show interfaces**	**debug ip ospf hello**
Hello and dead timers must match.	**show ip ospf interface**	**debug ip ospf hello**
Must be in the same area.	**show ip ospf interface brief**	**debug ip ospf adj**
RIDs must be unique.	**show ip ospf**	(N/A; log messages identify this problem)
Must pass any neighbor authentication.	**show ip ospf interface**	**debug ip ospf adj**

This topic looks at a couple of OSPF neighbor problems using the usual four-router network from Figure Q-4, with all interfaces in area 0. However, the following problems have been introduced into the design:

- R2 has been configured with both LAN interfaces in area 1, whereas the other three routers' G0/0 interfaces are assigned to area 0.
- R3 is using the same RID (1.1.1.1) as R1.
- R4 has been configured with a Hello/dead timer of 5/20 on its G0/0 interface, instead of the 10/40 used (by default) on R1, R2, and R3.

Figure Q-6 shows these same problems for reference.

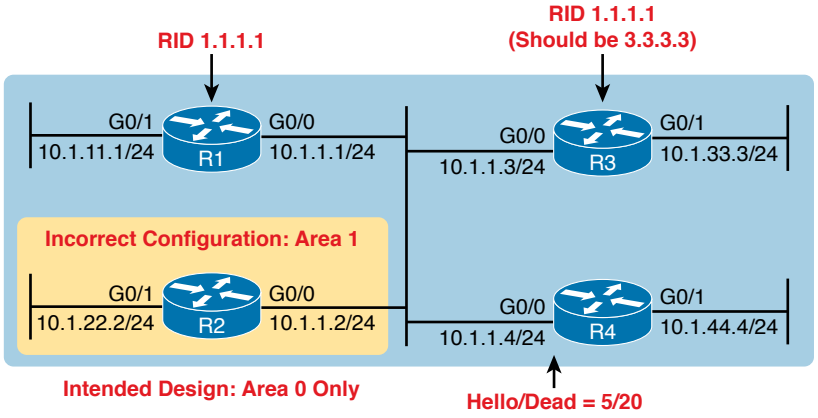

Figure Q-6 *Summary of Problems That Prevent OSPF Neighbors on the Central LAN*

Finding Area Mismatches

Earlier in this chapter, the "OSPF Interface Troubleshooting" section showed how to use the **show ip ospf interface** command to list the area numbers and find OSPF area mismatches. This next topic shows how to see that same issue using the **debug ip ospf adj** command, as shown in Example Q-11. This command lists messages related to OSPF neighbor adjacency events, and shows messages that identify the area mismatch (with R2).

Example Q-11 *Finding Mismatched Area Problem with R1* **debug**

```
R1# debug ip ospf adj
OSPF adjacency events debugging is on
R1#
*Nov 15 13:42:02.288: OSPF-1 ADJ   Gi0/0: Rcv pkt from 10.1.1.2, area 0.0.0.0,
  mismatched area 0.0.0.1 in the header
R1#
R1# undebug all
All possible debugging has been turned off
```

As noted in Table Q-4, the **debug ip ospf adj** command helps troubleshoot mismatched OSPF area problems. The first part of the highlighted message in the example lists shorthand about a received packet ("Rcv pkt") from 10.1.1.2, which is R2's IP address. The rest of the message mentions R1's area (0.0.0.0), and the area claimed by the other router (0.0.0.1). (Note that the message lists the 32-bit area number as a dotted-decimal number.)

This particular example focuses on the symptom (that a neighbor relationship does not start), and the debug messages that identify the problem (mismatched areas). However, finding the configuration error may take some work, because the problem could be more complex than just having the wrong area number configured on a command.

One harder-to-notice configuration error happens when the configuration has multiple **network** commands, with different area numbers, that all happen to match one interface's IP address. IOS stores the OSPF **network** commands to the configuration in the same order they are configured (which is the same order listed in the output of **show running-config**). IOS processes the commands in sequence, so that the first **network** command that matches a particular interface is used to set the OSPF area number.

For instance, imagine a router with interface G0/1 configured with IP address 1.1.1.1. The OSPF configuration lists the following two **network** commands, in that order. Both would match the interface IP address of 1.1.1.1, so IOS uses the first command, which lists area 1. IOS would not use the second command, even though it uses a wildcard mask that is more specific.

- **network 1.0.0.0 0.255.255.255 area 1**
- **network 1.1.1.1 0.0.0.0 area 0**

Another tricky configuration error that can result in an area mismatch occurs when configuring both the **network** OSPF subcommand and the **ip ospf** interface subcommand on the same router. IOS supports using both on the same router at the same time. However, IOS does not prevent a case in which a **network** command attempts to enable OSPF in one area, and the **ip ospf** interface subcommand attempts to enable OSPF in a different area. When that happens, IOS uses the area number defined in the **ip ospf** interface subcommand.

For instance, with the two **network** commands just listed, if the **ip ospf 1 area 5** command was configured on that router's interface, that interface would be in area 5; IOS would prefer that setting over any OSPF **network** command.

NOTE Using both **network** router subcommands and **ip ospf** interface subcommands allows an easier migration from the older to newer style OSPF configuration. However, most enterprises today would use either **network** commands or **ip ospf** commands in one router.

Finding Duplicate OSPF Router IDs

Next, Example Q-12 shows R1 and R3 both trying to use RID 1.1.1.1. Interestingly, both routers automatically generate a log message for the duplicate OSPF RID problem between R1 and R3; the end of Example Q-12 shows one such message. For the exams, just use the **show ip ospf** commands on both R3 and R1 to easily list the RID on each router, noting that they both use the same value.

Example Q-12 *Comparing OSPF Router IDs on R1 and R3*

```
! Next, on R3: R3 lists the RID of 1.1.1.1
!
R3# show ip ospf
 Routing Process "ospf 3" with ID 1.1.1.1
 Start time: 00:00:37.136, Time elapsed: 02:20:37.200
! lines omitted for brevity
```

```
! Back to R1: R1 also uses RID 1.1.1.1

R1# show ip ospf
 Routing Process "ospf 1" with ID 1.1.1.1
 Start time: 00:01:51.864, Time elapsed: 12:13:50.904
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Supports Link-local Signaling (LLS)
 Supports area transit capability
 Supports NSSA (compatible with RFC 3101)
 Event-log enabled, Maximum number of events: 1000, Mode: cyclic
 Router is not originating router-LSAs with maximum metric
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Incremental-SPF disabled
 Minimum LSA interval 5 secs
 Minimum LSA arrival 1000 msecs
 LSA group pacing timer 240 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
 Number of external LSA 0. Checksum Sum 0x000000
 Number of opaque AS LSA 0. Checksum Sum 0x000000
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
```

```
 Number of areas transit capable is 0
 External flood list length 0
 IETF NSF helper support enabled
 Cisco NSF helper support enabled
 Reference bandwidth unit is 100 mbps
    Area BACKBONE(0) (Inactive)
        Number of interfaces in this area is 3
        Area has no authentication
        SPF algorithm last executed 00:52:42.956 ago
        SPF algorithm executed 9 times
        Area ranges are
        Number of LSA 1. Checksum Sum 0x00C728
        Number of opaque link LSA 0. Checksum Sum 0x000000
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0

*May 29 00:01:25.679: %OSPF-4-DUP_RTRID_NBR: OSPF detected duplicate router-id
1.1.1.1 from 10.1.1.3 on interface GigabitEthernet0/0
```

First, focus on the problem: the duplicate RIDs. The first line of the **show ip ospf** command on the two routers quickly shows the duplicate use of 1.1.1.1. To solve the problem, assuming R1 should use 1.1.1.1 and R3 should use another RID (maybe 3.3.3.3), change the RID on R3, and restart the OSPF process. To do so, use the **router-id 3.3.3.3** OSPF subcommand and use the EXEC mode command **clear ip ospf process**.

Also, take a moment to read over the log message generated on each router when a duplicate RID exists.

Finally, note that the **show ip ospf** commands in Example Q-12 also show a common false positive for a root cause of OSPF neighbor problems. OSPF PIDs—the number of the **router ospf** command—do not have to match. Note that in Example Q-12 that same first line of output shows that R3 uses the **router ospf 3** command, per the phrase "Process ospf 3," whereas R1 uses the **router ospf 1** command, as noted with the phrase "Process ospf 1." These mismatched numbers are not a problem.

Finding OSPF Hello and Dead Timer Mismatches

Finally, consider the problem created on R4, with the configuration of a different Hello timer and dead timer as compared with the default settings on R1, R2, and R3. Whereas EIGRP allows neighbors to use a different Hello timer, OSPF does not, so this mismatch prevents R4 from becoming neighbors with any of the other three OSPF routers.

Example Q-13 shows the easiest way to find the mismatch, using the **show ip ospf interface** command on both R1 and R4. This command lists the Hello and dead timers for each interface, as highlighted in the example. Note that R1 uses 10 and 40 (Hello and dead), whereas R4 uses 5 and 20.

Example Q-13 *Finding Mismatched Hello/Dead Timers*

```
R1# show ip ospf interface G0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet Address 10.1.1.1/24, Area 0, Attached via Network Statement
  Process ID 1, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 1
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           1         no          no            Base
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 1.1.1.1, Interface address 10.1.1.1
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
! lines omitted for brevity
```

```
! Moving on to R4 next
!
R4# show ip ospf interface Gi0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet Address 10.1.1.4/24, Area 0, Attached via Network Statement
  Process ID 4, Router ID 10.1.44.4, Network Type BROADCAST, Cost: 1
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           1         no          no            Base
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 10.1.44.4, Interface address 10.1.1.4
  No backup designated router on this network
  Timer intervals configured, Hello 5, Dead 20, Wait 20, Retransmit 5
! lines omitted for brevity
```

The **debug ip ospf hello** command can also uncover this problem because it lists a message for each Hello that reveals the Hello/dead timer mismatch, as shown in Example Q-14.

Example Q-14 *Finding Mismatched Hello/Dead Timers with* **debug**

```
R1# debug ip ospf hello
OSPF hello events debugging is on
R1#
*Nov 15 14:05:10.616: OSPF-1 HELLO Gi0/0: Rcv hello from 10.1.44.4 area 0 10.1.1.4
*Nov 15 14:05:10.616: OSPF-1 HELLO Gi0/0: Mismatched hello parameters from 10.1.1.4
*Nov 15 14:05:10.616: OSPF-1 HELLO Gi0/0: Dead R 20 C 40, Hello R 5 C 10 Mask R
  255.255.255.0 C 255.255.255.0
```

Although debug messages can be a little difficult to understand, a few comments make the meaning of these messages much clearer. The highlighted message uses a *C* to mean "configured value"—in other words, the value on the local router, or R1 in this case. The *R* in the message means "received value," or the value listed in the received Hello. In this case

- "Dead R 20 C 40" means that R1 received a Hello with a dead timer set to 20, while R1's configured value is set to 40.

- "Hello R 5 C 10" means that R1 received a Hello with the Hello timer set to 5, while R1's configured value is set to 10.

Note that any IP subnet mismatch problems could also be found with this same debug, based on the received and configured subnet masks.

Other OSPF Issues

This last short discussion in this chapter looks at these two additional topics: shutting down the routing protocol process and the interface maximum transmission unit (MTU) size.

Shutting Down the OSPF Process

Cisco uses the IOS **shutdown** command in several contexts. You can use the **shutdown** command in interface configuration mode to disable the interface so that it no longer sends and receives packets. Cisco IOS switches allow the **shutdown** command in VLAN configuration mode, causing the switch to stop forwarding frames in that VLAN. In both cases, the **shutdown** command does not remove any configuration; it simply causes IOS to stop a particular function. Then, the **no shutdown** command in the same command mode re-enables that function.

IOS allows both the OSPFv2 and EIGRP routing protocol processes to be disabled and enabled with the **shutdown** and **no shutdown** commands, respectively, in routing protocol configuration mode. When a routing protocol process is shut down, IOS

- Brings down any existing neighbor relationships
- Does not form new neighbor relationships
- Quits sending Hello messages
- Does not remove routing protocol configuration

Basically, shutting down the routing protocol process gives the network engineer a way to stop using the routing protocol on that router, without having to remove all the configuration.

From a troubleshooting perspective, on the exam, what would you expect to see if a small design was configured perfectly, except that one router's OSPF process was shut down? First, the router with the shutdown routing protocol process would not have any OSPF neighbors, and other routers would not list that router as a neighbor. But because the OSPF **shutdown** subcommand does not remove any configuration, the **show ip ospf interfaces** command still shows evidence that OSPF is configured on the interfaces.

Example Q-15 shows an example on Router R5, as shown in Figure Q-7. R5 is a different router than the one used in earlier examples, but it begins the example with two OSPF neighbors, R2 and R3, with router IDs 2.2.2.2 and 3.3.3.3. The example shows the OSPF process being shut down, the neighbors failing, and those two key OSPF **show** commands: **show ip ospf neighbor** and **show ip ospf interface brief**.

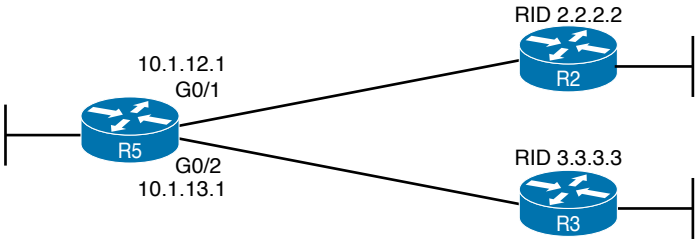

Figure Q-7 *Example Network to Demonstrate OSPF Process Shutdown*

Example Q-15 *Shutting Down an OSPF Process, and the Resulting Neighbor States*

```
R5# show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
2.2.2.2           1   FULL/DR         00:00:35    10.1.12.2       GigabitEthernet0/1
3.3.3.3           1   FULL/DR         00:00:33    10.1.13.3       GigabitEthernet0/2
R5# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)# router ospf 1
R5(config-router)# shutdown
R5(config-router)# ^Z
R5#
*Mar 23 12:43:30.634: %OSPF-5-ADJCHG: Process 1, Nbr 2.2.2.2 on GigabitEthernet0/1
  from FULL to DOWN, Neighbor Down: Interface down or detached
*Mar 23 12:43:30.635: %OSPF-5-ADJCHG: Process 1, Nbr 3.3.3.3 on GigabitEthernet0/2
  from FULL to DOWN, Neighbor Down: Interface down or detached
R5#
R5# show ip ospf neighbor
R5#
R5# show ip ospf interface brief
Interface    PID   Area            IP Address/Mask    Cost  State Nbrs F/C
Gi0/1        1     0               10.1.12.1/24       1     DOWN  0/0
Gi0/2        1     0               10.1.13.1/24       1     DOWN  0/0
```

The two **show** commands point out a couple of particularly important facts. First, before the **shutdown**, the **show ip ospf neighbor** command lists two neighbors. After the **shutdown**, the same command lists no neighbors at all. Second, the **show ip ospf interface brief** command does list the interfaces on which OSPF is enabled, on the local router's own IP addresses. However, it lists a state of DOWN, which is a reference to the neighbor's state.

Mismatched MTU Settings

The MTU size defines a per-interface setting used by the router for its Layer 3 forwarding logic, defining the largest network layer packet that the router will forward out each interface. For instance, the IPv4 MTU size of an interface defines the maximum size IPv4 packet that the router can forward out an interface.

Routers often use a default MTU size of 1500 bytes, with the ability to set the value as well. The **ip mtu** *size* interface subcommand defines the IPv4 MTU setting, and the **ipv6 mtu** *size* command sets the equivalent for IPv6 packets.

In an odd twist, two OSPFv2 routers can actually become OSPF neighbors, and reach 2-way state, even if they happen to use different IPv4 MTU settings on their interfaces. However, they fail to exchange their LSDBs. Eventually, after trying and failing to exchange their LSDBs, the neighbor relationship also fails.

The concepts behind what happens with an MTU mismatch work the same with both OSPFv2 and OSPFv3.

Command References

Tables Q-5, Q-6, and Q-7 list configuration, verification, and debug 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 Q-5 Appendix Q Configuration Command Reference

Command	Description
ip hello-interval eigrp *as-number timer-value*	Interface subcommand that sets the EIGRP Hello interval for that EIGRP process
ip hold-time eigrp *as-number seconds*	Interface subcommand that sets the EIGRP hold time for the interface
ip ospf hello-interval *seconds*	Interface subcommand that sets the interval for periodic Hellos
ip ospf dead-interval *number*	Interface subcommand that sets the OSPF dead timer
passive-interface *type number*	Router subcommand, for both OSPF and EIGRP that tells the routing protocol to stop sending Hellos and stop trying to discover neighbors on that interface

Table Q-6 Appendix Q **show** Command Reference

Command	Description
show ip protocols	Shows routing protocol parameters and current timer values, including an effective copy of the routing protocols' **network** commands and a list of passive interfaces
show ip eigrp interfaces	Lists the interfaces on which EIGRP has been enabled for each EIGRP process, except passive interfaces
show ip route eigrp	Lists only EIGRP-learned routes from the routing table
show ip eigrp neighbors	Lists EIGRP neighbors and status
show ip ospf interface brief	Lists the interfaces on which the OSPF protocol is enabled (based on the **network** commands), including passive interfaces
show ip ospf interface [*type number*]	Lists detailed OSPF settings for all interfaces, or the listed interface, including Hello and dead timers and OSPF area
show ip route ospf	Lists routes in the routing table learned by OSPF

Command	Description
show ip ospf neighbor	Lists neighbors and current status with neighbors, per interface
show ip ospf	Lists a group of messages about the OSPF process itself, listing the OSPF Router ID in the first line
show interfaces	Lists a long set of messages, per interface, that lists configuration, state, and counter information
show interfaces description	Lists one line of output per interface with brief status information

Table Q-7 Appendix Q **debug** Command Reference

Command	Description
debug eigrp packets	Lists log messages for EIGRP packets that flow in and out of the router
debug ip ospf adj	Issues log messages for adjacency events, meaning events related to routers becoming neighbors
debug ip ospf events	Issues log messages for each action taken by OSPF, including the receipt of messages
debug ip ospf packet	Issues log messages describing the contents of all OSPF packets
debug ip ospf hello	Issues log messages describing Hellos and Hello failures
undebug all	EXEC command used to disable all current debugs

APPENDIX R

Exam Topics Cross Reference

This appendix lists the exam topics associated with the CCNA 200-301 exam. Cisco lists the exam topics on its website. Even though changes to the exam topics are rare, you should always review those exam topics for any updates; check www.cisco.com/go/certifications and navigate to the correct exam.

Cisco organizes each list of exam topics by domains, which are major topic areas. Cisco states the percentage of the exam that should come from each domain, so you get some idea of the areas of importance. Traditionally, the score report you receive after taking the exam shows your percentage score in each domain.

This appendix includes two separate types of indices to exam topics:

- **CCNA 200-301 Exam Topic Order:** This section lists the CCNA 200-301 exam topics in the same order Cisco lists them on its website, with a list of associated book chapters. This first list shows a cross reference from each exam topic to the chapters that include at least some material about each topic.
- **Book Chapter Order Versus CCNA 200-301 Exam Topics:** This lists the same CCNA 200-301 exam topics but indexed by chapter instead of exam topic. This section lists the chapters in this book, along with the exam topics that the chapter includes. This section basically relists the kind of information found on the first page of each chapter, just in condensed form in one place.

CCNA 200-301 Exam Topic Order

The CCNA 200-301 exam includes six major topic areas (domains), each with a percentage listed. Table R-1 lists the domains and their percentages.

Table R-1 CCNA 200-301 Exam Topic Domains

Domain	Percentage
Domain 1: Network Fundamentals	20%
Domain 2: Network Access	20%
Domain 3: IP Connectivity	25%
Domain 4: IP Services	10%
Domain 5: Security Fundamentals	15%
Domain 6: Automation and Programmability	10%

Tables R-2 through R-7 list the exam topics within each of the six domains. Note that the *CCNA 200-301 Official Cert Guide, Volume 2*, covers some of the exam topics. These tables show where this book explains exam topics. Exam topics with no chapter listed are covered in Volume 2 only.

Table R-2 CCNA 200-301 Domain 1 Exam Topics (Network Fundamentals)

Exam Topic	Chapter(s)
1.1 Explain the role of network components	2, 3, 5, 7, 26
1.1.a Routers	3, 15
1.1.b L2 and L3 switches	2, 5, 7
1.1.c Next-generation firewalls and IPS	
1.1.d Access points	26
1.1.e Controllers (Cisco DNA Center and WLC)	29
1.1.f Endpoints	
1.1.g Servers	
1.2 Describe characteristics of network topology architectures	2, 3
1.2.a 2 tier	
1.2.b 3 tier	
1.2.c Spine-leaf	
1.2.d WAN	3
1.2.e Small office/home office (SOHO)	2, 15
1.2.f On-premises and cloud	
1.3 Compare physical interface and cabling types	1, 2
1.3.a Single-mode fiber, multimode fiber, copper	1, 2
1.3.b Connections (Ethernet shared media and point-to-point)	1, 2
1.3.c Concepts of PoE	
1.4 Identify interface and cable issues (collisions, errors, mismatch duplex, and/or speed)	7
1.5 Compare TCP to UDP	
1.6 Configure and verify IPv4 addressing and subnetting	6, 11, 12, 13, 14, 15, 17, 18, 22
1.7 Describe the need for private IPv4 addressing	11, 16
1.8 Configure and verify IPv6 addressing and prefix	23, 24
1.9 Compare IPv6 address types	23, 24

Exam Topic	Chapter(s)
1.9.a Global unicast	23, 24
1.9.b Unique local	23, 24
1.9.c Link local	24
1.9.d Anycast	24
1.9.e Multicast	24
1.9.f Modified EUI 64	24
1.10 Verify IP parameters for Client OS (Windows, Mac OS, Linux)	
1.11 Describe wireless principles	26
1.11.a Nonoverlapping Wi-Fi channels	26
1.11.b SSID	26
1.11.c RF	26
1.11.d Encryption	28
1.12 Explain virtualization fundamentals (virtual machines)	
1.13 Describe switching concepts	5, 8
1.13.a MAC learning and aging	5, 8
1.13.b Frame switching	5, 8
1.13.c Frame flooding	5, 8
1.13.d MAC address table	5, 8

Table R-3 CCNA 200-301 Domain 2 Exam Topics (Network Access)

Exam Topic	Chapter(s)
2.1 Configure and verify VLANs (normal range) spanning multiple switches	8
2.1.a Access ports (data and voice)	8
2.1.b Default VLAN	8
2.1.c Connectivity	8
2.2 Configure and verify interswitch connectivity	8
2.2.a Trunk ports	8
2.2.b 802.1Q	8
2.2.c Native VLAN	8
2.3 Configure and verify Layer 2 discovery protocols (Cisco Discovery Protocol and LLDP)	
2.4 Configure and verify (Layer 2/Layer 3) EtherChannel (LACP)	8, 9, 10, 17

Exam Topic	Chapter(s)
2.5 Describe the need for and basic operations of Rapid PVST+ Spanning Tree Protocol and identify basic operations	5, 9, 10
2.5.a Root port, root bridge (primary/secondary), and other port names	9, 10
2.5.b Port states (forwarding/blocking)	9, 10
2.5.c PortFast benefits	9, 10
2.6 Compare Cisco Wireless Architectures and AP modes	27
2.7 Describe physical infrastructure connections of WLAN components (AP, WLC, access/trunk ports, and LAG)	29
2.8 Describe AP and WLC management access connections (Telnet, SSH, HTTP, HTTPS, console, and TACACS+/RADIUS)	29
2.9 Configure the components of a wireless LAN access for client connectivity using GUI only such as WLAN creation, security settings, QoS profiles, and advanced WLAN settings	29

Table R-4 CCNA 200-301 Domain 3 Exam Topics (IP Connectivity)

Exam Topic	Chapter(s)
3.1 Interpret the components of routing table	16
3.1.a Routing protocol code	16
3.1.b Prefix	16
3.1.c Network mask	16
3.1.d Next hop	16
3.1.e Administrative distance	16
3.1.f Metric	16
3.1.g Gateway of last resort	16
3.2 Determine how a router makes a forwarding decision by default	16
3.2.a Longest match	16
3.2.b Administrative distance	16, 19, 20
3.2.c Routing protocol metric	19, 20
3.3 Configure and verify IPv4 and IPv6 static routing	16, 18, 25
3.3.a Default route	16, 18, 25
3.3.b Network route	16, 18, 25
3.3.c Host route	16, 18, 25
3.3.d Floating static	16, 18, 25

Exam Topic	Chapter(s)
3.4 Configure and verify single area OSPFv2	19, 20, 21
3.4.a Neighbor adjacencies	19, 20, 21
3.4.b Point-to-point	19, 20, 21
3.4.c Broadcast (DR/BDR selection)	19, 20, 21
3.4.d Router ID	19, 20, 21
3.5 Describe the purpose of first hop redundancy protocol	

Table R-5 CCNA 200-301 Domain 4 Exam Topics (IP Services)

Exam Topics	Chapter(s)
4.1 Configure and verify inside source NAT using static and pools	
4.2 Configure and verify NTP operating in a client and server mode	
4.3 Explain the role of DHCP and DNS within the network	
4.4 Explain the function of SNMP in network operations	
4.5 Describe the use of syslog features including facilities and levels	
4.6 Configure and verify DHCP client and relay	6
4.7 Explain the forwarding per-hop behavior (PHB) for QoS such as classification, marking, queuing, congestion, policing, shaping	
4.8 Configure network devices for remote access using SSH	6
4.9 Describe the capabilities and function of TFTP/FTP in the network	

Table R-6 CCNA 200-301 Domain 5 Exam Topics (Security Fundamentals)

Exam Topics	Chapter(s)
5.1 Define key security concepts (threats, vulnerabilities, exploits, and mitigation techniques)	
5.2 Describe security program elements (user awareness, training, and physical access control)	
5.3 Configure device access control using local passwords	6
5.4 Describe security password policies elements, such as management, complexity, and password alternatives (multifactor authentication, certificates, and biometrics)	
5.5 Describe remote access and site-to-site VPNs	
5.6 Configure and verify access control lists	

Exam Topics	Chapter(s)
5.7 Configure Layer 2 security features (DHCP snooping, dynamic ARP inspection, and port security)	
5.8 Differentiate authentication, authorization, and accounting concepts	
5.9 Describe wireless security protocols (WPA, WPA2, and WPA3)	28
5.10 Configure WLAN using WPA2 PSK using the GUI	29

Table R-7 CCNA 200-301 Domain 6 Exam Topics (Programmability and Automation)

Exam Topics	Chapter(s)
6.1 Explain how automation impacts network management	
6.2 Compare traditional networks with controller-based networking	
6.3 Describe controller-based and software defined architectures (overlay, underlay, and fabric)	
6.3.a Separation of control plane and data plane	
6.3.b North-bound and south-bound APIs	
6.4 Compare traditional campus device management with Cisco DNA Center enabled device management	
6.5 Describe characteristics of REST-based APIs (CRUD, HTTP verbs, and data encoding)	
6.6 Recognize the capabilities of configuration management mechanisms Puppet, Chef, and Ansible	
6.7 Interpret JSON encoded data	

Book Chapter Order Versus CCNA 200-301 Exam Topics

Cisco organizes its exam topics based on the outcome of your learning experience, which is typically not a reasonable order for building the content of a book or course. This section lists the book chapters in sequence, with the exam topics covered in each chapter.

Book Chapter	Exam Topics Covered
Part I: Introduction to Networking	
Chapter 1: Introduction to TCP/IP Networking	**1.0 Network Fundamentals** 1.3 Compare physical interface and cabling types *1.3.a Single-mode fiber, multimode fiber, copper* *1.3.b Connections (Ethernet shared media and point-to-point)*

Book Chapter	Exam Topics Covered
Chapter 2: Fundamentals of Ethernet LANs	**1.0 Network Fundamentals** 1.1 Explain the role and function of network components *1.1.b L2 and L3 switches* 1.2 Describe characteristics of network topology architectures *1.2.e Small office/home office (SOHO)* 1.3 Compare physical interface and cabling types *1.3.a Single-mode fiber, multimode fiber, copper* *1.3.b Connections (Ethernet shared media and point-to-point)*
Chapter 3: Fundamentals of WANs and IP Routing	**1.0 Network Fundamentals** 1.1 Explain the role and function of network components *1.1.a Routers* 1.2 Describe characteristics of network topology architectures *1.2.d WAN*
Part II: Implementing Ethernet LANs	
Chapter 4: Using the Command-Line Interface	**None**
Chapter 5: Analyzing Ethernet LAN Switching	**1.0 Network Fundamentals** 1.1 Explain the role and function of network components *1.1.b L2 and L3 switches* 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** 2.5 Describe the need for and basic operations of Rapid PVST+ Spanning Tree Protocol and identify basic operations
Chapter 6: Configuring Basic Switch Management	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting **4.0 IP Services** 4.6 Configure and verify DHCP client and relay 4.8 Configure network devices for remote access using SSH **5.0 Security Fundamentals** 5.3 Configure device access control using local passwords

Book Chapter	Exam Topics Covered
Chapter 7: Configuring and Verifying Switch Interfaces	**1.0 Network Fundamentals** 1.1 Explain the role and function of network components *1.1.b L2 and L3 switches* 1.4 Describe switching concepts
Part III: Implementing VLANs and STP	
Chapter 8: Implementing Ethernet Virtual LANs	**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 Connectivity* 2.2 Configure and verify interswitch connectivity *2.2.a Trunk ports* *2.2.b 802.1Q* *2.2.c Native VLAN*
Chapter 9: Spanning Tree Protocol Concepts	**2.0 Network Access** 2.4 Configure and verify (Layer 2/Layer 3) EtherChannel (LACP) 2.5 Describe the need for and basic operations of Rapid PVST+ Spanning Tree Protocol and identify basic operations *2.5.a Root port, root bridge (primary/secondary), and other port names* *2.5.b Port states (forwarding/blocking)* *2.5.c PortFast benefits*
Chapter 10: RSTP and EtherChannel Configuration	**2.0 Network Access** 2.4 Configure and verify (Layer 2/Layer 3) EtherChannel (LACP) 2.5 Describe the need for and basic operations of Rapid PVST+ Spanning Tree Protocol and identify basic operations *2.5.a Root port, root bridge (primary/secondary), and other port names* *2.5.b Port states (forwarding/blocking)* *2.5.c PortFast benefits*

Book Chapter	Exam Topics Covered
Part IV: IPv4 Addressing	
Chapter 11: Perspectives on IPv4 Subnetting	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting 1.7 Describe the need for private IPv4 addressing
Chapter 12: Analyzing Classful IPv4 Networks	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting
Chapter 13: Analyzing Subnet Masks	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting
Chapter 14: Analyzing Existing Subnets	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting
Part V: IPv4 Routing	
Chapter 15: Operating Cisco Routers	**1.0 Network Fundamentals** 1.1 Explain the role and function of network components *1.1.a Routers* 1.2 Describe characteristics of network topology architectures *1.2.e Small office/home office (SOHO)* 1.6 Configure and verify IPv4 addressing and subnetting
Chapter 16: Configuring IPv4 Addressing and Static Routes	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting **3.0 IP Connectivity** 3.1 Interpret the components of routing table *3.1.a Routing protocol code* *3.1.b Prefix* *3.1.c Network mask* *3.1.d Next hop* *3.1.e Administrative distance* *3.1.f Metric* *3.1.g Gateway of last resort* 3.2 Determine how a router makes a forwarding decision by default *3.2.a Longest match* *3.2.b Administrative distance* 3.3 Configure and verify IPv4 and IPv6 static routing *3.3.a Default route* *3.3.b Network route* *3.3.c Host route* *3.3.d Floating static*

Book Chapter	Exam Topics Covered
Chapter 17: IP Routing in the LAN	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting **2.0 Network Access** 2.4 Configure and verify (Layer 2/Layer 3) EtherChannel (LACP)
Chapter 18: Troubleshooting IPv4 Routing	**1.0 Network Fundamentals** 1.6 Configure and verify IPv4 addressing and subnetting **3.0 IP Connectivity** 3.3 Configure and verify IPv4 and IPv6 static routing *3.3.a Default route* *3.3.b Network route* *3.3.c Host route* *3.3.d Floating static*
Part VI: OSPF	
Chapter 19: Understanding OSPF Concepts	**3.0 IP Connectivity** 3.2 Determine how a router makes a forwarding decision by default *3.2.b Administrative distance* *3.2.c Routing protocol metric* 3.4 Configure and verify single area OSPFv2 *3.4.a Neighbor adjacencies* *3.4.b Point-to-point* *3.4.c Broadcast (DR/BR selection)* *3.4.d (Router ID)*
Chapter 20: Implementing OSPF	**3.0 IP Connectivity** 3.2 Determine how a router makes a forwarding decision by default *3.2.b Administrative distance* *3.2.c Routing protocol metric* 3.4 Configure and verify single area OSPFv2 *3.4.a Neighbor adjacencies* *3.4.b Point-to-point* *3.4.c Broadcast (DR/BR selection)* *3.4.d (Router ID)*
Chapter 21: OSPF Network Types and Neighbors	**3.0 IP Connectivity** 3.4 Configure and verify single area OSPFv2 *3.4.a Neighbor adjacencies* *3.4.b Point-to-point* *3.4.c Broadcast (DR/BR selection)* *3.4.d (Router ID)*

Book Chapter	Exam Topics Covered
Part VII: IP Version 6	
Chapter 22: Fundamentals of IP Version 6	**1.0 Network Fundamentals** 1.8 Configure and verify IPv6 addressing and prefix
Chapter 23: IPv6 Addressing and Subnetting	**1.0 Network Fundamentals** 1.8 Configure and verify IPv6 addressing and prefix 1.9 Compare and contrast IPv6 address types *1.9.a Global unicast* *1.9.b Unique local*
Chapter 24: Implementing IPv6 Addressing on Routers	**1.0 Network Fundamentals** 1.8 Configure and verify IPv6 addressing and prefix 1.9 Compare and contrast IPv6 address types *1.9.a Global unicast* *1.9.b Unique local* *1.9.c Link local* *1.9.d Anycast* *1.9.e Multicast* *1.9.f Modified EUI 64*
Chapter 25: Implementing IPv6 Routing	**3.0 IP Connectivity** 3.3 Configure and verify IPv4 and IPv6 static routing *3.3.a Default route* *3.3.b Network route* *3.3.c Host route* *3.3.d Floating static*
Part VIII: Wireless LANs	
Chapter 26: Fundamentals of Wireless Networks	**1.0 Network Fundamentals** 1.1 Explain the role and function of network components *1.1.d Access Points* 1.11 Describe wireless principles *1.11.a Nonoverlapping Wi-Fi Channels* *1.11.b SSID* *1.11.c RF*
Chapter 27: Analyzing Cisco Wireless Architectures	**2.0 Network Access** 2.6 Compare Cisco Wireless Architectures and AP modes

Book Chapter	Exam Topics Covered
Chapter 28: Securing Wireless Networks	**1.0 Network Fundamentals** 1.11 Describe wireless principles 1.11.d Encryption **5.0 Security Fundamentals** 5.9 Describe wireless security protocols (WPA, WPA2, and WPA3)
Chapter 29: Building a Wireless LAN	**2.0 Network Access** 2.7 Describe physical infrastructure connections of WLAN components (AP, WLC, access/trunk ports, and LAG) 2.8 Describe AP and WLC management access connections (Telnet, SSH, HTTP, HTTPS, console, and TACACS+/RADIUS) 2.9 Configure the components of a wireless LAN access for client connectivity using GUI only such as WLAN creation, security settings, QoS profiles, and advanced WLAN settings **5.0 Security Fundamentals** 5.10 Configure WLAN using WPA2 PSK using the GUI

Pearson

Where are the companion content files?

Thank you for purchasing this Premium Edition version of CCNA 200-301 Official Cert Guide, Volume 1

This product comes with companion content. You have access to these files by following the steps below:

1. Go to **ciscopress.com/account** and log in.
2. Click on the "Access Bonus Content" link in the Registered Products section of your account page for this product, to be taken to the page where your downloadable content is available.

Please note that many of our companion content files can be very large, especially image and video files.

If you are unable to locate the files for this title by following the steps at left, please visit **ciscopress.com/support** and select the chat, phone, or web ticket options to get help from a tech support representative.
