CCIE Enterprise Infrastructure Foundation

Narbik Kocharians

Cisco Press

CCIE Enterprise Infrastructure Foundation

Narbik Kocharians

Copyright© 2023 Pearson Education, Inc.

Published by: Cisco Press

ScoutAutomatedPrintCode

Library of Congress Control Number: 2022902108

ISBN-13: 978-0-13-737424-3

ISBN-10: 0-13-737424-0

Warning and Disclaimer

Trademark Acknowledgments

Special Sales

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers’ feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Editor-in-Chief: Mark Taub

Alliances Manager, Cisco Press: Arezou Gol

Director, ITP Product Management: Brett Bartow

Executive Editor: James Manly

Managing Editor: Sandra Schroeder

Development Editor: Ellie Bru

Project Editor: Mandie Frank

Copy Editor: Kitty Wilson

Technical Editors: Sarah Anand; Dante McNeil

Editorial Assistant: Cindy Teeters

Designer: Chuti Prasertsith

Composition: codeMantra

Indexer: Timothy Wright

Proofreader: Donna E. Mulder

Credits

Unnumbered figures on pages 860-861	PuTTY

About the Author

Narbik Kocharians, CCIE No. 12410 (Routing and Switching, Service Provider, and Security) is a triple CCIE with more than 46 years of experience in this industry. He has designed, implemented, and supported numerous small, mid-size, and large enterprise networks.

Narbik is the president of Micronics Networking and Training, Inc. (www.MicronicsTraining.com), where almost all Cisco-authorized and custom courses are conducted, including CCIE-DC, CCIE-SP, CCIE-Enterprise Infrastructure, CCDE, ACI, and many more.

About the Technical Reviewers

Sarah Anand has been affiliated with different networking technologies, including Cisco-specific implementations, for 7 years, with a focus on routing/switching and service provider technologies. Currently she works as a technical writer and editor, training network engineers in vendor-specific and industry-standard technologies. She has a degree in computer science and enjoys spending free time exploring passions in web design and search engine optimization.

Dante McNeil has 10 years of IT networking experience in the nonprofit, enterprise, K–12, and higher education spaces, with a focus on advanced networking implementations and Cisco technologies. He also spends time writing and creating network training content for networking engineers. He holds a bachelor of science degree in computing information sciences from Jacksonville University. In his spare time, he enjoys roller coasters, video games, and road trips.

Dedications

I would like to dedicate this book to my beautiful wife, Janet, my children and their spouses, Chris and Nona (aka Siroon Achik), Patrick and Diana (aka Bestelik Jan), Alexandra (aka Achiko) and Sevak, and Daniel (aka Chompolik), as well as our first grandson, Matthew (aka Jigar), whom I LOVE so much, he brightens my day every morning!

I would like to acknowledge with gratitude the support, sacrifice, and love of my family for making this book possible. I thank God for the health and wisdom that He has instilled in me, my lovely family, my first grandson Mathew, and my father, who was my best friend.

Acknowledgments

A very special thanks to James and Eleanor. I remember brainstorming with James for hours about this book, and eventually he came up with the ultimate solution. I would like to thank Eleanor for having a tremendous amount of patience and professionalism.

I would also like to thank my tech editors, Sarah Anand and Dante McNeil, two gifted network engineers with a tremendous amount of knowledge. God willing, I will be working with these two champions for a long time to come. They are not CCIEs yet, but their knowledge is on par with the best CCIEs out there.

Contents at a Glance

Reader Services

Contents

Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italic* indicates arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets ([ ]) indicate an optional element.
- Braces ({ }) indicate a required choice.
- Braces within brackets ([{ }]) indicate a required choice within an optional element.

Introduction

Enterprise networking has undergone many small changes over the years, building from simple shared bus LANs to intricate routing and switching architectures and wireless communications. Behind all of this is a need to ensure high reliability, agility, and speed. Through the decades, many different networking technologies, from physical connections to software protocols, have been created to assist enterprise networks in reaching those goals. For seasoned networking veterans, working with the various protocols and architectures is second nature. However, those who are just starting to build their careers and trying to study more advanced areas of network engineering may be overwhelmed by the multitude of routing protocols, Layer 2 features, and new buzzwords like "software-defined."

This book is written as a foundation guide for the most common enterprise networking concepts that are required for a network engineer looking to move forward to more advanced aspects of networking. It combines aspects of theory instruction with practical application. Topics such as LAN switching, IP routing, and overlay networking technologies such as DMVPN are explained as foundational topics, including examples. Each chapter also functions as a lab manual with a task-oriented structure. Lab scenarios are presented as either configuration objectives, troubleshooting scenarios, or design scenarios. Each lab scenario includes full solutions and explanations. For beginner to intermediate readers, the solutions can be read while solving the tasks. Advanced readers can challenge their knowledge and skills by solving tasks first and then comparing their solutions to the ones provided in this book.

This book is not meant to be an exhaustive study of all the included technologies. It is meant to provide enough information on all topics to allow you to speak intelligently about each technology and even implement some of the configurations, if necessary, in your own environment. It takes topics from Cisco's CCIE Enterprise Infrastructure certification blueprint but includes some legacy topics, where necessary, to facilitate understanding.

Who This Book Is For

Although the title of this book is *CCIE Enterprise Infrastructure Foundation*, the target audience is not limited to just those seeking expert-level certification. Any person looking to learn a little bit more about these foundational technologies will find this book very accessible.

This book breaks down complicated topics and provides examples to maximize understanding. It does, however, assume some basic networking knowledge. The following types of readers will get the most out of this book:

- Those who have completed CCNA certification and are part of the way through their preparation for CCNP Enterprise certification
- Those who have completed CCNP Enterprise certification and are pursuing CCIE
- Those who are currently working in an environment that is implementing specific technologies covered in this book
- Those who are migrating from another vendor to a Cisco environment and need to understand Cisco configurations for common networking protocols

How This Book Is Organized

This book is divided into the 11 chapters described here. Every chapter can stand alone and can be used as a reference for the technologies it covers.

Chapter 1: Switching

Chapter 1 introduces Layer 2 concepts such as preventing loops with Spanning Tree Protocol, segmenting with VLANs, extending VLANs between switches through trunking, and bonding multiple Ethernet links together to increase bandwidth between network nodes. It covers topics such as Spanning Tree Protocol, RSTP, MSTP, VTP and VTP pruning, 802.1Q and ISL trunking, and LACP and PAgP.

Chapter 2: IP Prefix Lists

Chapter 2 introduces a common route filtering mechanism known as a prefix list. It explains why prefix lists were invented and why they are used over access lists for route filtering. This chapter shows how to write prefix lists and apply them in various routing protocols for filtering routes.

Chapter 3: RIPv2

Chapter 3 introduces Routing Information Protocol (RIP). RIP may not be included on the exam, but it is a perfect example of a simple distance vector routing protocol that follows all the standard distance vector designs. It focuses on the simplicity of RIP configuration, advanced RIP filtering scenarios, and RIP configuration challenges.

Chapter 4: EIGRP

Chapter 4 focuses on Cisco's improvement on its own version of Interior Gateway Routing Protocol (IGRP), Enhanced Interior Gateway Routing Protocol (EIGRP). It introduces EIGRP as a distance vector protocol that forms neighbor relationships and keeps a topology table like some other protocols. EIGRP is considered an advanced distance vector protocol that uses more than simple hop counts to learn loop-free paths through a network. This chapter covers EIGRP configuration topics such as EIGRP classic and address family configuration, EIGRP stub routing, and EIGRP with BFD.

Chapter 5: OSPF

Chapter 5 introduces the Open Shortest Path First (OSPF) routing protocol. It begins with an analysis of how OSPF builds its link-state database (LSDB) with various linkstate advertisements (LSA) and uses that information to calculate loop-free routed paths through a network. This chapter also details multiarea OSPF design, filtering, and virtual links. It includes a detailed walkthrough on OSPF's best-path determination to help you understand OSPF's path selection process.

Chapter 6: BGP

Chapter 6 introduces Border Gateway Protocol (BGP), the protocol that routes the Internet. It explains BGP operation between autonomous systems (external BGP, or eBGP) and within a single autonomous system (internal BGP, or iBGP). Topics covered include BGP session establishment, route reflectors and confederations, aggregation, and filtering. This chapter includes a detailed walkthrough of the BGP best-path determination process.

Chapter 7: DMVPN

Chapter 7 focuses on Cisco's original SD-WAN technology, known as Dynamic Multipoint VPN (DMVPN). It explains DMVPN from the ground up, introducing concepts such as overlay and underlay networking, the link between DMVPN and NHRP, DMVPN routing using common routing protocols, and different DMVPN designs. It covers DMVPN Phase 1 through Phase 3 configurations, NHRP shortcut switching enhancements, hub-and-spoke networking designs, and (m)GRE tunnels.

Chapter 8: MPLS and L3VPNs

Chapter 8 introduces Multiprotocol Label Switching (MPLS) and the suite of services MPLS can provide. This chapter begins with an introduction to MPLS labels and Label Distribution Protocol (LDP). It also introduces the most common MPLS service, MPLS Layer 3 VPN (L3VPN). Topics covered include CE and PE routers, MPLS core configuration, LDP session establishment, BGP route targets and route distinguishers, and exchange of IGP routes between two sites connected by an MPLS L3VPN.

Chapter 9: IPv6

Chapter 9 introduces Internet Protocol Version 6 (IPv6), which is the successor to IPv4 due to its massive address space. It also details IPv6 address types, assignment, and configuration. Topics covered include IPv6 NDP, IPv6 SLAAC, DMVPN for IPv6, OSPF for IPv6 (OSPFv3), EIGRP for IPv6, and BGP for IPv6.

Chapter 10: SD-WAN

Chapter 10 introduces Cisco's new SD-WAN platform, which is based on its acquisition of Viptela. This chapter details basic SD-WAN components, such as vSmart, vManage, and vBond, as well as the setup and configuration required to join vEdge routers to an SD-WAN solution. Topics covered include onboarding WAN edge devices, unicast routing, segmentation, vManage device templates, ZTP, and application-aware policies.

Chapter 11: SD-Access

Chapter 11 introduces Cisco's SD-Access solution for creating scalable, automated, and resilient enterprise fabric. This chapter covers configuration of the SD-Access policy engine as well as SDA design and implementation. Topics covered include Cisco ISE, pxGrid, XMPP, SDA hierarchy global IP pools, DNAC, and LAN automation.

Chapter 1

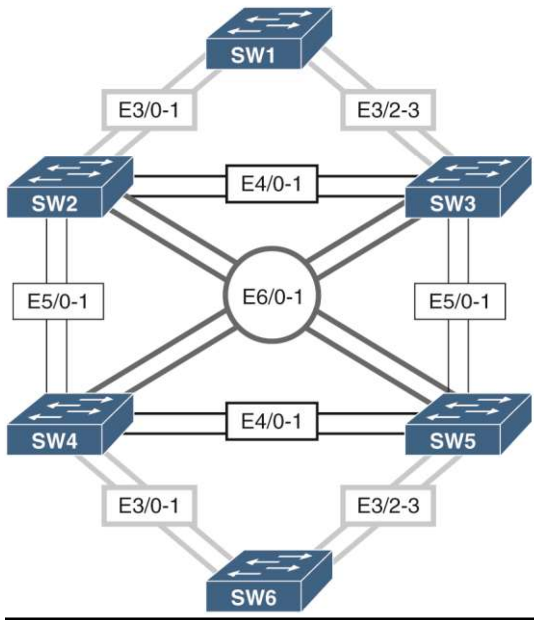
CCIE
Enterprise Infrastructure
Foundation
v1.0

www.MicronicsTraining.com

Narbik Kocharians
CCSI, CCIE #12410
R&S, Security, SP

Switching

Lab 1
Configuring Trunks



What is covered in this lab:

This lab focuses mainly on configuring trunk links and VLANs and controlling which VLANs are allowed on a particular trunk link. This lab covers the following topics: dynamic trunks, basic VTP operation, how to modify the allowed VLAN lists on trunk links, and VTP pruning.

Task 1

Shut down all ports on all six switches and configure the VTP domain name to be **TST**.

When receiving a switch and beginning the configuration process for the first time, it is always a best practice to shut down all ports that are connected to neighboring switches or end devices. Doing so prevents potential catastrophic harm to the network during the configuration change should an accidental misconfiguration occur. Such misconfigurations could introduce Layer 2 Bridging loops or lost VLAN configuration information.

Once the configuration changes have been made and verified, it is then safe to bring the ports up and verify once again that the desired change has been met. This task starts off by shutting down all ports on all six switches. Normally, such a task would be very time consuming. Each interface would have to be manually configured with the shutdown command one-by-one. However, Cisco IOS switches have what is known as the interface range command that allows selection of multiple interfaces at once for configuration. This and future tasks make full use of this command to shut down the interfaces on the switch.

The next part of the task begins the configuration for the VTP settings for the switched domain. VTP is the VLAN Trunking Protocol developed by Cisco. It was developed to allow switches to dynamically synchronize the contents of their VLAN databases with each other to ease VLAN configuration in the network. VTP switches that synchronize their VLAN databases belong to the same VTP domain which can be manually configured. VTP is automatically enabled on all trunk ports in the topology with some default settings.

The following "show vtp status" output reveals some of the default settings the switch uses for VTP:

On Any Switch:

```
Switch#show vtp status

VTP Version capable                : 1 to 3
VTP version running                : 1
VTP Domain Name                    :
VTP Pruning Mode                   : Disabled
VTP Traps Generation               : Disabled
Device ID                          : aabb.cc80.0e00
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode                   : Server
Maximum VLANs supported locally      : 1005
Number of existing VLANs             : 5
Configuration Revision               : 0
MD5 digest                         : 0x57 0xCD 0x40 0x65 0x63 0x59 0x47 0xBD
                                     0x56 0x9D 0x4A 0x3E 0xA5 0x69 0x35 0xBC
```

Notice the VTP domain name in the above output is empty. An empty or blank VTP domain name means that the switch currently does not belong to any VTP domain. In addition, the switch is operating as a VTP server. This is the default mode on Cisco IOS. In this mode, the devices advertise the contents of their VLAN databases to neighboring VTP switches. Addtionally, a device in this mode can create, modify and delete VLANs.

By default, all switches belong to the {NULL} VTP domain. While initialized with a {NULL} VTP Domain value, the switch will not send any of its own VTP advertisements, but will respond and react to received VTP advertisements. The switch will adopt the VTP domain name of any VTP advertisement received as its own VTP domain name and begins sending its own VTP advertisements out of its own trunk links in accordance to its VTP operating mode.

The task specifies that all switches should be manually configured with the VTP domain name of TST. This is configured with the VTP domain name *domain-name* command in global configuration mode. The following is the complete configuration required on all switches to complete this task:

On All Switches:

```
SWx(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-
3,e6/0-3
SWx(config-if-range)#shut

SWx(config)#vtp domain TST
```

You should see the following console message:

```
Changing VTP domain name from NULL to TST
```

To Verify the configuration:

```
Switch#show vtp status

VTP Version capable               : 1 to 3
VTP version running               : 1
VTP Domain Name                   : TST
VTP Pruning Mode                  : Disabled
VTP Traps Generation              : Disabled
Device ID                         : aabb.cc80.0e00
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)
```

```
Feature VLAN:
--------------
VTP Operating Mode                    : Server
Maximum VLANs supported locally       : 1005
Number of existing VLANs              : 5
Configuration Revision                : 0
MD5 digest                           : 0xAD 0x9C 0xE6 0xFA 0x40 0x3D 0x31 0x02
                                       0xFA 0xD9 0xA9 0xE3 0x84 0x71 0x45 0xDD
```

Task 2

Configure the following hostnames:

Second switch: SW2
Third switch: SW3
Fourth switch: SW4
Fifth switch: SW5

On the second switch:

```
SW2(config)#hostname SW2
```

On the third switch:

```
SW3(config)#hostname SW3
```

On the fourth switch:

```
SW4(config)#hostname SW4
```

On the fifth switch:

```
SW5(config)#hostname SW5
```

Task 3

Configure a dot1q trunk between SW3 and SW5 using their E5/0 interfaces, based on the following policy:

SW3, E5/0

This port should be configured into a permanent trunk mode, and it should negotiate to convert the neighboring interface into a trunk.

SW5, E5/0

This port should be configured to actively attempt to convert the link to a trunk. You should not configure **switchport trunk encapsulation dot1q** on this port.

Cisco switch interfaces can operate in Trunk mode, where multiple VLANs can be carried across the link, or Access mode, where only a single VLAN is carried across the link. These modes can be configured in one of two ways:

- **Static configuration**
- **Dynamic configuration**

Static configuration involves manual configuration of the switchport mode command under the interface, explicitly specifying whether the port is in access or trunk mode. Dynamic configuration relies on the Dynamic Trunking Protocol or DTP to determine the trunk status.

DTP isn't limited to only determining the trunk mode, it can also be used to determine the trunk encapsulation. Cisco switches support both the Cisco-proprietary InterSwitch Link (ISL) and IEEE's 802.1q trunk encapsulation styles. DTP makes this determination either automatically or based on the settings specified by the switchport trunk encapsulation command in interface configuration mode.

The two solutions (ISL, 802.1q) take different approaches to encode the VLAN information. Cisco's ISL takes the approach of encapsulating the entire ethernet frame with an ISL header and trailer. Within the ISL header, among other fields, is the VLAN ID field that corresponds to the VLAN the Ethernet frame originated from. The ISL header is 24-bytes in total with a 4-byte CRC trailer used to detect errors in transmission. In total, ISL adds an additional 40 bytes of overhead to the Ethernet frame, but does not require any modification of the original Ethernet frame.

The IEEE 802.1Q encapsulation method isn't really encapsulating the Ethernet frame as ISL does. Instead, it inserts a VLAN "shim" header into the Ethernet frame. The 802.1Q header is greatly simplified from the ISL header containing only Protocol Identifier, Priority, VLAN ID, and Canonical format indicator fields. Of particular note is the VLAN ID field which carries the VLAN ID of the VLAN from which the frame originated. The 802.1Q header is 4-bytes in total and is inserted after the source address field of the original Ethernet header. Since 802.1Q modifies the ethernet header, the switch must recalculate the FCS trailer of the original Ethernet frame to compensate for the additional header length.

Aside from the physical encoding of the Ethernet frame, the method of operation is the same for both protocols. On Cisco switches, interfaces that need to carry VLAN information are called trunk ports. Trunk ports can be dynamically configured or statically assigned. When the switch needs to forward a frame out of a trunk port, it looks at the trunk port's configured VLAN encapsulation method. If it is ISL, the switch computes the ISL header and trailer, encapsulates the Ethernet frame and sends it across the trunk link. If it is 802.1Q, the switch calculates the 802.1Q "shim" header, inserts it into the Ethernet frame, recalculates the Ethernet FCS and forwards the new Ethernet frame out of the trunk link. Because 802.1Q merely inserts a "shim" header, the resulting frame can be referred to as an 802.1Q-tagged frame. This fact is reflected in most non-Cisco hardware where such ports are called "tagged" ports instead of "trunk" ports.

Switches connected by an interswitch link, must be configured to use the same encapsulation method for encoding VLAN membership for ethernet frames, otherwise the link will not function properly. In both cases, the receiving switch examines the ethernet packet, checks the VLAN ID of the ISL or 802.1Q header, removes the VLAN encapsulation (if necessary) and forwards the bare Ethernet frame out the appropriate ports only if they are a member of the same VLAN as indicated in the VLAN ID field.

This task states that an 802.1q trunk should be configured between SW3 and SW5's E5/0 interfaces. The subtask requirements state that SW3's E5/0 interface should permanently be in a trunk mode and should cause the neighboring interface to convert into a trunk mode. The task is completed by first setting SW3's E5/0 interface to use 802.1q trunk encapsulation using the switchport trunk encapsulation dot1q interface-level configuration command. Then, the interface is set to be a static trunk port using the switchport mode trunk interface-level configuration command, as seen below:

On SW3:

```
Switch(config)#interface e5/0
Switch(config-if)#switchport trunk encapsulation dot1q
Switch(config-if)#switchport mode trunk
Switch(config-if)#no shut
```

NOTE: When statically declaring a port to operate in trunk mode, the trunk encapsulation must be defined with the switchport trunk encapsulation command first. Failing to do so results in the command switchport mode trunk being rejected by IOS. In such a situation, the switch will log the following error message:

```
SW3(config-if)#switchport mode trunk
Command rejected: An interface whose trunk encapsulation is "Auto" can
not be configured to "trunk" mode.
```

As the second part of the task states, SW5's interface should be configured to dynamically negotiate its trunking mode. This is accomplished by enabling DTP using the switchport mode dynamic desirable command show below:

On SW5:

```
SW5(config)#interface e5/0
SW5(config-if)#switchport mode dynamic desirable
SW5(config-if)#no shut
```

NOTE: On some switch platforms, the default port mode is dynamic desirable.

Once configured with the above command, the switch will perform dynamic trunk mode and encapsulation negotiation with the other side of the link using DTP. SW5 will send DTP frames out of its E5/0 interface with the intention of negotiating a trunk link with SW3. The task also restricts the usage of the switchport trunk encapsulation dot1q command on the E5/0 interface. This is because the trick lies within SW3.

SW3, being configured as a static 802.1q trunk, will send its own DTP frame to SW5 that indicates that it would like to participate in an 802.1q trunk as shown in the packet capture below.

--- DTP Packet capture ---

```
Dynamic Trunk Protocol:  (Operating/Administrative): Trunk/On (0x81)
(Operating/Administrative): 802.1Q/802.1Q (0xa5): aa:bb:cc:00:0b:05
    Version: 1
    Domain
    Trunk Status
        Type: Trunk Status (0x0002)
        Length: 5
        Value: Trunk/On (0x81)

1... .... = Trunk Operating Status: Trunk (0x1)

.... .001 = Trunk Administrative Status: On (0x1) Trunk Type

        Type: Trunk Type (0x0003)
        Length: 5
        Value: 802.1Q/802.1Q (0xa5)

101. .... = Trunk Operating Type: 802.1Q (0x5) .... .101 = Trunk Administrative Type: 802.1Q
(0x5)

[ -- output omitted for brevity -- ]
```

This happens because static configuration of trunk mode or encapsulation does not disable the DTP on an interface. In other words, the switchport mode trunk command does not disable transmission of DTP frames on a link.

SW5 will use the information from the DTP frame that SW3 sent and set the encapsulation on its E5/0 interface to 802.1q. SW3 is forcing SW5 to negotiate an 802.1q trunk relationship. The following shows the complete configuration commands and show command verification output required to complete the task

On SW3:

```
SW3(config)#interface e5/0
SW3(config-if)#switchport trunk encapsulation dot1q

SW3(config-if)#switchport mode trunk
SW3(config-if)#no shut
```

On SW5:

```
SW5(config)#interface e5/0
SW5(config-if)#switchport mode dynamic desirable
SW5(config-if)#no shut
```

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk | include 802

Et5/0       on           802.1q          trunking      1
```

On SW5:

```
SW5#show interfaces trunk | include 802

Et5/0       desirable    n-802.1q        trunking      1
```

As seen in the show interfaces trunk output above, SW3's E5/0 interface is configured as a static 802.1q trunk whereas E5/0 on SW5 has been negotiated as an 802.1q trunk. This is evidenced by the "desirable" and "n-802.1q" output on SW5. The "desirable" relates to the dynamic configuration and the "n-802.1q" means it negotiated 802.1q encapsulation on the trunk link.

In addition to the **show interfaces trunk** command the **show interface switchport** command can confirm the same settings as shown below. The Administrative Mode on SW3 is set to "trunk" while on SW5 it is set to "dynamic desirable." This relates to the explicit configuration of the switchport. The Operational Mode on each switch is set to "trunk" indicating that both switch ports are operating in the trunk mode:

On SW3:

```
SW3#show interfaces e5/0 switchport

Name: Et5/0
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
```
(The rest of the output is omitted for brevity)

On SW5:

```
SW5#show interfaces e5/0 switchport
Name: Et5/0
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: On
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
```
(The rest of the output is omitted for brevity)

Finally, the Administrative and Operational Trunking Encapsulation is set to "**dot1q**" on SW3 because it has been statically configured as an 802.1q trunk. The Administrative Trunking Encapsulation on SW5 is "negotiate" because it is configured to dynamically negotiate the trunk encapsulation based on the **switchport mode dynamic desirable** command entered earlier. The Operational Trunking Encapsulation for SW5 is set to "802.1q" because it has negotiated an 802.1q encapsulation with SW3.

The Administrative Mode in this context is an expression of the administrator's intended configuration for the trunk port. On SW3, the administrator intends the port to operate as a trunk using 802.1q

encapsulation. On SW5, the administrator intends on the port dynamically determining whether or not to become an 802.1q trunk, ISL trunk, or a simple access port. The inclusion of this dynamic negotiation creates the need for the Operational Mode. The Operational Mode confirms the results of the dynamic negotiation. In case of SW3, because dynamic negotiation is not enabled, it will always operate as an 802.1q Trunk regardless. SW5, on the other hand, can operate in any of the three modes. The Operational status here indicates that it has negotiated an 802.1q trunk.

Task 4

Configure a trunk between SW3 and SW5, using their E5/1 interfaces. You should use an industry-standard protocol for the trunk encapsulation, based on the following policy:

SW3, E5/1

This port should be configured into permanent trunk mode, and it should negotiate to convert the neighboring interface into a trunk.

SW5, E5/1

This port should be configured to negotiate a trunk only if it receives negotiation packets from a neighboring port; this port should never start the negotiation process. You should not configure **switchport trunk encapsulation dot1q** on this port.

Currently there are two main methods of trunking encapsulation: The Cisco Proprietary Inter-Switch Link (ISL) and IEEE Standard 802.1Q encapsulation. Since the task explicitly states to use an industry standard protocol, the 802.1q encapsulation method will be configured on theE5/1 link between SW3 and SW5.

First, the task requires the following:

- **Setting the E5/1 interface on SW3 in a permanent trunking mode.**
- **The restriction in sub task 2 prevents the use of switchport trunk encapsulation dot1q command on SW5's E5/1 interface.**

The above requirements and restrictions are similar to SW3's configuration for its E5/0 interface in Task 3 resulting in same configuration commands on its E5/1 interface:

On SW3:

```
SW3(config)#interface e5/1
SW3(config-if)#switchport trunk encapsulation dot1q
SW3(config-if)#switchport mode trunk
SW3(config-if)#no shut
```

The configuration for SW5 for its E5/1 interface in this task is however slightly different from task 3:

On SW5:

```
SW5(config)#interface e5/1
SW5(config-if)#switchport mode dynamic auto
SW5(config-if)#no shut
```

Unlike task 3, task 4 requires SW5 to convert its E5/1 interface into a trunk port provided it receives DTP negotiation packets from SW3 over that link. As already mentioned, SW3 will send DTP packets even if the trunk mode has been statically set for its E5/1 interface. Therefore, all SW5 has to do is react to these DTP frames from SW3 and convert its own E5/1 interface to trunk. This feature can be activated with **dynamic auto** setting on SW5's E5/1 interface.

In the **Auto mode** (configured with the **switchport mode dynamic auto** command in interface configuration mode), DTP on SW5's e5/1 interface passively waits to receive a DTP frame from the neighboring switch, SW3. Once a DTP frame is received, SW5 will respond with its own DTP frame. Depending on the settings on the far end of the trunk link, the interface will negotiate to become a trunk or an access port. If the far end indicates trunking mode then the port will negotiate to trunk mode. If the far end indicates access mode, then the port will negotiate to access mode. In the case where the interface negotiates to become a trunk port, DTP will also negotiate the trunk encapsulation, just as seen in Task 3.
This operation is opposite of the dynamic desirable mode where the local switch port will actively send DTP frames in an attempt to negotiate a trunk link with a potential far-end switch interface. In Auto mode, no such frames are sent until a DTP frame has been received from a far-end switch.

Because SW3's e5/1 interface is configured as a static 802.1q trunk, SW3 will send DTP frames to negotiate the other end to be an 802.1q trunk as well. SW5's E5/1 interface, upon receiving these

frames, will accept the configuration because it has been configured to operate in the **dynamic auto** mode.
The following details the relevant configuration commands required on SW3 and SW5 to complete the task:

On SW3:

```
SW3(config)#interface e5/1
SW3(config-if)#switchport trunk encapsulation dot1q
SW3(config-if)#switchport mode trunk
SW3(config-if)#no shut
```

On SW5:

```
SW5(config)#interface e5/1
SW5(config-if)#switchport mode dynamic auto
SW5(config-if)#no shut
```

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et5/0       on               802.1q         trunking      1
Et5/1       on               802.1q         trunking      1
```

(The rest of the output is omitted for brevity)

On SW5:

In the output of the following show command you should note the difference between "dynamic desirable" and "Dynamic auto".

```
SW5#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et5/0       desirable        n-802.1q       trunking      1
Et5/1       auto             n-802.1q       trunking      1
```

(The rest of the output is omitted for brevity)

The **show interfaces E5/1 switchport** command output can also be used to verify the above settings. SW3's administrative and operation mode setting reflect the static configurations made by the administrator. SW5's administrative mode shows dynamic auto (as result of the **switchport dynamic auto** command) and the operational mode is trunk (as a result of DTP negotiation between itself and SW3). The administrative trunking encapsulation is negotiated with SW3 as well, resulting in dot1q encapsulation being used on the trunk link. The 802.1q trunk link is now operational between SW3 and SW5's E5/1 interfaces.

On SW3:

```
SW3#show interfaces e5/1 switchport
Name: Et5/1
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```
(The rest of the output is omitted for brevity)

On SW5:

```
SW5#show interfaces e5/1 switchport
Name: Et5/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: trunk
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: dot1q
```
(The rest of the output is omitted for brevity)

Task 5

Configure a trunk link between SW4 and SW5, using their E4/0 interfaces. These ports should be configured to negotiate the neighboring interface into a dot1q trunk, but they should not be in permanent trunk mode.

This task forbids a permanent trunk mode on SW4 and SW5's E4/0 link. This hints towards not configuring the switchport mode trunk command on SW4 and SW5's E4/0 interface and instead have the both the switches dynamically negotiate the trunk mode. In addition, the task also specifies the 802.1.q trunk encapsulation method.

When configuring automatic trunk negotiation, there are two options desirable and auto. In desirable mode, the switch will actively send DTP frames out of the configured port in an attempt to negotiate a trunk port. In auto mode, the switch will wait passively to receive a DTP frame from the far end before negotiating a trunk link. The task requires the switches to be configured to negotiate the trunk but not be in a permanent trunking mode. This hints to the desirable configuration on the interfaces.

In addition to sending DTP frames, the contents of the DTP frame is important. The task states that the trunk link should be an 802.1q trunk but that this should be negotiated. In this case, it is not sufficient to only input the dynamic desirable command on the interfaces. This is because both switches support ISL trunk encapsulation as well as shown in the below:

On SW4:

```
SW4(config)#interface e4/0
SW4(config-if)#switchport trunk encapsulation ?
  dot1q      Interface uses only 802.1q trunking encapsulation when trunking
  isl        Interface uses only ISL trunking encapsulation when trunking
  negotiate  Device will negotiate trunking encapsulation with peer on
             interface
```

On SW5:

```
SW5(config)#interface e4/0
SW5(config-if)#switchport trunk encapsulation ?
  dot1q      Interface uses only 802.1q trunking encapsulation when trunking
  isl        Interface uses only ISL trunking encapsulation when trunking
  negotiate  Device will negotiate trunking encapsulation with peer on
             interface
```

By default, the trunk port operates in the negotiate mode. In this mode, if the trunk encapsulation isn't explicitly set to 802.1q, DTP will negotiate an ISL trunk. For this reason, to complete the task, the switchport trunk encapsulation dot1q command needs to be entered on both sides of the trunk link. This command constrains DTP to only negotiate an 802.1q trunk link during negotiation.

Keeping the above in mind, the task is completed by issuing the switchport trunk encapsulation dot1q and the switchport mode dynamic desirable command on SW4 and SW5's E4/0 interface:

On Both Switches:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk encapsulation dot1q
SWx(config-if)#switchport mode dynamic desirable
SWx(config-if)#no shut
```

To verify the configuration:

On SW4:

```
SW4#show interfaces trunk | include 802

Et4/0       desirable    802.1q         trunking      1
```

On SW5:

```
SW5#show interfaces trunk | include 802

Et4/0       desirable       802.1q         trunking      1
Et5/0       desirable       n-802.1q       trunking      1
Et5/1       auto            n-802.1q       trunking      1
```

The administrative mode on both switches shows dynamic desirable. This output confirms that both switches are configured to actively send DTP frames to negotiate the trunking mode. The "802.1q" indicates the encapsulation method has been statically set to 802.1q The static configuration of 802.1q, once again, forces DTP to use 802.1q encapsulation as its negotiated encapsulation method. It also forces the "802.1q" to be displayed in the output of the show interfaces trunk command instead of "n-802.1q" as shown in the output above for the Et5/0 and Et5/1 interfaces. The key here is that the mode is set to "desirable" AND the encapsulation method is statically set to 802.1q. This is an indication that DTP will attempt to negotiate an 802.1q trunk. The "n" only appears if the local switch does not have a static setting for trunk encapsulation as an indication that the method was negotiated and not statically set by the administrator.

Once both switches receive each other's DTP frames, negotiation occurs. The result of this negotiation is seen as trunk as the operational mode. The administrative and operational trunk encapsulation on both ends show the manually configured encapsulation method of dot1q.

On SW4:

```
SW4#show interfaces e4/0 switchport
Name: Et4/0
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

On SW5:

```
SW5#show interfaces e4/0 switchport
Name: Et4/0
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

Task 6

Configure a dot1q trunk between SW4 and SW5, using their E4/1 interfaces, based on the following policy:

SW4, E4/1

This port should be configured to actively attempt to convert the link to a trunk. This port should not be in permanent trunk mode.

SW5, E4/1

This port should be configured to negotiate a trunk only if it receives negotiation packets from a neighboring port; this port should never start the negotiation process or be configured with the **switchport trunk encapsulation dot1q** command.

This task demonstrates the trunking capabilities when both sides of the link are in dynamic trunk mode. In this case, the SW4 side of the trunk negotiation should be configured in the desirable trunk mode as follows:

On SW4:

```
SW4(config)#interface e4/1
SW4(config-if)#switchport trunk encapsulation dot1q
SW4(config-if)#switchport mode dynamic desirable
SW4(config-if)#no shut
```

With the above configuration, SW4 actively sends DTP frames out its E4/1 interface connected to SW5 with the intention of forming an 802.1q trunk link. The switchport trunk encapsulation dot1q command ensures that 802.1q is theencapsulation method SW4 attempts to negotiate with SW5 on the other side of the link.

SW5, on the other hand, should not begin negotiations for forming a trunk link nor should it have a preference for trunk encapsulation mode. This is a clear hint towards using the dynamic auto mode of trunk negotiation. The configuration on SW5 as seen below configures it to passively wait to receive DTP frames from the far end switch port by configuring the dynamic auto mode.

On SW5:

```
SW5(config)#interface e4/1
SW5(config-if)#switchport mode dynamic auto
SW5(config-if)#no shut
```

Notice how this time, the switchport trunk encapsulation dot1q command is omitted in the configuration. This omission was a direct result of the task requirements, stating that it should not be configured with the “switchport trunk encapsulation dot1q” command.

After configuring both ends of the trunk, the **show interface trunk** output on SW4 and SW5 is as follows

To verify the configuration:

On SW4:

```
SW4#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et4/0       desirable        802.1q         trunking      1
```

```
Et4/1         desirable         802.1q         trunking       1
```

On SW5:

```
SW5#show interfaces trunk

Port          Mode              Encapsulation  Status         Native vlan
Et4/0         desirable         802.1q         trunking       1
Et4/1         auto              n-802.1q       trunking       1
Et5/0         desirable         n-802.1q       trunking       1
Et5/1         auto              n-802.1q       trunking       1
```

The output above confirms that SW4's E4/1 interface is configured in the dynamic desirable mode using 802.1q trunk encapsulation. SW5's E4/1 interface is configured in the dynamic auto mode. Its trunk encapsulation shows that it has negotiated 802.1q trunking with the "n-" in front of the 802.1q designation.

The show interfaces E4/1 switchport output below also confirms the above configuration. Since SW4 is actively sending out DTP frames to negotiate a trunk, the administrative mode shows dynamic desirable. SW5 is configured to passively listen to DTP frames as indicated with the administrative mode dynamic auto below. SW4 will send DTP frames to SW5 to negotiate an 802.1q trunk link. SW5 will respond and agree to the configuration. As a result, the operational mode for the E4/1 interface on both SW4 and SW5 is shown as trunk.

On SW4:

```
SW4#show interfaces e4/1 switchport
Name: Et4/1
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

On SW5:

```
SW5#show interfaces e4/1 switchport
Name: Et4/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: trunk
Administrative Trunking Encapsulation: negotiate
```

```
Operational Trunking Encapsulation: dot1q
```

Since the encapsulation is manually set on SW4, the administrative trunking encapsulation is seen as dot1q. SW5 sets the encapsulation method to 802.1q based on the DTP frames received from SW4. As such, the administrative trunking encapsulation is seen as negotiate, while the operational trunk encapsulation has been set to dot1q as a result of the DTP negotiation process.

Task 7

Configure a dot1q trunk between SW3 and SW4, using their E6/0 interfaces; these switches should be configured into permanent trunk mode and negotiate the neighboring interface into a trunk.

The configuration for this task is very simple. The trunk mode should be manually set on both switches with the switchport mode trunk command. Prior to this, however, IOS needs to be instructed on the encapsulation method to be used for this link or else the static trunk command will be rejected by the software. Thus, the switchport trunk encapsulation dot1q command should be entered first, to designate the trunk encapsulation mode followed by the switchport mode trunk command to set the permanent trunking mode as seen below:

On Both Switches:

```
SWx(config)#interface e6/0
SWx(config-if)#switchport trunk encapsulation dot1q
SWx(config-if)#switchport mode trunk
SWx(config-if)#no shut
```

The **show interface trunk** command output from SW3 and SW4 below verify the result of the above configuration settings:

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk | include 802


Port        Mode            Encapsulation  Status        Native vlan
Et5/0       on              802.1q         trunking      1
Et5/1       on              802.1q         trunking      1
Et6/0       on              802.1q         trunking      1
```

On SW4:

```
SW4#show interfaces trunk | include 802

Port        Mode            Encapsulation  Status        Native vlan
Et4/0       desirable       802.1q         trunking      1
Et4/1       desirable       802.1q         trunking      1
Et6/0       on              802.1q         trunking      1
```

As seen above, the trunk mode and encapsulation on both switches is set to **on, indicating trunk mode is statically configured as "on"**, and **802.1q**. This is also reflected in the below output where both the administrative modes and operational modes are set to trunk and 802.1q:

On SW3:

```
SW3#show interfaces e6/0 switchport
Name: Et6/0
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

On SW4:

```
SW4#show interfaces e6/0 switchport
Name: Et6/0
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
```

The ports have been configured as permanent trunk interfaces. As a result, they will still send DTP messages out to "negotiate" a trunk link. The difference is the port will only operate in trunk mode and will advertise this in the DTP messages it sends out. In this way, the port is saying that its own operating mode is non-negotiable because of its static configuration and will force other ports operating in dynamic modes to be in trunking mode.

Task 8

Configure a dot1q trunk between SW3 and SW4, using their E6/1 interfaces. These ports should not use DTP to negotiate a trunk.

This task requires configuring a dot1q trunk on the E6/1 interface between SW3 and SW4 without the use of DTP. On certain switching platforms, DTP is turned on by default even if the port is statically configured to function as an access or trunk port

DTP can be turned off using the switchport nonegotiate command in interface configuration mode. In this mode, the switch will disable DTP on the port, preventing the processing and sending of DTP frames. However, issuing this command by itself results in IOS rejecting the command as seen below:

```
SW3(config-if)#switchport nonegotiate

Command rejected: Conflict between 'nonegotiate' and 'dynamic' status on
this interface: Et6/1
```

This is because, prior to turning off DTP, the switch needs to be told manually about the encapsulation method and the mode in which the port should operate. On most Cisco switching platforms, the default mode of operation is to use dynamic negotiation for trunk status as shown below:

```
SW3:

SW3#show interfaces e6/1 switchport
Name: Et6/1
Switchport: Enabled
Administrative Mode: dynamic auto
Operational Mode: down
Administrative Trunking Encapsulation: negotiate
Negotiation of Trunking: On
```

In order for the port to function properly without DTP, Cisco IOS needs to have the static configuration settings. To comply, first, set the encapsulation method to 802.1.q with the switchport trunk encapsulation dot1q command. Then, set the trunk mode with the switchport mode trunk command. Finally, DTP is turned off with the switchport nonegotiate command. The following configurations are made on SW3 and SW4:

On Both Switches:

```
SWx(config)#interface e6/1
SWx(config-if)#switchport trunk encapsulation dot1q
SWx(config-if)#switchport mode trunk
SWx(config-if)#switchport nonegotiate
SWx(config-if)#no shut
```

Remember, the above commands are order specific. This means switchport mode trunk cannot be entered before the switchport trunk encapsulation dot1q command. Likewise, the switchport nonegotiate command cannot be entered before the switchport mode trunk command.

The show interfaces trunk output below confirms the manual setting of an 802.1q trunk link with the on and 802.1q keywords. This output however by itself does not reveal that DTP is disabled on the E6/1 link. To confirm DTP is not enabled the show interfaces E6/1 switchport command output can be used:

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk

Port          Mode             Encapsulation  Status        Native vlan
Et5/0         on               802.1q         trunking      1
Et5/1         on               802.1q         trunking      1
Et6/0         on               802.1q         trunking      1
Et6/1         on               802.1q         trunking      1

SW3#show interfaces e6/1 switchport
Name: Et6/1
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: Off
```

On SW4:

```
SW4#show interfaces trunk

Port          Mode             Encapsulation  Status        Native vlan
Et4/0         desirable        802.1q         trunking      1
Et4/1         desirable        802.1q         trunking      1
Et6/0         on               802.1q         trunking      1
Et6/1         on               802.1q         trunking      1

SW4#show interfaces e6/1 switchport
Name: Et6/1
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: Off
```

Task 9

Configure trunking on the E4/0-1 interfaces of SW2 and SW3, the E5/0-1 interfaces on SW2 and SW4, and the E6/0-1 interfaces of SW2 and SW5. These ports should be in permanent trunk mode.

This task requires statically configuring the mentioned ports to trunk. For this, following the order as described in tasks 7 and 8, first the encapsulation method is manually set to 802.1q with the switchport trunk encapsulation dot1q command. Then the command switchport mode trunk is issued to set the ports to permanent trunking mode. To ease with configuration, the interface range command is used to configure multiple ports at the same time on all of the switches:

On SW2:

```
SW2(config)#interface range e4/0-1,e5/0-1,e6/0-1
SW2(config-if-range)#switchport trunk encapsulation dot1q
SW2(config-if-range)#switchport mode trunk
SW2(config-if-range)#no shut
```

On SW3:

```
SW3(config)#interface range e4/0-1
SW3(config-if-range)#switchport trunk encapsulation dot1q
SW3(config-if-range)#switchport mode trunk
SW3(config-if-range)#no shut
```

On SW4:

```
SW4(config)#interface range e5/0-1
SW4(config-if-range)#switchport trunk encapsulation dot1q
SW4(config-if-range)#switchport mode trunk
SW4(config-if-range)#no shut
```

On SW5:

```
SW5(config)#interface range e6/0-1

SW5(config-if-range)#switchport trunk encapsulation dot1q
SW5(config-if-range)#switchport mode trunk
SW5(config-if-range)#no shut
```

<u>To verify the configuration:</u>

<u>On SW2:</u>

```
SW2#show interfaces trunk | include 802

Et4/0       on              802.1q        trunking      1
Et4/1       on              802.1q        trunking      1
Et5/0       on              802.1q        trunking      1
Et5/1       on              802.1q        trunking      1
Et6/0       on              802.1q        trunking      1
Et6/1       on              802.1q        trunking      1
```

<u>On SW3:</u>

```
SW3#show interfaces trunk

Et4/0       on              802.1q        trunking      1
Et4/1       on              802.1q        trunking      1
Et5/0       on              802.1q        trunking      1
Et5/1       on              802.1q        trunking      1
Et6/0       on              802.1q        trunking      1
Et6/1       on              802.1q        trunking      1
```

<u>On SW4:</u>

```
SW4#show interfaces trunk

Et4/0       desirable       802.1q        trunking      1
Et4/1       desirable       802.1q        trunking      1
Et5/0       on              802.1q        trunking      1
Et5/1       on              802.1q        trunking      1
Et6/0       on              802.1q        trunking      1
Et6/1       on              802.1q        trunking      1
```

<u>On SW5:</u>

```
SW5#show interfaces trunk | include 802
Et4/0       desirable       802.1q        trunking      1
Et4/1       auto            n-802.1q      trunking      1
Et5/0       desirable       n-802.1q      trunking      1
Et5/1       auto            n-802.1q      trunking      1
Et6/0       on              802.1q        trunking      1
Et6/1       on              802.1q        trunking      1
```

Task 10

Configure the following VLANs on SW2 and ensure that they are propagated to the other switches:

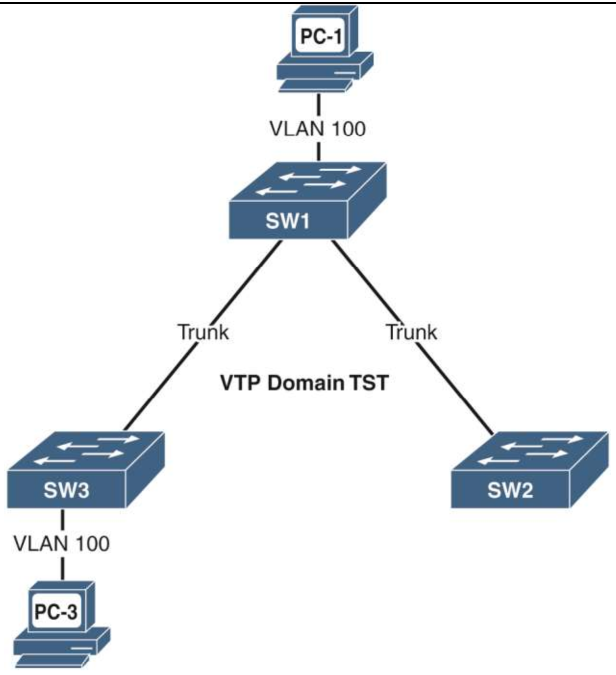
VLANs 2–10, 100, 200, 300, 400, 230, 350, 450, 240, 250, and 340

This task requires configuring the VLANs mentioned only on SW2. Configuring VLANs can be a manual process. However, recall in Task 1, SW2 through SW5 were all configured to belong to the VTP domain TST. All tasks up to this point configured trunk links between switches. Now that all of the switch-to-switch links are properly configured as trunk ports, they can relay and receive VTP messages.

In other words, through VTP, the VLANs configured on SW2 will be advertised and synchronized to all the other switches in this topology. VTP performs this synchronization through the use of three message types:

1. **Summary Advertisements**
2. **Subset Advertisements**
3. **Advertisement Requests**

To help demonstrate what each advertisement type does, picture the following small, switched network below:



All switches above have been configured to participate in VTP domain TST.

Without any changes happening in the network, VTP servers will periodically send out VTP summary advertisements every 300 seconds or whenever a change is detected in the VLAN database. These advertisements contain information about the sending switch's VTP version, its configured domain name, and the configuration revision number.

The VTP version can be either 1, 2, or 3. The domain name indicates to which VTP domain the advertising switch belongs. If the domain name in the summary advertisement doesn't match the locally configured domain name, the switch will ignore and drop the summary advertisement message. The revision number is an indication of how many times the VLAN database has been updated by the switch sending the summary advertisement. This number is increased every time a change is made to the VLAN configuration. It is also used to help neighboring switches determine if their version of the VLAN database has the latest configuration revision.

The VTP summary advertisement also includes a field called the followers field. This field indicates how many additional subset advertisements the switch is sending out after the summary advertisement. For example, VLAN 100, pictured above, is added to SW1. As a reaction, Switch-1 sends the following VTP summary advertisement message to Switch-2 and Switch-3:

```
VLAN Trunking Protocol
    Version: 0x01
    Code: Summary Advertisement (0x01)
    Followers: 1
    Management Domain Length: 3
    Management Domain: TST
    Configuration Revision Number: 1
    Updater Identity: 0.0.0.0
    Update Timestamp: 20-07-05 16:09:57
    MD5 Digest: 7d5a50584977788466c5ebd8eae182b5
```

From the summary advertisement above, Switch-2 and Switch-3 know the following:

1. **Switch-1 is using VTPv1 messaging**
2. **Switch-1 is in the VTP domain TST**
3. **Switch-1's VTP database is at configuration revision 1**
4. **There is one subset advertisement following this message**

After sending the summary advertisement, Switch-1 follows up with a VTP subset advertisement. Subset advertisements are VTP messages that contain actual information about the VLANs configured on the switch including but not limited to VLAN number, name, and type information. They are usually sent after summary advertisements. In Switch-1's case, its subset advertisement contains information about the newly created VLAN 100 shown below:

```
VLAN Trunking Protocol
    Version: 0x01
    Code: Subset Advertisement (0x02)
    Sequence Number: 1
    Management Domain Length: 3
    Management Domain: TST
    Configuration Revision Number: 1
    VLAN Information
    VLAN Information
        VLAN Information Length: 20
        Status: 0x00
        VLAN Type: Ethernet (0x01)
        VLAN Name Length: 8
        ISL VLAN ID: 0x0064
        MTU Size: 1500
        802.10 Index: 0x00018704
        VLAN Name: VLAN0100
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
```

When the Switch-2 and Switch-3 receive the subset advertisement above, they will update their own VLAN databases with the information contained. In this case, the switches will create VLAN 100 if they do not already have it created in their local VLAN database. If the VLAN exists, it will have the information, such as type, name, and MTU, updated to match what was advertised in the subset message.

The above all leads to the following sequence of events for VTP to synchronize VLAN configuration among a group of switches:

1. VTP server sends out VTP summary advertisement with followers field set to 1 or greater
2. VTP clients receive the summary advertisement and expect subset advertisements to follow
3. VTP server sends subset advertisements accordingly.

This process works well if there is a change made on the server that requires subset advertisements to follow the summary advertisement. There are situations where a new switch comes online with a revision number of 0 and receives a periodic summary message that has the followers field set to 0 and a higher revision number. The followers field set to 0 means there will not be any subset advertisements following the summary message.

In the above case, the new switch will make use of the third VTP message type, the advertisement request message. The advertisement request message is sent by a switch to indicate that it requires VLAN configuration information to update its local VLAN database. The following is an example of an advertisement request message.

```
VLAN Trunking Protocol
    Version: 0x01
    Code: Advertisement Request (0x03)
    Reserved: 00
    Management Domain Length: 4
    Management Domain: TST
    Start Value: 0x0000
```

The advertisement request message has the simplest form with the only notable field being the VTP domain name which is used to ensure only VTP servers participating in the same domain respond to the advertisement request. Switches send this message out whenever they receive a summary advertisement with a higher revision number than its own and followers field set to 0. When a VTP server receives the advertisement request message, it will respond with a subset advertisement containing the contents of the current version of the VLAN database. From this subset advertisement, the receiving switch can update its own local VLAN database to be synchronized with the rest of the VTP domain.

As an example, the E0/1 interface that connects to Switch-1 on Switch-2 is shutdown. VLAN 200 is configured on Switch-1.

On Switch-2:

```
Switch-2(config)#interface e0/1
Switch-2(config-if)#shut ! Interface shutdown on Switch-2
```

On Switch-1:

```
Switch-1(config)#vlan 200 ! VLAN 200 added on Switch-1
Switch-1(config-vlan)#exit
```

When the interface is brought back up on Switch-2, it receives a summary advertisement from Switch-1 with a higher configuration revision number than its own and the followers field set to 0. This value of 0 indicates that Switch-1 is not planning on sending a subset advertisement following its summary advertisement.

Switch-2:

```
Switch-2#show vtp status
VTP Version capable               : 1 to 3
VTP version running               : 1
VTP Domain Name                   : TST
VTP Pruning Mode                  : Disabled
VTP Traps Generation              : Disabled
Device ID                         : aabb.cc80.0200
Configuration last modified by 0.0.0.0 at 7-5-20 16:09:57
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode                  : Server
Maximum VLANs supported locally     : 1005
Number of existing VLANs            : 6
Configuration Revision              : 1 ! Revision number 1 on Switch-2
MD5 digest                          : 0x7D 0x5A 0x50 0x58 0x49 0x77 0x78 0x84
                                      0x66 0xC5 0xEB 0xD8 0xEA 0xE1 0x82 0xB5
Switch-2(config)#interface e0/1
```

```
Switch-2(config-if)#no shut ! Interface unshut on Switch-2

! Summary Advertisement with revision number 2 from Switch-1

VLAN Trunking Protocol
    Version: 0x01
    Code: Summary Advertisement (0x01)
    Followers: 0 ! Followers field is set to 0
    Management Domain Length: 3
    Management Domain: TST
    Configuration Revision Number: 2
    Updater Identity: 0.0.0.0
    Update Timestamp: 20-07-05 16:22:47
    MD5 Digest: bfe9a9d0ddf6bef5d9d1633a811ebd35
```

This discrepancy obviously means that the VLAN database on Switch-2 isn't updated. Since Switch-2 needs to synchronize its own VLAN database to match the rest of the VTP domain, it sends an **advertisement request** to Switch-1 to get the subset advertisements it needs.

In response, Switch-1 sends a subset advertisement that carries information about its updated VLAN database. The capture below shows the adverisment request from Switch-2 and the subset advertisement response by Switch-1

```
! Advertisement request sent to Switch-1

VLAN Trunking Protocol
    Version: 0x01
    Code: Advertisement Request (0x03)
    Reserved: 00
    Management Domain Length: 3
    Management Domain: TST
    Start Value: 0x0000

! Subset Advertisement sent by Switch-1

VLAN Trunking Protocol
    Version: 0x01
    Code: Subset Advertisement (0x02)
    Sequence Number: 1
    Management Domain Length: 3
    Management Domain: TST
    Configuration Revision Number: 2
    VLAN Information
    VLAN Information
    VLAN Information
```

```
        VLAN Information Length: 20
        Status: 0x00
        VLAN Type: Ethernet (0x01)
        VLAN Name Length: 8
        ISL VLAN ID: 0x00c8
        MTU Size: 1500
        802.10 Index: 0x00018768
        VLAN Name: VLAN0200
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
```

It is this exchange of summary advertisements, subset advertisements, and advertisement request that allows VTP to synchronize VLAN information between switches. Summary advertisements and advertisement requests organize the exchange of subset advertisements. Subset advertisements contain the actual data that should be synchronized. The above example was to demonstrate how the VLAN information entered on SW2 in the lab topology will be distributed to all of the other switches in the domain.

VLANs are configured with the VLAN keyword. Multiple VLANs can be configured with a single command with each VLAN or VLAN range separated by a comma. As the task requires, the following shows the VLANs configuration on SW2.

Note : It is important to issue the exit command after configuring VLANs for the VLANs to be added to the VLAN database*:*

On SW2

```
SW2(config)#vlan 2-10,100,200,300,400,230,350,450,240,250,340
```

You must "exit" for the VLANs to be propagated:

```
SW2(config-vlan)#exit
```

After configuring the above VLANs, SW2 first sends out a VTP summary advertisement frame out of all of its trunk links. This summary advertisement frame contains all of the information in SW2's VLAN database that has changed. An example packet capture of the summary advertisement sent to SW5 over the E6/1 trunk link is shown below:

```
Frame 56: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface 0
Ethernet II, Src: aa:bb:cc:00:0a:06 (aa:bb:cc:00:0a:06), Dst: CDP/VTP/DTP/PAgP/UDLD
```

```
(01:00:0c:cc:cc:cc)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 1
Logical-Link Control
VLAN Trunking Protocol
    Version: 0x01
    Code: Summary Advertisement (0x01)
    Followers: 1
    Management Domain Length: 3
    Management Domain: TST
    Configuration Revision Number: 1
    Updater Identity: 0.0.0.0
    Update Timestamp: 20-07-01 18:30:03
    MD5 Digest: 0f71b236b776e0a7d9f6ee6a3ee312dc
```

As seen above, the summary advertisement carries within itself, the VTP domain name TST. When SW5 receives this summary advertisement, it compares the VTP domain name to its own. Since the advertised domain name matches its own, SW5 then computes the MD5 digest of the message. If it results in the same value then SW5 knows definitively that it should accept the VTP information in the VTP frame. Below is SW5's VTP information:

```
SW5#show vtp status
VTP Version capable                : 1 to 3
VTP version running                : 1
VTP Domain Name                    : TST
VTP Pruning Mode                   : Disabled
VTP Traps Generation               : Disabled
Device ID                          : aabb.cc80.0d00
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode                   : Client
Maximum VLANs supported locally      : 1005
Number of existing VLANs             : 5
Configuration Revision               : 0
MD5 digest                           : 0x57 0xCD 0x40 0x65 0x63 0x59 0x47 0xBD
                                       0x56 0x9D 0x4A 0x3E 0xA5 0x69 0x35 0xBC
```

NOTE: the MD5 digest is recalculated based on the contents of the VLAN database whenever a VTP message is received. See the show vtp status output for confirmation of SW5's MD5 calculation

The followers field in the summary advertisement lets SW5 know that a subset advertisement is following SW2's summary advertisement. The subset advertisement is what actually carries the VLAN information.

The packet capture below shows the subset advertisement sent to SW5. Each of the VLAN information entries below corresponds to the VLANs configured on SW2.

```
Frame 57: 610 bytes on wire (4880 bits), 610 bytes captured
(4880 bits) on interface 0
Ethernet II, Src: aa:bb:cc:00:0a:06 (aa:bb:cc:00:0a:06), Dst:
CDP/VTP/DTP/PAgP/UDLD (01:00:0c:cc:cc:cc)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 1
Logical-Link Control
VLAN Trunking Protocol
    Version: 0x01
    Code: Subset Advertisement (0x02)
    Sequence Number: 1
    Management Domain Length: 3
    Management Domain: TST
    Configuration Revision Number: 1
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
    VLAN Information
```

For a detailed look into the VLAN Information entries, the following expands one entry that carries information for the VLAN 230:

```
VLAN Information
    VLAN Information Length: 20
    Status: 0x00
    VLAN Type: Ethernet (0x01)
    VLAN Name Length: 8
    ISL VLAN ID: 0x00e6
    MTU Size: 1500
    802.10 Index: 0x00018786
    VLAN Name: VLAN0230
```

The "ISL VLAN ID" field in the above carries the VLAN ID in hexadecimal. In the above the hex code 0x00e6 translates to decimal 230 which is one of the VLANs configured on SW2.

All of the above VLAN information in subset advertisements is advertised by SW2 out of all its trunk interfaces. All switches in the topology receive this information and populate their own VLAN databases with the VLANs configured on SW2. This can be verified on all the switches with the show vlan brief | include VLAN command. As an example, the following output is displayed from SW5:

To verify the configuration:

On All Switches:

```
SWx#show vlan brief | include VLAN

VLAN Name                             Status    Ports
2    VLAN0002                         active
3    VLAN0003                         active
4    VLAN0004                         active
5    VLAN0005                         active
6    VLAN0006                         active
7    VLAN0007                         active
8    VLAN0008                         active
9    VLAN0009                         active
10   VLAN0010                         active
100  VLAN0100                         active
200  VLAN0200                         active
230  VLAN0230                         active
240  VLAN0240                         active
250  VLAN0250                         active
300  VLAN0300                         active
340  VLAN0340                         active
```

```
350  VLAN0350                         active
400  VLAN0400                         active
450  VLAN0450                         active
```

After adding the VLANs, SW5's show vtp status output reveals when the configuration was last updated, its revision number has incremented, and the MD5 digest now matches that which was advertised by SW2:

```
SW5#show vtp status
VTP Version capable               : 1 to 3
VTP version running               : 1
VTP Domain Name                   : TST
VTP Pruning Mode               : Disabled
VTP Traps Generation           : Disabled
Device ID                      : aabb.cc80.0d00
Configuration last modified by 0.0.0.0 at 20-07-01 18:30:03

Feature VLAN:
--------------
VTP Operating Mode                : Client
Maximum VLANs supported locally   : 1005
Number of existing VLANs          : 24
Configuration Revision            : 1
MD5 digest                        : 0x0F 0x71 0xB2 0x36 0xB7 0x76 0xE0 0xA7
                                    0xD9 0xF6 0xEE 0x6A 0x3E 0xE3 0x12 0xDC
```

Task 11

Configure the trunks based on the following policy:

Policy Item	Trunk Interface	Between Switches	Allowed VLAN/s
1	E4/1	SW2 ←→ SW3	ONLY 230
2	E5/0	SW3 ←→ SW5	ONLY 350
3	E4/0	SW4 ←→ SW5	ONLY 450
4	E5/0	SW2 ←→ SW4	ONLY 240
5	E6/0	SW2 ←→ SW5	ONLY 250
6	E6/0	SW3 ←→ SW4	ONLY 340

Trunk links have a list of VLANs that are allowed to flow over them. By default, all VLANs are included in this list. This task specifies several VLAN policies that make changes to the list of VLANs that should be allowed over a particular trunk link. Each policy lists the trunk interface and switch pair to which the policy applies and which VLANs are allowed or disallowed to be carried across the trunk.

The allowed VLAN list of a trunk link is configured using the switchport trunk allowed vlan {{ add | except | remove } *vlan_list* | all | none } command.

Without additional parameters, the command configures the allowed VLAN list to match the specified VLAN list exactly. This task examines this use-case for the command.

Policy item 1:

As seen below, all VLANs are allowed on the E4/1 trunk interface as expected by the default settings of trunk links:

On SW2:

```
SW2#show interfaces trunk | include Vlans|Et4/1

Et4/1       on              802.1q        trunking      1
Port        Vlans allowed on trunk
Et4/1       1-4094
Port        Vlans allowed and active in management domain
Et4/1       1-10,100,200,230,240,250,300,340,350,400,450
Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       1-10,100,200,230,240,250,300,340,350,400,450
```

On SW3:

```
SW3#show interfaces trunk | include Vlans|Et4/1

Et4/1       on              802.1q        trunking      1
Port        Vlans allowed on trunk
Et4/1       1-4094
Port        Vlans allowed and active in management domain
Et4/1       1-10,100,200,230,240,250,300,340,350,400,450
Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       1-10,100,200,230,240,250,300,340,350,400,450
```

To configure the first policy in the task:

On SW2 and SW3:

The policy for this trunk interface states that only VLAN 230 should be allowed on this trunk link. This is achieved with the switchport trunk allowed vlan 230 command on the E4/1 interfaces on SW2 and SW3:

```
SWx(config)#interface e4/1
SWx(config-if)#switchport trunk allowed vlan 230
```

To verify the configuration:

On SW2:

As a result of the above command, the show interfaces trunk command output on SW2 and SW3 verify 230 to be the only VLAN allowed on the E4/1 interface:

```
SW2#show interfaces trunk | include Vlans|Et4/1

Et4/1       on              802.1q        trunking      1
Port        Vlans allowed on trunk
Et4/1       230
Port        Vlans allowed and active in management domain
Et4/1       230
Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       230
```

Similar configurations as the above are made on the remaining trunk links stated in this task

Policy item 2:

To configure the second policy in the task:

On SW3 and SW5:

```
SWx(config)#interface e5/0
SWx(config-if)#switchport trunk allowed vlan 350
```

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk | include Vlan|Et5/0

Et5/0       on              802.1q        trunking      1
Port        Vlans allowed on trunk
Et5/0       350
Port        Vlans allowed and active in management domain
Et5/0       350
Port        Vlans in spanning tree forwarding state and not pruned
Et5/0       350
```

<u>On SW5:</u>

```
SW5#show interfaces trunk | include Vlans|Et5/0

Et5/0       desirable       n-802.1q      trunking      1
Port        Vlans allowed on trunk
Et5/0       350
Port        Vlans allowed and active in management domain
Et5/0       350
Port        Vlans in spanning tree forwarding state and not pruned
Et5/0       none
```

<u>Policy item 3:</u>

<u>To configure the third policy in the task:</u>

<u>On SW4 and SW5:</u>

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk allowed vlan 450
```

<u>To verify the configuration:</u>

<u>On SW4:</u>

```
SW4#show interfaces trunk | include Vlan|Et4/0

Et4/0       desirable       802.1q        trunking      1
Port        Vlans allowed on trunk
Et4/0       450
Port        Vlans allowed and active in management domain
Et4/0       450
Port        Vlans in spanning tree forwarding state and not pruned
```

```
Et4/0         450
```

On SW5:

```
SW5#show interfaces trunk | include Vlan|Et4/0

Et4/0         desirable        802.1q         trunking      1
Port          Vlans allowed on trunk
Et4/0         450
Port          Vlans allowed and active in management domain
Et4/0         450
Port          Vlans in spanning tree forwarding state and not pruned
Et4/0         none
```

Policy item 4:

To configure the fourth policy in the task:

On SW2 and SW4:

```
SWx(config)#interface e5/0
SWx(config-if)#switchport trunk allowed vlan 240
```

To verify the configuration:

On SW2:

```
SW2#show interfaces trunk | include Vlans|Et5/0

Et5/0         on               802.1q         trunking      1
Port          Vlans allowed on trunk
Et5/0         240
Port          Vlans allowed and active in management domain
Et5/0         240
Port          Vlans in spanning tree forwarding state and not pruned
Et5/0         240
```

On SW4:

```
SW4#show interfaces trunk | inc Vlans|Et5/0

Et5/0         on               802.1q         trunking      1
Port          Vlans allowed on trunk
Et5/0         240
```

```
Port          Vlans allowed and active in management domain
Et5/0         240
Port          Vlans in spanning tree forwarding state and not pruned
Et5/0         240
```

Policy item 5:

To configure the fifth policy in the task:

On SW2 and SW5:

```
SWx(config)#interface e6/0
SWx(config-if)#switchport trunk allowed vlan 250
```

To verify the configuration:

On SW2:

```
SW2#show interfaces trunk | include Vlan|Et6/0

Et6/0         on                802.1q         trunking      1
Port          Vlans allowed on trunk
Et6/0         250
Port          Vlans allowed and active in management domain
Et6/0         250
Port          Vlans in spanning tree forwarding state and not pruned
Et6/0         250
```

On SW5:

```
SW5#show interfaces trunk | include Vlan|Et6/0

Et6/0         on                802.1q         trunking      1
Port          Vlans allowed on trunk
Et6/0         250
Port          Vlans allowed and active in management domain

Et6/0         250
Port          Vlans in spanning tree forwarding state and not pruned
Et6/0         250
```

Policy item 6:

To configure the sixth policy in the task:

On SW3 and SW4:

```
Cat-x(config)#interface e6/0
Cat-x(config-if)#switchport trunk allowed vlan 340
```

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk | include Vlan|Et6/0

Et6/0       on               802.1q         trunking      1
Port        Vlans allowed on trunk
Et6/0       340
Port        Vlans allowed and active in management domain
Et6/0       340
Port        Vlans in spanning tree forwarding state and not pruned
Et6/0       340
```

On SW4:

```
SW4#show interfaces trunk | include Vlan|Et6/0

Et6/0       on               802.1q         trunking      1
Port        Vlans allowed on trunk
Et6/0       340
Port        Vlans allowed and active in management domain
Et6/0       340
Port        Vlans in spanning tree forwarding state and not pruned
Et6/0       none
```

Task 12

Add VLANs to the allowed list of the trunks, based on the following chart:

Policy Item	Trunk Interface	Between Switches	Add to the Allowed VLAN/s
1	E4/1	SW2 ←→ SW3	100
2	E5/0	SW3 ←→ SW5	200
3	E4/0	SW4 ←→ SW5	300

4	E6/0	SW2 ←→ SW5	400

Similar to Task 11, this task specifies several VLAN policies that make changes to the list of VLANs that should be allowed over a particular trunk link. Each policy lists the trunk interface and switch pair to which the policy applies and which VLANs are allowed or disallowed to be carried across the trunk.

The allowed VLAN list of a trunk link is configured using the **switchport trunk allowed vlan {{ add | except | remove } *vlan_list* | all | none }** command.

When used with the **add** parameter, the switch will add the indicated VLANs to the current list of allowed VLANs on the trunk link. This operation is examined in this task below.

Policy item 1: ←

The command **switchport trunk allowed VLAN add *vlan number*** can be used to add an additional VLAN to the already existing allowed VLAN list.

The following is the output of the **show int trunk | include Vlans|Et4/1** command. As seen, the only VLAN allowed over the E4/1 trunk interface between SW2 and SW3 is VLAN 230:

```
SW2#show interfaces trunk | include Vlans|Et4/1
Et4/1       on              802.1q        trunking      1
Port        Vlans allowed on trunk
Et4/1       230
Port        Vlans allowed and active in management domain
Et4/1       230
Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       230
```

To configure the first policy in the task:

To add VLAN 100 as an addition to the allowed VLAN list as the policy requires, the command switchport trunk allowed VLAN add 100 is configured under the E4/1 interface on SW2 and SW3:

On SW2 and SW3:

```
SWx(config)#interface e4/1
SWx(config-if)#switchport trunk allowed vlan add 100
```

To verify the configuration:

The show interfaces trunk | include Vlans|Et4/1 command outputs below, verify the above configuration where traffic for both 100 and 230 are now allowed to be carried over the trunk link E4/1 between SW2 and SW3:

On SW2:

```
SW2#show interfaces trunk | include Vlans|Et4/1

Et4/1       on              802.1q       trunking     1
Port        Vlans allowed on trunk
Et4/1       100,230
Port        Vlans allowed and active in management domain
Et4/1       100,230
Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       230
```

On SW3:

```
SW3#show interfaces trunk | include Vlans|Et4/1

Et4/1       on              802.1q       trunking     1
Port        Vlans allowed on trunk
Et4/1       100,230
Port        Vlans allowed and active in management domain
Et4/1       100,230
Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       none
```

Similar configurations as above are made on the respective switches for the remaining policies in this task below:

Policy item 2: ←

To configure the second policy in the task:

On SW3 and SW5:

```
SWx(config)#interface e5/0
SWx(config-if)#switchport trunk allowed vlan add 200
```

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk | include Vlan|Et5/0

Et5/0       on              802.1q       trunking     1
Port        Vlans allowed on trunk
Et5/0       200,350
Port        Vlans allowed and active in management domain
Et5/0       200,350
Port        Vlans in spanning tree forwarding state and not pruned
Et5/0       200,350
```

On SW5:

```
SW5#show interfaces trunk | include Vlan|Et5/0

Et5/0       desirable       n-802.1q     trunking     1
Port        Vlans allowed on trunk
Et5/0       200,350
Port        Vlans allowed and active in management domain
Et5/0       200,350
Port        Vlans in spanning tree forwarding state and not pruned

Et5/0       none
```

Policy item 3:

To configure the third policy in the task:

On SW4 and SW5:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk allowed vlan add 300
```

To verify the configuration:

On SW4:

```
SW4#show interfaces trunk | include Vlan|Et4/0

Et4/0       desirable       802.1q       trunking     1
Port        Vlans allowed on trunk
Et4/0       300,450
Port        Vlans allowed and active in management domain
```

```
Et4/0        300,450
Port         Vlans in spanning tree forwarding state and not pruned
Et4/0        300,450
```

On SW5:

```
SW5#show interfaces trunk | include Vlan|Et4/0

Et4/0        desirable        802.1q         trunking      1
Port         Vlans allowed on trunk
Et4/0        300,450
Port         Vlans allowed and active in management domain
Et4/0        300,450
Port         Vlans in spanning tree forwarding state and not pruned
Et4/0        none
```

Policy item 4:

To configure the fourth policy in the task:

On SW2 and SW5:

```
SWx(config)#interface e6/0
SWx(config-if)#switchport trunk allowed vlan add 400
```

To verify the configuration:

On SW2:

```
SW2#show interfaces trunk | include Vlan|Et6/0

Et6/0        on               802.1q         trunking      1
Port         Vlans allowed on trunk
Et6/0        250,400
Port         Vlans allowed and active in management domain
Et6/0        250,400
Port         Vlans in spanning tree forwarding state and not pruned
Et6/0        250,4000
```

On SW5:

```
SW5#show interfaces trunk | include Vlan|Et6/0

Et6/0        on               802.1q         trunking      1
```

```
Port        Vlans allowed on trunk
Et6/0       250,400
Port        Vlans allowed and active in management domain
Et6/0       250,400
Port        Vlans in spanning tree forwarding state and not pruned
Et6/0       250,400
```

Task 13

Remove VLANs from the allowed list of trunks, based on the following chart:

Policy Item	Trunk Interface	Between Switches	Allowed VLAN/s
1	E5/1	SW2 ←→ SW4	Remove 1, 4 – 10 ONLY
2	E5/1	SW3 ←→ SW5	Remove 2, 4 – 10 ONLY

This task, too, specifies several VLAN policies that make changes to the list of VLANs that should be allowed over a particular trunk link. Each policy lists the trunk interface and switch pair to which the policy applies and which VLANs are allowed or disallowed to be carried across the trunk.

The allowed VLAN list of a trunk link is configured using the switchport trunk allowed vlan {{ add | except | remove } *vlan_list* | all | none } command.

When used with the remove parameter, the switch will remove only the indicated VLANs from the current list of allowed VLANs on the trunk link. This operation is examined in this task below.

Policy item 1:

The show interfaces trunk | include Vlan|Et5/1 output from SW2 reveals that all VLANs are currently allowed to traverse the E5/1 trunk link between SW2 and SW4:

```
SW2#show interfaces trunk | include Vlan|Et5/1

Et5/1       on              802.1q        trunking      1
Port        Vlans allowed on trunk
Et5/1       1-4094
Port        Vlans allowed and active in management domain
Et5/1       1-10,100,200,230,240,250,300,340,350,400,450
Port        Vlans in spanning tree forwarding state and not pruned
Et5/1       1-10,100,200,230,240,250,300,340,350,400,450
```

To configure the first item in the task:

As seen above, all VLANs are allowed to traverse the E5/1 trunk link. Policy 1 of this task requires prohibiting traffic belonging to VLAN 1 and VLANs 4 through 10 from traversing this link. This is accomplished using the switchport trunk allowed vlan remove 1, 4-10 command on the E5/1 interface on SW2 and SW4 as shown below.

On SW2 and SW4:

```
SWx(config)#interface e5/1
SWx(config-if)#switchport trunk allowed vlan remove 1,4-10
```

To verify the configuration:

On SW2:

```
SW2#show interfaces trunk | include Vlan|Et5/1

Et5/1        on               802.1q         trunking      1
Port         Vlans allowed on trunk
Et5/1        2-3,11-4094
Port         Vlans allowed and active in management domain
Et5/1        2-3,100,200,230,240,250,300,340,350,400,450
Port         Vlans in spanning tree forwarding state and not pruned
Et5/1        2-3,100,200,230,240,250,300,340,350,400,450
```

On SW4:

```
SW4#show interfaces trunk | include Vlan |Et5/1

Et5/1        on               802.1q         trunking      1
Port         Vlans allowed on trunk
Et5/1        2-3,11-4094
Port         Vlans allowed and active in management domain
Et5/1        2-3,100,200,230,240,250,300,340,350,400,450
Port         Vlans in spanning tree forwarding state and not pruned
Et5/1        2-3,100,200,230,250,300,340,350,400,450
```

Policy item 2: ←

To configure the second policy in the task:

On SW3 and SW5:

```
SWx(config)#interface e5/1
SWx(config-if)#switchport trunk allowed vlan remove 2,4-10
```

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk | include Vlan|Et5/1

Et5/1       on              802.1q       trunking      1
Port        Vlans allowed on trunk
Et5/1       1,3,11-4094
Port        Vlans allowed and active in management domain
Et5/1       1,3,100,200,230,240,250,300,340,350,400,450
Port        Vlans in spanning tree forwarding state and not pruned
Et5/1       1,3,100,200,230,240,250,300,340,350,400,450
```

On SW5:

```
SW5#show interface trunk | include Vlan|Et5/1

Et5/1       auto            n-802.1q     trunking      1
Port        Vlans allowed on trunk
Et5/1       1,3,11-4094
Port        Vlans allowed and active in management domain
Et5/1       1,3,100,200,230,240,250,300,340,350,400,450
Port        Vlans in spanning tree forwarding state and not pruned
Et5/1       none
```

Task 14

Configure SW2, SW3, and SW5, based on the following chart:

Policy Item	Trunk Interface	Between Switches	Allowed VLAN/s
1	E4/0	SW2 ←→ SW3	None
2	E6/1	SW2 ←→ SW5	None

This task continues the theme, specifying several VLAN policies that make changes to the list of VLANs that should be allowed over a particular trunk link. Each policy lists the trunk interface and switch pair to which the policy applies and which VLANs are allowed or disallowed to be carried across the trunk.

The allowed VLAN list of a trunk link is configured using the switchport trunk allowed vlan {{ add | except | remove } *vlan_list* | all | none } command.

When using the base command with no additional parameters and none as the VLAN list, the switch will remove ALL VLANs from the allowed VLAN list on the trunk link. The following task demonstrates this function.

Policy item #1:

As seen in the show interfaces trunk | include Vlan|Et4/0 command output from SW2 below, all VLANs are currently allowed on the E4/0 trunk link between SW2 and SW3.

```
SW2#show interfaces trunk | include Vlan |Et4/0

Et4/0       on                802.1q         trunking      1
Port        Vlans allowed on trunk
Et4/0       1-4094
Port        Vlans allowed and active in management domain
Et4/0       1-10,100,200,230,240,250,300,340,350,400,450
Port        Vlans in spanning tree forwarding state and not pruned
Et4/0       1-10,100,200,230,240,250,300,340,350,400,450
```

To configure the first policy in the task:

Policy 1 states that no VLANs should be allowed on this trunk link. This can be achieved with the switchport trunk allowed vlan none command on the E4/0 interfaces on SW2 and SW3:

On SW2 and SW3:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk allowed vlan none
```

On SW2:

```
SW2#show interfaces trunk | include Vlan |Et4/0

Et4/0       on                802.1q         trunking      1
Port        Vlans allowed on trunk
Et4/0       none
Port        Vlans allowed and active in management domain
Et4/0       none
Port        Vlans in spanning tree forwarding state and not pruned
Et4/0       none
```

On SW3:

```
SW3#show interfaces trunk | include Vlan |Et4/0

Et4/0       on                802.1q         trunking      1
Port        Vlans allowed on trunk
Et4/0       none
Port        Vlans allowed and active in management domain
Et4/0       none
Port        Vlans in spanning tree forwarding state and not pruned
Et4/0       none
```

Policy item #2:

To configure the second policy in the task:

A similar policy as above is configured on the trunk link E6/1 between SW2 and SW5 to prevent any VLANs from traversing this link. The **show interfaces trunk | include Vlan|Et6/1** command output on both switches confirms the same:

On SW2 and SW5:

```
SWx(config)#interface e6/1
SWx(config-if)#switchport trunk allowed vlan none
```

To verify the configuration:

On SW2:

```
SW2#show interfaces trunk | include Vlan |Et6/1

Et6/1       on                802.1q         trunking      1
Port        Vlans allowed on trunk
Et6/1       none
Port        Vlans allowed and active in management domain
Et6/1       none
Port        Vlans in spanning tree forwarding state and not pruned
Et6/1       none
```

On SW5:

```
SW5#show interfaces trunk | include Vlan |Et6/1

Port        Vlans allowed on trunk
```

```
Et6/1         none

Port          Vlans allowed and active in management domain
Et6/1         none

Port          Vlans in spanning tree forwarding state and not pruned
Et6/1         none
```

Task 15

Configure SW2, SW4, and SW5, based on the following chart:

Policy Item	Trunk Interface	Between Switches	Allowed VLAN/s
1	E4/1	SW4 <-> SW5	All except 450
2	E5/1	SW2 <-> SW4	All except 240

This task specifies additional VLAN policies that make changes to the list of VLANs that should be allowed over a particular trunk link. Each policy lists the trunk interface and switch pair to which the policy applies and which VLANs are allowed or disallowed to be carried across the trunk.

The allowed VLAN list of a trunk link is configured using the **switchport trunk allowed vlan** {{ **add** | **except** | **remove** } *vlan_list* | **all** | **none** } command.

When used with the **except** parameter, the switch will allow all VLANs except for the VLANs included in the VLAN list. This operation is examined in this task below.

Policy item #1:

As seen below, currently all VLANs are allowed on the E4/1 trunk link between SW4 and SW5:

```
SW4#show interfaces trunk | include Vlan |Et4/1

Et4/1         desirable        802.1q         trunking      1
Port          Vlans allowed on trunk
Et4/1         1-4094
Port          Vlans allowed and active in management domain
Et4/1         1-10,100,200,230,240,250,300,340,350,400,450
Port          Vlans in spanning tree forwarding state and not pruned
Et4/1         1-10,100,200,230,240,250,300,340,350,400,450
```

To configure the first policy in the task:

Policy 1 in this task requires that the E4/1 trunk between SW4 and SW5 carry traffic for all VLANs except VLAN 450. This can be done with the **switchport trunk allowed vlan except 450** command on the E4/1 interfaces on SW4 and SW5 as seen below:

The following command allows all VLANs except 450:

On SW4 and SW5:

```
SWx(config)#interface e4/1
SWx(config-if)#switchport trunk allowed vlan except 450
```

To verify the configuration:

On SW4:

```
SW4#show interfaces trunk | include Vlan |Et4/1

Et4/1       desirable       802.1q        trunking      1

Port        Vlans allowed on trunk
Et4/1       1-449,451-4094
Port        Vlans allowed and active in management domain
Et4/1       1-10,100,200,230,240,250,300,340,350,400
Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       1-10,100,200,230,240,250,300,340,350,400
```

On SW5:

```
SW5#show interfaces trunk | include Vlan |Et4/1

Et4/1       auto            n-802.1q      trunking      1
Port        Vlans allowed on trunk
Et4/1       1-449,451-4094

Port        Vlans allowed and active in management domain
Et4/1       1-10,100,200,230,240,250,300,340,350,400

Port        Vlans in spanning tree forwarding state and not pruned
Et4/1       2-10,200,240,340,350
```

Policy item #2:

To configure the second policy in the task:

Similar configurations as the above are made to allow all VLANs except VLAN 240 on the E5/1 trunk link between SW2 and SW4. The **show int trunk | include Vlan|Et5/1** command output verifies the result of the configuration:

On SW2 and SW4:

```
Cat-x(config)#interface e5/1
Cat-x(config-if)#switchport trunk allowed vlan except 240
```

To verify the configuration:

On SW2:

```
SW2#show interfaces trunk | include Vlan |Et5/1

Et5/1         on                802.1q         trunking      1
Port          Vlans allowed on trunk
Et5/1         1-239,241-4094

Port          Vlans allowed and active in management domain
Et5/1         1-10,100,200,230,250,300,340,350,400,450

Port          Vlans in spanning tree forwarding state and not pruned
Et5/1         2-3,100,200,230,250,300,340,350,400,450
```

On SW4:

```
SW4#show interfaces trunk | include Vlan |Et5/1

Et5/1         on                802.1q         trunking      1
Port          Vlans allowed on trunk
Et5/1         1-239,241-4094

Port          Vlans allowed and active in management domain
Et5/1         1-10,100,200,230,250,300,340,350,400,450

Port          Vlans in spanning tree forwarding state and not pruned
```

```
Et5/1        1-10,100,200,230,250,300,340,350,400,450
```

Task 16

Configure SW3 and SW4, based on the following chart. You may override some of the previous tasks to accomplish this task.

Policy Item	Trunk Interface	Between Switches	Allowed VLAN/s
1	E6/0	SW3 ←→ SW4	All
2	E6/1	SW3 ←→ SW4	All

This task is the final specification of VLAN policies that make changes to the list of VLANs that should be allowed over a particular trunk link. Each policy lists the trunk interface and switch pair to which the policy applies and which VLANs are allowed or disallowed to be carried across the trunk.

The allowed VLAN list of a trunk link is configured using the **switchport trunk allowed vlan** {{ **add** | **except** | **remove** } *vlan_list* | **all** | **none** } command.

When used with no additional parameters and **all** as the VLAN list, the switch will allow all VLANs across the trunk link. This operation is examined in this task below.

Policy item #1:

To configure the first policy in the task:

This task requires permitting ALL Vlans on the E6/0 trunk link between SW3 and SW4. The **show interfaces trunk** output below shows the only VLAN allowed on this trunk link is VLAN 340.

```
SW3#show interfaces trunk | include Vlan |Et6/0

Et6/0       on                802.1q         trunking      1
Port        Vlans allowed on trunk
Et6/0       340
Port        Vlans allowed and active in management domain
Et6/0       340
Port        Vlans in spanning tree forwarding state and not pruned
Et6/0       340
```

On SW3 and SW4:

```
SWx(config)#interface e6/0
SWx(config-if)#switchport trunk allowed vlan all
```

To verify the configuration:

On SW3:

```
SW3#show interfaces trunk | include Vlan |Et6/0

Et6/0         on                 802.1q         trunking      1
Port          Vlans allowed on trunk
Et6/0         1-4094


Port          Vlans allowed and active in management domain
Et6/0         1-10,100,200,230,240,250,300,340,350,400,450

Port          Vlans in spanning tree forwarding state and not pruned
Et6/0         1-10,100,200,230,240,250,300,340,350,400,450
```

On SW4:

```
SW4#show interfaces trunk | include Vlan |Et6/0

Et6/0         on                 802.1q         trunking      1
Port          Vlans allowed on trunk
Et6/0         1-4094

Port          Vlans allowed and active in management domain
Et6/0         1-10,100,200,230,240,250,300,340,350,400,450

Port          Vlans in spanning tree forwarding state and not pruned
Et6/0         1-10,200,240,250,300,340,350,400,450
```

Task 17

Erase the config.text and vlan.dat files on SW1-5 and reload them before proceeding to the next task.

VLAN information on Cisco switches is stored in a separate file in the switch's flash storage known as the VLAN database. This means wiping the configuration of the device by removing the startup configuration file, is not enough to remove the VLAN configurations themselves. In order to wipe VLAN configurations from a switch, the VLAN database needs to be deleted from flash storage with the **delete** command. The file typically exists as **flash:vlan.dat**. The startup configuration file is then erased with the **erase startup-config** command and the switch is then rebooted.

Note: On virtual platforms such as EVE-ng, the VLAN.dat file can be found under the unix directory.

To follow the task and the VLAN.dat file, issue the following commands on SW2 through SW5 and then reload all the devices for the changes to effect:

On the first Switch:

```
Switch#delete vlan.dat-00009
Delete filename [vlan.dat-00009]?
Delete unix:/vlan.dat-00009? [confirm]

Switch#write erase
Erasing the nvram filesystem will remove all configuration files!
Continue? [confirm] ← Press Enter

Switch#reload
```

On SW2:

```
SW2#delete vlan.dat-00160
Delete filename [vlan.dat-00160]? ← The name may be different in your IOS
Delete unix:/vlan.dat-00160? [confirm]

SW2#write erase
Erasing the nvram filesystem will remove all configuration files!
Continue? [confirm] ← Press Enter

SW2#reload
```

On SW3:

```
SW2# delete vlan.dat-00176
Delete filename [vlan.dat-00176]? ← The name may be different in your IOS

Delete unix:/vlan.dat-00176? [confirm]

SW2#write erase
```

```
Erasing the nvram filesystem will remove all configuration files!
Continue? [confirm] ← Press Enter

SW2#reload
```

On SW4:

```
SW2#delete vlan.dat-00192
Delete filename [vlan.dat-00192]? ← The name may be different in your IOS
Delete unix:/vlan.dat-00192? [confirm]

SW2#write erase
Erasing the nvram filesystem will remove all configuration files!
Continue? [confirm] ← Press Enter

SW2#reload
```

On SW5:

```
SW2#delete vlan.dat-00208
Delete filename [vlan.dat-00208]? ← The name may be different in your IOS
Delete unix:/vlan.dat-00208? [confirm]

SW2#write erase
Erasing the nvram filesystem will remove all configuration files!
Continue? [confirm] ← Press Enter

SW2#reload
```

Task 18

Configure SW2 and SW3 based on the following policies:

- Configure the hostnames of Switch 2 and Switch 3 to be SW2 and SW3,
- Shut down all the ports on SW2 and SW3.
- Configure a dot1q trunk between SW2 and SW3 using port E4/0.
- Ensure that both switches belong to the VTP domain TST.

The policies above represent a good base configuration for the two switches and combine all of the tasks learned previously into one test. The interface range command is used extensively to make quick configurations to ports that share the same configuration state.

The ports are all shut down initially. Then, the E4/0 interface is configured as a static 802.1q trunking port on each switch. Finally, the VTP domain name is set to "TST".

On switch 2:

```
Switch(config)#hostname SW2
```

On switch 3:

```
Switch(config)#hostname SW3
```

On both switches:

```
SWx(config)#interface range e0/1-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-
3,e6/0-3
SWx(config-if-range)#shut
```

On SW2 and SW3:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk encapsulation dot1q
SWx(config-if)#switchport mode trunk
SWx(config-if)#no shut

SWx(config)#vtp domain TST
Changing VTP domain name from NULL to TST
```

To verify the configuration:

On SW3:

```
SW3#show vtp status | include Domain

VTP Domain Name                 : TST
```

Task 19

Configure VLAN 100 on SW2 and assign its E0/1 interface to this VLAN.

This task requires configuring the E0/1 interface on SW2 as an access port that belongs to VLAN 100. VLAN 100 currently does not exist on SW2. There are two ways to achieve this.

As seen earlier, one of the ways to create a VLAN on a switch with the **vlan** *vlan number* command. Alternatively, when configuring an access port for a particular VLAN with the **switchport access vlan** *vlan-number* command, IOS automatically creates the VLAN if it doesn't already exist. This can be seen below where SW2's E0/1 is first statically set to function as an access port with the **switchport mode access** command. The port is then assigned the VLAN 100 with the **switchport access vlan 100** command:

On SW2:

```
SW2(config)#interface e0/1
SW2(config-if)#switchport mode access
SW2(config-if)#switchport access vlan 100
% Access VLAN does not exist. Creating vlan 100
SW2(config-if)#no shut
```

Notice the message "**% Access VLAN does not exist. Creating vlan 100**". This is a confirmation that IOS recognized the indicated VLAN (100 in this case) was not already defined in the switch and automatically added it to the configuration.

The **show vlan brief | include VLAN** command output confirms the VLAN 100 creation along with the port assignment:

To verify the configuration:

On SW2:

```
SW2#show vlan brief | include VLAN

VLAN Name                             Status    Ports
100  VLAN0100                         active    Et0/1
```

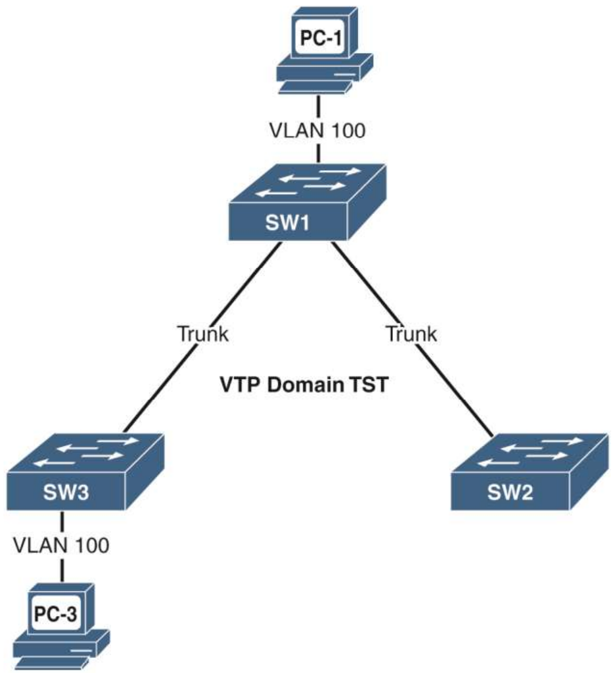
On SW3:

```
SW3#show vlan brief | include VLAN

VLAN Name                             Status    Ports
100  VLAN0100                         active
```

VTP Pruning

In Task 10, it was explained that VTP is a protocol that can be used to synchronize the VLAN databases between switches that participate in the same VTP domain. VTP uses three message types to achieve this synchronization: **summary advertisement**, **advertisement request**, and **subset advertisements**. This synchronization process makes it such that an administrator only needs to configure VLANs on a single VTP server switch and have those configurations propagated for synchronization to all remaining switches in the VTP domain.

VTP can be used for more than simple VLAN database synchronization. It can be used to reduce unnecessary broadcast flooded traffic originating in a specific VLAN from entering sections of the network that do not require that VLAN traffic. For example, the sample topology from Task 10 is examined:



In this topology, the interfaces connecting Switch-1 to Switch-2 and Switch-3 are configured as trunk links. All three switches belong to the same VTP domain, **TST**.

Previously, VLAN 100 was created on Switch-1. Switch-1 propagated information about this VLAN to Switch-2 and Switch-3 using VTP. Looking at the diagram, notice only Switch-1 and Switch-3 have interfaces connected to hosts in VLAN 100. Switch-2 does not have any interfaces connected to VLAN 100. Since Switch-2 has no interfaces in VLAN 100, there is no need for it to receive broadcast traffic originating from hosts in VLAN 100 connected to other switches in the network. Recall, ARP requests are broadcasted to all devices in the same VLAN. They are generated by end hosts to resolve the MAC address of a target device.

When PC-1 needs to reach PC-3 in the topology, it starts by sending a broadcast ARP frame to learn PC-3's MAC address. The following packet capture shows that the ARP frame was broadcasted to both Switch-2 and Switch-3:

On Switch-3:

```
Frame 4990: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
Ethernet II, Src: aa:bb:cc:00:04:20 (aa:bb:cc:00:04:20), Dst: Broadcast
(ff:ff:ff:ff:ff:ff)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: aa:bb:cc:00:04:20 (aa:bb:cc:00:04:20)
    Sender IP address: 100.1.1.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 100.1.1.2
```

On Switch-2

```
Frame 4134: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
Ethernet II, Src: aa:bb:cc:00:04:20 (aa:bb:cc:00:04:20), Dst: aa:bb:cc:00:05:10
(aa:bb:cc:00:05:10)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: aa:bb:cc:00:04:20 (aa:bb:cc:00:04:20)
    Sender IP address: 100.1.1.1
    Target MAC address: aa:bb:cc:00:05:10 (aa:bb:cc:00:05:10)
    Target IP address: 100.1.1.2
```

Switch-2 has received the broadcast traffic even though it will ultimately drop it because it does not have any hosts in that VLAN. The VLAN 100 broadcast in this case was unnecessarily broadcasted to Switch-2. This unnecessary broadcast was because Switch-1 has included VLAN 100 in its allowed VLAN list for the trunk link connected to Switch-2 as shown below:

```
Switch-1#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et0/0       on               802.1q         trunking      1
Et0/1       on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094
Et0/1       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,100,200
Et0/1       1,100,200

Port        Vlans in spanning tree forwarding state and not pruned
Et0/0       1,100
Et0/1       1,100 ! VLAN 100 is allowed on the trunk link to Switch-2
```

This inclusion extends VLAN 100's broadcast domain to Switch-2. One way to stop the unnecessary broadcast of VLAN 100 traffic to Switch-2 is to manually remove VLAN 100 from the Switch-1/Switch-2 trunk link by using the **switchport trunk allowed-vlan remove 100** command in interface configuration mode. This process, called manual pruning, is demonstrated extensively in previous tasks.

The problem with manual pruning is that if a host were connected to Switch-2 in VLAN 100, that VLAN would need to be manually added to the allowed VLAN list on the Switch-1/Switch-2 trunk link again. Multiply this process by hundreds of VLANs and potentially hundreds of clients moving around in the network, and it is clear that this manual pruning process can cause high administrative overhead.

This is where **VTP pruning** comes into play. VTP pruning is a feature of VTP that allows dynamic pruning of VLANs from trunk links where the VLAN traffic is not needed. The process utilizes a fourth VTP message type, called the **VTP membership advertisement**, or **VMA**. These are basically VTP Join/Prune messages.

The basic process for VTP pruning is that a switch sends a VMA for all VLANs for which it is interested in receiving traffic. The switch makes this determination based on whether or not it contains any interfaces that are currently associated with a VLAN that exists in the local switch's VLAN database.

Once again, referring back to the sample topology shown earlier, when VTP pruning is enabled on all the switches, they exchange VMAs with each other. Because Switch-3 is interested in receiving traffic for VLAN 100, its VMA includes VLAN 1 and VLAN 100, as shown in the following capture.

Note : VLAN 1 is pruning ineligible, which means this VLAN will always be included in the advertised active VLAN field and cannot be removed.

From Switch-3:

```
VLAN Trunking Protocol
    Version: 0x01
    Code: Join/Prune Message (0x04)
    Reserved: 00
    Management Domain Length: 3
    Management Domain: TST
    First VLAN ID: 0
    Last VLAN ID: 1007
    Advertised active (i.e. not pruned) VLANs
        VLAN: 1
        VLAN: 100
```

The **Advertised active (i.e. not pruned) VLANs** field indicates the VLANs for which Switch-3 is interested in receiving traffic. Any VLAN that is not assigned to an interface on Switch-3 is omitted from this list. For example, the VMA sent from Switch-2 to Switch-1 includes only VLAN 1. This is because Switch-2 does not have any interfaces assigned to VLAN 100, and thus it has no need for VLAN 100 traffic. Any traffic sent by Switch-1 to Switch-2 in VLAN 100 would inevitably be a futile effort since Switch-2 would end up dropping the traffic anyway.

From Switch-2:

```
VLAN Trunking Protocol
    Version: 0x01
    Code: Join/Prune Message (0x04)
    Reserved: 00
    Management Domain Length: 3
    Management Domain: TST
    First VLAN ID: 0
    Last VLAN ID: 1007
    Advertised active (i.e. not pruned) VLANs
```

```
       VLAN: 1
```

The following shows the state of the **show interface trunk** output on Switch-1. As a result of the VMA message exchanges, Switch-1 will now send traffic for VLAN 100 out its trunk link E0/0 to Switch-3 only.

```
Switch-1#show interfaces trunk

Port        Mode                Encapsulation  Status
Native vlan
Et0/0       on                  802.1q         trunking      1
Et0/1       on                  802.1q         trunking      1

Port        Vlans allowed on trunk
Et0/0       1-4094
Et0/1       1-4094

Port        Vlans allowed and active in management domain
Et0/0       1,100,200
Et0/1       1,100,200

Port        Vlans in spanning tree forwarding state and not
pruned
Et0/0       1,100 ! VLAN 1 and 100 allowed to Switch-2
Et0/1       1 ! Only VLAN 1 allowed to Switch-2
```

VTP pruning is disabled by default and can be enabled by using the **vtp pruning** command in global configuration mode. VTP keeps a list of VLANs that are allowed to be dynamically pruned; it is called the **pruning-eligible list** or the **pruning VLANs enabled list**, depending on the **show** command being used. This list can be seen using the **show interface switchport** command, as shown here:

```
Switch-1#show interfaces e0/0 switchport | include Pruning
Pruning VLANs Enabled: 2-1001
```

The pruning-eligible list can be modified much like the allowed VLAN list of a trunk link with the **switchport trunk pruning vlan** command and the following additional parameters:

```
Switch-1(config-if)#switchport trunk pruning vlan ?
  WORD    VLAN IDs of the allowed VLANs when this port is in
trunking mode
```

```
add     add VLANs to the current list
except  all VLANs except the following
none    no VLANs
remove  remove VLANs from the current list
```

The result of this command can be seen with the sample topology shown earlier. Before any configuration changes are made, the **show interface e0/1 pruning** command output on Switch-1 reveals that it is pruning VLAN 100 on the E0/1 trunk link connected to Switch-2. This is a result of the missing VLAN 100 in the VMA sent by Switch-2, as shown here:

```
Switch-1#show interface e0/1 pruning

Port                Vlans pruned for lack of request by
neighbor
Et0/1               100

Port                Vlan traffic requested of neighbor
Et0/1               1,100
```

Next, VLAN 200 is configured on Switch-2. The pruning-eligible list on Switch-2 is then modified to include only VLAN 200 with the **switchport trunk pruning vlan 200** command:

```
Switch-2(config-if)#vlan 200
Switch-2(config-vlan)#exit
Switch-2(config)#int e0/1
Switch-2(config-if)#switchport trunk pruning vlan 200
```

Notice how the pruning-eligible list on Switch-2 shows only VLAN 200:

```
Switch-2#show interfaces e0/1 switchport | include Pruning
Pruning VLANs Enabled: 200
```

The result of this configuration is that all other VLANs are now considered to be pruning ineligible. As a result, the VMA sent by Switch-2 to Switch-1 will include VLAN 100, thus preventing Switch-1 from pruning this VLAN off the trunk link connected to Switch-2. You can see this in the following packet capture, where VLAN 100 is included in the **Advertised active (i.e. not pruned) VLANs** field:

```
VLAN Trunking Protocol
```

```
    Version: 0x01
    Code: Join/Prune Message (0x04)
    Reserved: 00
    Management Domain Length: 3
    Management Domain: TST
    First VLAN ID: 0
    Last VLAN ID: 1007
    Advertised active (i.e. not pruned) VLANs
        VLAN: 1
        VLAN: 100
```

Task 20 through 27 below pertain to VTP pruning for the lab topology.

Task 20

Configure the switches such that they restrict flooded traffic to those trunk links the traffic must use to access the appropriate network device(s).

This task gives a good representation of a vague question. At the onset it can be difficult to figure out what exactly the task is asking. Understanding the results of prior configurations provides the proper context for this task.

In Task 19, VLAN 100 on SW2 and SW3 was configured. Of the two switches, only SW2 actually has an access port in VLAN 100, evidenced by the **show vlan brief | include VLAN** command output below.

On SW2:

```
SW2#show vlan brief | include VLAN

VLAN Name                                  Status    Ports
100  VLAN0100                              active    Et0/1
```

On SW3:

```
SW3#show vlan brief | include VLAN

VLAN Name                                  Status    Ports
100  VLAN0100                              active
```

Notice there are no ports indicated under the ports field on SW3. This simply indicates that SW3 does not have any ports configured for VLAN 100. With default settings on the E4/0 trunk port, SW3 will still receive all broadcast traffic sent to VLAN 100. With no ports configured for VLAN 100 on SW3, it is unnecessary for SW2 to send any traffic for VLAN 100 across the E4/0 trunk link to SW3 because it will ultimately be dropped.

The task's goal is to clean up this interaction by preventing unnecessary flooded traffic to SW3 for VLANs which it does not need. To accomplish this task, the VTP pruning feature is enabled.

The **show vtp status | include Prunning outtput** below, shows VTP pruning is by default turned off on both switches:

On SW2:

```
SW2#show vtp status |  include Prunning
VTP Pruning Mode                    : Disabled
```

On SW3:

```
SW3#show vtp status |  include Prunning
VTP Pruning Mode                    : Disabled
```

To enable VTP pruning, the **vtp pruning** command is issued in global configuration mode on SW2. Once enabled, SW2 will advertise this setting to the entire VTP domain, including SW3. The output of the command **show vtp status | include Prunning after configuration** verifies this:

On SW2:

```
SW2#vtp pruning
Pruning switched on
```

To verify the configuration:

```
SW2#show vtp status | include Prunning
VTP Pruning Mode                    : Enabled
```

On SW3:

```
SW3#show vtp status | include Prunning
VTP Pruning Mode                    : Enabled
```

Once VTP pruning has been enabled, all of the switches in the VTP domain begin sending VMA messages to their neighboring switches to indicate the VLANs from which they would like to receive traffic. The results of this interaction are shown on SW3 using the **show interface e4/0 pruning** command:

```
SW3#show interfaces e4/0 pruning

! Section 1

Port                    Vlans pruned for lack of request by neighbor
Et4/0                   none

! Section 2

Port                    Vlan traffic requested of neighbor
Et4/0                   1
```

The output above is broken down into two sections:

The first section of the **show interface pruning** output lists the VLANs the local switch has pruned off the trunk link. In the output above, SW3 is not pruning any VLANs from the E4/0 trunk link connected to SW2. This is indicated by the **none** keyword and is the result of SW3 receiving VMAs including all configured VLANs on the trunk link (in this case VLANs 1 and 100).

The second section of the **show interface pruning** output lists the VLANs for which the local switch has sent VMAs. Because SW3 does not have any active ports configured for VLAN 100, it does not include the VLAN in the VMA it sends to SW2. The only VLAN included in the VMA will be VLAN 1, and this is the reason the output above shows 1 under the VLAN traffic requested of neighbor section.

SW2's **show interface E4/0 pruning** output shows the other side of the pruning process. The first section shows the VLANs SW2 will prune off the trunk link to SW3, VLAN 100 in this case. SW2 prunes this because it has NOT received a VMA for VLAN 100 from SW3. As a result, SW2 will not send any VLAN 100 traffic out the E4/0 link to SW3. The second section reveals the VLANs for which SW2 expects to receive traffic from SW3, VLAN 1 and VLAN 100 as seen below. These are the VLANs for which SW2 has sent VMAs The complete output is shown below:

```
SW2#show interfaces e4/0 pruning

! Section 1

Port                    Vlans pruned for lack of request by neighbor
```

```
Et4/0                   100
! Section 2

Port                    Vlan traffic requested of neighbor
Et4/0                   1,100
```

Task 21

Configure VLANs 200, 300, 400, 500, and 600 on SW2 and ensure that these VLANs are propagated to SW3.

This task requires configuring VLANs 200, 300, 400, 500 and 600 on SW2. This is done with the **vlan** *vlan-number* command as seen below. Once again, it is important to issue the **exit** command after the vlan configuring to allow the VLANs to be added to the database:

On SW2:

```
SW2(config)#vlan 200,300,400,500,600
SW2(config-vlan)#exit
```

NOTE: In the above configuration if "**exit**" is not entered, the VLANs will not be propagated.

The **show vlan brief | include VLAN** command output from SW3 verifies that the above VLANs have been propagated via VTP to SW3:

On SW3:

```
SW3#show vlan brief | include VLAN

VLAN Name                             Status    Ports
100  VLAN0100                         active
200  VLAN0200                         active
300  VLAN0300                         active
400  VLAN0400                         active
500  VLAN0500                         active
600  VLAN0600                         active
```

To verify the configuration:

Since SW3 does not have any active ports configured for any of the above VLANs, it does not include these VLANs in its VMAs to SW2. The result is SW2 prunes off these VLANs from the E4/0 trunk link connected to SW3. This can be seen in the first section of the **show interface pruning** command output from SW2 below:

On SW2:

```
SW2#show interfaces pruning

! Section 1

Port          Vlans pruned for lack of request by neighbor
Et4/0         100,200,300,400,500,600


! Section 2

Port          Vlan traffic requested of neighbor
Et4/0         1,100
```

In addition, since SW2 has active ports in VLAN 100 and VLAN 1 cannot be pruned, these VLANs are included in its VMAs to SW3. This is reflected in the second section of the output above. The result of this can also be seen in the highlighted output below where SW3 prunes off VLANs 200, 300, 400, 500, and600 since it does not receive any VMAs for these VLANs from SW2:

On SW3:

```
SW3#show interfaces pruning

Port          Vlans pruned for lack of request by neighbor
Et4/0         200,300,400,500,600

Port          Vlan traffic requested of neighbor
Et4/0         1
```

Task 22

Configure the E3/3 interface of SW3 in VLAN 100.

This task requires assigning the E3/3 interface on SW3 to VLAN 100. The purpose of this task is to demonstrate that once SW3 assigns VLAN 100 to the E3/3 interface, it will include VLAN 100 in the VMA message sent to SW2. This will cause SW2 to unprune VLAN 100 from the E4/0 trunk link and flood any traffic for VLAN 100 over it.

The following first configures the E3/3 interface as an access port with the **switchport mode access** command. The port is then assigned to VLAN 100 with the **switchport access vlan 100** command:

On SW3:

```
SW3(config)#interface e3/3
SW3(config-if)#switchport mode access
SW3(config-if)#switchport access vlan 100
SW3(config-if)#no shut
```

As a result of the above configuration, **vlan traffic requested of neighbor** portion of the output below shows that SW3 expects to receive VLAN traffic for VLAN 1 and VLAN 100 over the trunk link:

Note: You may have to wait for 30 seconds for STP convergence:

On SW3:

```
SW3#show interfaces pruning


Port          Vlans pruned for lack of request by neighbor
Et4/0         200,300,400,500,600

Port          Vlan traffic requested of neighbor
Et4/0         1,100
```

This configuration can be verified on SW2 as well using the **show interfacespruning** command. The first section of the output below shows the VLANs pruned off the trunk link to SW3. The second section lists the VLANs SW2 has included in its VMAs to SW3, indicating the VLANs for which it wishes to receive traffic:

On SW2:

```
SW2#show interfaces pruning

Port                Vlans pruned for lack of request by neighbor
Et4/0               200,300,400,500,600

Port                Vlan traffic requested of neighbor
Et4/0               1,100
```

The first section of the above output reveals that VLAN 100 is no longer pruned.

Task 23

Configure the switches such that only VLAN 300 is pruned.

To begin solving this task the **show interface e4/0 pruning** command output from SW2 is examined below:

On SW2:

```
SW2#show interfaces e4/0 pruning

Port        Vlans pruned for lack of request by neighbor
Et4/0       200,300,400,500,600

Port        Vlan traffic requested of neighbor
Et4/0       1,100
```

As seen above, SW2 has pruned the VLANs 200, 300, 400, 500 and 600 from the E4/0 trunk link connected to SW3. This is because SW3 has not requested traffic for these VLANs.

The task requires that only VLAN 300 is pruned off the trunk link. This can be achieved by modifying the prune eligible list on both switches with the **switchport trunk pruning vlan 300** command. This command will configure the switches such that ONLY VLAN 300 is eligible for being pruned. The **show int erfacesinterfaces switchport** command verifies the prune eligible list as seen below:

On SW2 and SW3:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk pruning vlan 300
```

On SW2:

```
SW2#show interfaces e4/0 switchport | include Pruning
Pruning VLANs Enabled: 300
```

On SW3:

```
SW3#show interfaces e4/0 switchport | include Pruning
Pruning VLANs Enabled: 300
```

Both switches now exchange VMAs with each other that excludes VLAN 300 from the "Advertised active (i.e. not pruned) VLANs" list. As a result, the **show interfaces pruning** command output on SW2 and SW3 only shows VLAN 300 under the "Vlans pruned for lack of request by neighbor" section:

On SW2:

```
SW2#show interfaces e4/0 pruning

Port        Vlans pruned for lack of request by neighbor
Et4/0       300

Port        Vlan traffic requested of neighbor
Et4/0       1,100,200,400,500,600
```

On SW3:

```
SW3#show interfaces e4/0 pruning

Port        Vlans pruned for lack of request by neighbor
Et4/0       300

Port        Vlan traffic requested of neighbor
Et3/0       1,100,200,400,500,600
```

Task 24

Configure the switches such that VLAN 200 is also pruned. You should not use the command from the previous task to accomplish this task.

This task requires pruning off VLAN 200 as well from the trunk link E4/0 connecting SW2 and SW3. This can be done with the **add** parameter of the **switchport trunk pruning vlan** command. The **add** keyword results in adding VLAN 200 to the already existing VLAN pruning eligible list:

On SW2 and SW3:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk pruning vlan add 200
```

To verify the configuration:

Note: VLAN 200 is added to the list of pruned VLANs

On SW2:

```
SW2#show interfaces pruning

Port        Vlans pruned for lack of request by neighbor
Et4/0       200,300

Port        Vlan traffic requested of neighbor
Et4/0       1,100,400,500,600
```

On SW3:

```
SW3#show interfaces pruning

Port        Vlans pruned for lack of request by neighbor
Et4/0       200,300
```

Note: VLAN 200 is added to the list of pruned VLANs

```
Port        Vlan traffic requested of neighbor
Et4/0       1,100,400,500,600
```

Task 25

Configure SW2 and SW3 such that none of the VLANs are pruned.

This task requires configuring the switches such that none of the VLANs are eligible for pruning on the trunk link between SW2 and SW3. This can be done with the none parameter added to the switchport trunk pruning vlan command as seen below. The show interface e4/0 switchport command output also shows the prune eligible list to be NONE:

On Both Switches:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk pruning vlan none
```

To verify the configuration:

On SW3:

```
SW3#show interfaces e4/0 switchport | include Pruning

Pruning VLANs Enabled: NONE
```

On SW2:

```
SW2#show interfaces e4/0 pruning

Port         Vlans pruned for lack of request by neighbor
Et4/0        none
```

Note: None of the VLANs are pruned

```
Port         Vlan traffic requested of neighbor
Et4/0        1,100,200,300,400,500,600
```

On SW3:

```
SW3#show interfaces e4/0 pruning

Port         Vlans pruned for lack of request by neighbor
Et4/0        none
```

Note: None of the VLANs are pruned

```
Port         Vlan traffic requested of neighbor
Et4/0        1,100,200,300,400,500,600
```

Task 26

Configure SW2 and SW3 such that all configured VLANs in the VLAN database are pruned.

This task requires ALL VLANs in the VLAN database to be pruned from the trunk link E4/0 between SW2 and SW3. The current VLAN database holds VLANs 1,100,200,300,400,500,600. To remove these VLANs the command **switchport trunk pruning vlan 1,100,200,300,400,500,600** is issued on SW2 and SW3's E4/0 interface. However, such a configuration results in the following error message **Command rejected: Bad VLAN pruning list:**

On Both Switches:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk pruning vlan 1,100,200,300,400,500,600
```

You should receive the following errors:

```
Command rejected Bad VLAN pruning list.
```

As already mentioned, VLAN 1 is prune ineligible. The above error message was generated because VLAN 1 cannot be pruned. To bypass this error the command is entered again but this time without specifying VLAN 1:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk pruning vlan 100,200,300,400,500,600
```

To verify the configuration:

Observing the output below we notice all the above listed VLANs except VLAN 100 has been pruned off the trunk link E4/0 between SW2 and SW3. The reason for this is that Task 19 and Task 22 of this section configured access ports in VLAN 100 on SW2 and SW3. In other words, VLAN 100 cannot be pruned because both the switches have port membership in VLAN 100.

On SW2:

```
SW2#show interfaces e4/0 pruning
```

```
Port          Vlans pruned for lack of request by neighbor
Et4/0         200,300,400,500,600

Port          Vlan traffic requested of neighbor
Et4/0         1,100
```

To summarize the points in this task:

- VLAN 1 cannot be pruned.
- If the local switch has port membership in a given VLAN, that VLAN cannot be pruned.

Task 27

Configure SW2 and SW3 such that VLAN 200 is no longer pruned; do not use a command that was used before to accomplish this task.

This task requires removing VLAN 200 from the existing VLAN prune eligible list. This can be done with the **switchport trunk pruning vlan remove 200** **command** on the trunk interface E4/0 on SW2 and SW3 as seen below:

On Both Switches:

```
SWx(config)#interface e4/0
SWx(config-if)#switchport trunk pruning vlan remove 200
```

To verify the configuration:

The **switchport interface pruning** command output below verifies the above configuration. VLAN 200 is removed from the Vlans pruned for lack of request by neighbor section for the trunk link between SW2 and SW3:

On SW2:

```
SW2#show interfaces e4/0 pruning

Port          Vlans pruned for lack of request by neighbor
Et4/0         300,400,500,600

Port          Vlan traffic requested of neighbor
```

```
Et4/0         1,100,200
```

Note: VLAN 200 was removed from the list of VLANs being pruned.

On SW3:

```
SW3#show interfaces e4/0 pruning

Port          Vlans pruned for lack of request by neighbor
Et4/0         300,400,500,600

Port          Vlan traffic requested of neighbor
Et4/0         1,100,200
```

NOTE: VLAN 200 was removed from the list of VLANs being Pruned.

With the "**switchport trunk pruning vlan remove**" command, you are asking the switch to remove the specified VLAN from the list of pruned VLANs, meaning "**Do not prune this VLAN**".

Task 28

Erase the vlan.dat and config.text files and reload the switches before proceeding to the next lab.

VLAN information on Cisco switches is stored in a separate file in the switch's flash storage known as the VLAN database. This means wiping the configuration of the device by removing the startup configuration file, is not enough to remove the VLAN configurations themselves. In order to wipe VLAN configurations from a switch, the VLAN database needs to be deleted from flash storage with the **delete** command. The file typically exists as **flash:vlan.dat**. The startup configuration file is then erased with the **erase startup-config** command and the switch is then rebooted.

Note: On virtual platforms such as EVE-ng, the VLAN.dat file can be found under the unix directory.

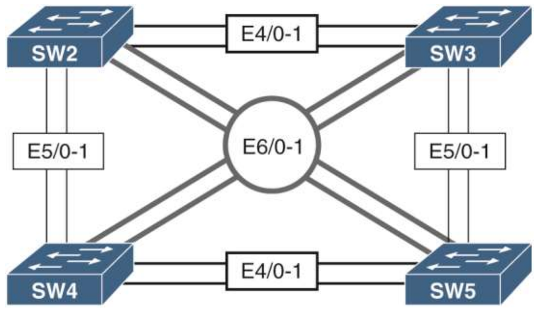
To follow the task and the VLAN.dat file, issue the following commands on the switches and then reload all the devices for the changes to effect:

```
SWx#delete vlan.dat-00010
Delete filename [vlan.dat-00010]?
Delete unix:/vlan.dat-00010? [confirm]
```

→ This name may differ

```
SWx#erase startup-config
Erasing the nvram filesystem will remove all configuration files!
Continue? [confirm]
[OK]
Erase of nvram: complete
```

Lab 2
Configuring EtherChannels



Task 1

Configure the hostname of the switches based on the provided diagram. Ensure that all the ports of these four switches are in shutdown mode. Configure these four switches in a VTP domain called **TST**.

The following first uses the **hostname** command on each switch to set the hostnames as shown in the diagram. The **interface range** command is used on all switches to shutdown all the interfaces on the switch. The VTP domain name is set manually on each switch with the **vtp domain TST** command:

On the second Switch:

```
Switch(config)#hostname SW2


SW2(config)#interface range e0/1-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-
3,e6/0-3
SW2(config-if-range)#shut
```

```
SW2(config)#vtp domain TST

Changing VTP domain name from NULL to TST
```

On the third Switch:

```
Switch(config)#hostname SW3

SW3(config)#interface range e0/1-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-
3,e6/0-3
SW3(config-if-range)#shut

SW3(config)#vtp domain TST

Changing VTP domain name from NULL to TST
```

On the fourth Switch:

```
Switch(config)#hostname SW4

SW4(config)#interface range e0/1-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3
SW4(config-if-range)#shut

SW4(config)#vtp domain TST

Changing VTP domain name from NULL to TST
```

On the fifth Switch:

```
Switch(config)#hostname SW5

SW5(config)#interface range e0/1-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3
SW5(config-if-range)#shut

SW5(config)#vtp domain TST

Changing VTP domain name from NULL to TST
```

Task 2

Configure ports E4/0 and E4/1 on SW2 and SW3 as trunk links, using an industrystandard protocol. These links should appear to Spanning Tree Protocol as a single link. If one of the links fails, the traffic should use the other link, without any interruption. The ports on SW2 should be configured such that they only respond to PAgP packets and never start the negotiation process.

The task above makes mention to a very specific problem the current topology has. Looking at the topology diagram, SW2 and SW3 have redundant links connecting the two switches, E4/0 and E4/1. In normal network operation, Spanning-Tree Protocol (STP) will only allow one of these two links to be in the forwarding mode. This move by STP effectively halves the available bandwidth between the two switches. The other link will only be used if the primary fails and after slowly transitioning through the STP states.

In order to allow both links to be used simultaneously, the switch needs to be able to represent them as a single path to STP. This can be accomplished with **EtherChannel Bundling.** An EtherChannel bundle is a collection of individual Ethernet links. These links are represented by a single logical interface known as the **Port Channel Interface.** The port channel interface holds all of the configuration information for the bundled links. It is also the interface that is presented to STP by the switch instead of the individual links that make up the EtherChannel bundle. Traffic is load balanced between the available links within the EtherChannel based on a load-balancing method.

Using EtherChannel, fault-tolerant, high-speed links can be created between switches. An EtherChannel can consist of up to 8 bundled links. If a single link fails, traffic forwarding continues across the remaining links. The downside to EtherChannel is the configuration. In order to function properly, all links within the bundle must have identical configurations with regards to trunk mode, encapsulation, speed and duplex settings.

For proper EtherChannel operation, both devices on either side of the EtherChannel link must agree that the EtherChannel should exist between them. These hurdles make managing EtherChannel configurations somewhat difficult. The reason is each device may send frames from the same ethernet source MAC address across different links. If the far-end switch has not properly bundled the link, it may cause MAC address flapping. Additionally, loops could form because STP BPDUs are only sent out of one link in the bundle to the far-end switch. This means the far-end switch would only receive a BPDU on one of its ports, making it assume the rest of the bundled ports are not connected to a neighboring bridge resulting in it making its port Designated, forming a layer 2 loop.

To help with configuration consistency amongst member interfaces and with the remote EtherChannel endpoint, two EtherChannel bundling protocols exist: **Port Aggregation Protocol (PAgP)** and **Link**

Aggregation Control Protocol (LACP). PAgP is a Cisco Proprietary protocol whereas LACP is an industry standard, defined in the IEEE 802.3ad specification. Once configured, the EtherChannel protocol handles the following tasks:

- Ensures member interfaces have compatible configurations to participate in the EtherChannel bundle
- Ensures the far side switch agrees that the EtherChannel link exists between the two switches over the proper connected interfaces.

If the EtherChannel protocol determines incompatible configurations on member interfaces, the EtherChannel protocol will suspend the interface from participating in the EtherChannel bundle until the configurations are made consistent. If the far end switch does not agree to the EtherChannel link, the EtherChannel protocol will not bring the EtherChannel up on the local switch, allowing the member interfaces to operate as independent interfaces.

Individual interfaces are configured to participate in an EtherChannel bundle using the **channel-group** command. This command has five different options for configuring EtherChannel:

1. **ON** : Forces interfaces to operate as an EtherChannel without the use of PAgP or LACP. This is also considered to be manual configuration.
2. **ACTIVE** : Enables LACP active negotiation for EtherChannel bundling with the remote interface
3. **PASSIVE** : Enables passive LACP negotiation. The interface will respond to LACP frames sent by the remote interface but will not send any first.
4. **DESIRABLE** : Enables PAgP active negotiation for the EtherChannel bundling with the remote interface.
5. **AUTO** - Enables passive PAgP negotiation for the EtherChannel bundling with the remote interface. The interface will respond to PAgP frames sent by the remote interface but will not send any first.

In EtherChannel bundles, both sides of the link must agree on the EtherChannel protocol (PAgP, LACP, or none) and at least one side of the link must be configured to initiate the bundling process. The following table indicates the different combinations of interface configurations and whether or not an EtherChannel bundle will be formed:

If SW2 is configured in	If SW3 is configured in	Will an EtherChannel be	Protocol used

		established	
Desirable	Desirable	YES	PAgP
Desirable	Auto	YES	PAgP
Auto	Auto	NO	-
Active	Active	YES	LACP
Active	Passive	YES	LACP
Passive	Passive	NO	-
On	On	YES	None
ON	Auto	NO	-
On	Desirable	NO	-
On	Passive	NO	-
On	Active	NO	-
Desirable	Active	NO	-
Desirable	Passive	NO	-
Auto	Passive	NO	-

Looking at the table above, the basic conclusion is in order for the EtherChannel bundle to form, one of two things must be true:

1. **If manual configuration is being used, both sides should be configured in the ON mode.**
2. **If PAgP or LACP are in use, at least one side should be configured to actively negotiate an EtherChannel.**

For example, from the table, if one side is set to Auto and the other is set to Passive, they not only do not agree on EtherChannel protocol (Auto indicates PAgP operation and Passive indicates LACP operation) but neither side will actively attempt to bring up the EtherChannel. Instead, a combination such as one side set to Desirable and the other set to Auto will work. In this case, both sides agree to PAgP operation and the side set to Desirable will begin the EtherChannel negotiations.

The configuration of EtherChannels should be done in a certain order. The following is my recommendation of the order when setting up EtherChannels:

Step 1: Use the **default interface** command for all the interfaces that are to be bundled up into the same port channel. The default interface command resets the interface back to their default values. Additionally, for ease of configuration, the **default interface range** command can be used to reset multiple interfaces to their default state.

Step 2: Configure the EtherChannel by issuing the **channel-group** *channel-number* **mode** command to the physical interfaces. This will result in automatic port-channel interface creation. The channel-number value does not have to match on both sides.

Step 3: Configure the trunk related parameters directly on the port-channel interface configuration mode

Step 4: Reset the ports in the group by first issuing the **shutdown** command followed by the **no shutdown** command.

Task 2 requires bundling up the E4/0 and E4/1 interface on SW2 and SW3 into an EtherChannel. It specifically requires the use of PAgP as the negotiation protocol. In addition, the task further states that SW2 should never start the negotiation process. Instead, it should wait until it receives PAgP frames from SW3 to form a bundle. This means, the port-channel on SW2 should be configured in the **auto** mode. In this mode, the switch passively waits for PAgP frames from the far-end switch, SW3. The port-channel on SW3 will be configured in **desirable** mode. With this configuration, SW3 will be actively attempting to form a bundle with SW2 by sending out PAgP frames.

Finally, the configured port-channel between SW2 and SW3 will be used to carry traffic for multiple VLANs. This means the port-channel between SW2 and SW3 needs to be configured as a trunk link. The trunk parameters as specified in the task states to use the industry standard encapsulation protocol 802.1q. This is done with the **switchport trunk encapsulation dot1q** command under the port-channel interfaces on SW2 and SW3. The port-channel is then manually set to function as a trunk port with the **switchport mode trunk** command.

All of the above configurations are made on SW2 and SW3 below in the order specified above:

Step One:

On SW2:

```
SW2(config)#default interface range e4/0-1
```

```
SW2(config)#interface range e4/0-1
SW2(config-if-range)#no shut
```

Step Two:

```
SW2(config)#interface range e4/0-1
SW2(config-if-range)#channel-group 23 mode auto
```

You should see the following console messages:

```
Creating a port-channel interface Port-channel 23

%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet4/0, changed
state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet4/1, changed
state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet4/0, changed
state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet4/1, changed
state to up
```

Note: The interface port-channel 23 is created automatically:

```
SW2#show run interface port-channel 23 | begin interface

interface Port-channel23
end
```

Step Three:

```
SW2(config)#interface port-channel 23
SW2(config-if)#switchport trunk encapsulation dot1q
SW2(config-if)#switchport mode trunk
```

On SW3:

```
SW3(config)#default interface range e4/0-1

SW3(config)#interface range e4/0-1
SW3(config-if-range)#channel-group 32 mode desirable

SW3(config)#interface port-channel 32
SW3(config-if)#switchport trunk encapsulation dot1q
SW3(config-if)#switchport mode trunk
```

Step Four:

On SW2 and SW3:

```
SWx(config-if)#interface range e4/0-1
SWx(config-if-range)#shut
SWx(config-if-range)#no shut
```

To verify the configuration:

The result of the above configuration can be verified in the **show interfaces trunk** output below where the port channel between SW2 and SW3 functions as a 802.1q trunk:

On SW2:

```
SW2#show interface trunk

Port        Mode             Encapsulation  Status        Native vlan
Po23        on               802.1q         trunking      1

Port        Vlans allowed on trunk
Po23        1-4094

Port        Vlans allowed and active in management domain
Po23        1

Port        Vlans in spanning tree forwarding state and not pruned
Po23        1
```

On SW3:

```
SW3#show interface trunk

Port        Mode             Encapsulation  Status        Native vlan
Po32        on               802.1q         trunking      1

Port        Vlans allowed on trunk
Po32        1-4094

Port        Vlans allowed and active in management domain
Po32        1

Port        Vlans in spanning tree forwarding state and not pruned
```

```
Po32        1
```

The show interfaces e4/0 switchport | include Operational Mode command shows the Operational Mode of the member interfaces of the EtherChannel bundle have been set to trunk with the added mention that they belong to a member of a PortChannel interface:

On SW2:

```
SW2#show interface e4/0 switchport | include Operational Mode

Operational Mode: trunk (member of bundle Po23)

SW2#show interface e4/1 switchport | include Operational Mode

Operational Mode: trunk (member of bundle Po23)
```

On SW3:

```
SW3#show interface e4/0 switchport | include Operational Mode

Operational Mode: trunk (member of bundle Po32)

SW3#show interface e4/1 switchport | include Operational Mode

Operational Mode: trunk (member of bundle Po32)
```

Another result of this configuration is how the switch perceives the two interfaces. Notice in the show interfaces trunk output above only a single interface, the PortChannel interface, was listed as operating in trunk mode. This is because the switch no longer considers each individual link as separate, but rather identifies them as a single logical link represented by the PortChannel interface. This treatment extends to how the switch presents the link to Spanning-Tree Protocol. The **show spanning-tree vlan 1** output from SW2 below confirms that the switch lists the PortChannel interface and not the individual links (e4/0 and e4/1) as a link that is running the STP instance:

```
SW2#show spanning-tree vlan 1

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769
             Address     aabb.cc00.0a00
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
```

```
            Address     aabb.cc00.0a00
            Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
            Aging Time  15  sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Po23                Desg FWD 56        128.65   P2p
```

Task 3

Configure ports E5/0 and E5/1 on SW2 and SW4 as trunk links, using an industrystandard protocol. These links should appear to Spanning Tree Protocol as a single link.
If one of the links fails, the traffic should use the other link, without any interruption.
These ports should *not* negotiate an etherchannel by exchanging **LACP** or **PAgP** protocols to accomplish this task.

This task requires EtherChannel bundling of the redundant links E5/0-1 connecting SW2 and SW4. The configuration to establish the port-channel between SW2 and SW4 follows the configuration steps stated earlier.

First, the E5/0-1 interfaces on SW2 and SW4 are reset to their default state with the **default interface range e5/0-1** command. The **no shut** command is then used to bring these interfaces up. These interfaces are then assigned to their respective EtherChannel groups with the **channel-group** *channel-group-number* command.

Since the task restricts the use of any dynamic negotiation protocol for port-channel establishment, the **on** mode will be used to manually configure the port-channel between the switches. The port-channel between SW2 and SW4 is then configured for a static 802.1q trunk with the **switchport trunk encapsulation dot1q** and **switchport mode trunk** command. The physical interfaces are then reset by first shutting them down with the **shut** command and then bringing them up with the no **shut** command:

On SW2:

Step One:

```
SW2(config)#default interface range e5/0-1

SW2(config)#interface range e5/0-1
SW2(config-if-range)#no shut
```

Step Two:

```
SW2(config)#interface range e5/0-1
SW2(config-if-range)#channel-group 24 mode on
```

Step Three:

```
SW2(config-if-range)#interface port-channel 24
SW2(config-if)#switchport trunk encapsulation dot1q
SW2(config-if)#switchport mode trunk
```

On SW4:

Step One:

```
SW4(config)#default interface range e5/0-1

SW4(config)#interface range e5/0-1
SW4(config-if-range)#no shut
```

Step Two:

```
SW4(config)#interface range e5/0-1
SW4(config-if-range)#channel-group 42 mode on
```

Step Three:

```
SW4(config-if-range)#interface port-channel 42
SW4(config-if)#switchport trunk encapsulation dot1q
SW4(config-if)#switchport mode trunk
```

Step Four:

On SW2 and SW4

```
SWx(config)#interface range e5/0-1
SWx(config-if-range)#shut
SWx(config-if-range)#no shut
```

To verify the configuration:

The **show interface trunk** command output from SW2 and SW4 verifies the 802.1q static trunk port-channel between them:

```
SW2#show interfaces trunk | include 802

Po23        on                802.1q        trunking       1
Po24        on                802.1q        trunking       1
```

On SW4:

```
SW4#show interfaces trunk | include 802

Port        Mode             Encapsulation  Status        Native vlan
Po42        on               802.1q         trunking      1
```

Notice the **show etherchannel protocol** command output from SW2 below. The highlighted output in black is the port-channel set up between SW2 and SW3 in task 2. The **Protocol** portion of this output shows the protocol used for this EtherChannel negotiation is PAgP. In contrast, the **Protocol** portion of the output highlighted in green shows **ON**, meaning neither LACP or PAgP was used to negotiate this port channel:

```
SW2#show etherchannel protocol

      Channel-group listing:
               ----------------------

Group: 23
----------
Protocol:  PAgP

Group: 24
----------
Protocol:   -  (Mode ON)
```

Note: PAgP is used for EtherChannel negotiation

Note: PAgP or LACP is not in use

Finally, the **show EtherChannel summary** command provides a lot of information about the operational state of the port channels. The output from SW4 is shown as an example. The **SU** in parenthesis in the highlighted output below indicates that Po42 is a **Layer 2** port-channel and is currently **in use**. The P indicated against the physical interfaces show that these interfaces have been bundled to form the port-channel 42:

```
SW4#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      N - not in use, no aggregation
        f - failed to allocate aggregator

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

        A - formed by Auto LAG

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

Group  Port-channel  Protocol    Ports
------+-------------+-----------+--------------------
42     Po42(SU)          -        Et5/0(P)    Et5/1(P)
```

Task 4

Ensure that all the EtherChannels created on SW2 are load balanced based on the destination MAC address.

When forwarding frames out of an EtherChannel bundle, the switch needs to determine which member interface should be used to transmit the frame. The switch cannot send some bits of a frame over one interface and the remaining over another. A single frame should be sent, in full, over a single interface.

The switch determines which individual link of an EtherChannel will be used to transmit the frame based on a hashing algorithm. This algorithm is called the EtherChannel load-balancing method and is based on the source/destination MAC address, source/destination IP address, or both depending on the setting and switching platform being used. This algorithm is only applied whenever the switch determines an incoming or self-generated frame should be forwarded out of an EtherChannel interface.

In order to achieve maximum load balancing between member interfaces, the EtherChannel load-balancing method needs to be set to provide diverse hash value results. For instance, if the EtherChannel bundle were connected to a single end device, using source MAC or IP address as the load-balancing method would generate in the same hash result for all traffic originating from the device. This is because the source MAC and IP address would be the same for all traffic sent by the single device. The result is only a single link of the EtherChannel bundle would be utilized. Instead, in such a situation, a method that incorporates destination MAC or IP addressing would be preferred as the variety would be much greater for traffic originating from that device.

The **port-channel load-balance ?** command options below reveals the load balancing options available on these switches. These options may vary between different platforms.

```
SW2(config)#port-channel load-balance ?
  dst-ip       Dst IP Addr
  dst-mac      Dst Mac Addr
  src-dst-ip   Src XOR Dst IP Addr
  src-dst-mac  Src XOR Dst Mac Addr
  src-ip       Src IP Addr
  src-mac      Src Mac Addr
```

The following describes briefly each of the load balancing options presented above:

1. **Destination IP address** - With this option, when packets are forwarded to an EtherChannel, they are distributed across the interfaces in the channel based on the **destination IP address** of the incoming or self-generated frame.
2. **Destination MAC address** - With this option, when packets are forwarded to an EtherChannel, they are distributed across the interfaces in the channel based on the **destination MAC address** of the incoming or self-generated frame.
3. **Source and Destination IP address** - With this option, when packets are forwarded to a EtherChannel, they are distributed across the interfaces in the channel based on the **source and destination IP address pair** of the incoming or self-generated frame.
4. **Source and Destination MAC address** - With this option, when packets are forwarded to a EtherChannel, they are distributed across the interfaces in the channel based on the **source and destination MAC address pair** of the incoming or self-generated frame.
5. **Source IP address** - With this option, when packets are forwarded to a EtherChannel, they are distributed across the interfaces in the channel based on the **source IP address** of the incoming or self-generated frame.
6. **Source MAC address** - With this option, when packets are forwarded to a EtherChannel, they are distributed across the interfaces in the channel based on the **source MAC address** of the incoming or self-generated frame.

This task requires the load-balancing method on SW2 for all its port channels to be based on the **destination MAC address**. The default load balancing method in use can be checked with the **show etherchannel load-balance** command. As seen below, SW2 by default load balances on the Source and Destination IP address pair:

```
SW2#show EtherChannel load-balance
```

```
EtherChannel Load-Balancing Configuration:
        src-dst-ip
```

The task specifies that the switches should use destination MAC addresses as input to the hashing algorithm.To modify the load balancing method to the specified destination MAC mode, the command **port-channel load-balance dst-mac** is issued on SW2. The **show etherchannel load-balance** output verifies this configuration change:

To configure the load balancing based on the destination MAC addresses:

```
SW2(config)#port-channel load-balance dst-mac
```

To verify the configuration:

```
SW2#show etherchannel load-balance

EtherChannel Load-Balancing Configuration:

        dst-mac

EtherChannel Load-Balancing Addresses Used Per-Protocol:
   Non-IP: Destination MAC address
     IPv4: Destination MAC address
     IPv6: Destination MAC address
```

Since the command is entered in the global configuration mode, it effects all EtherChannel bundles created on the local switch.

Task 5

Configure ports E6/0 and E6/1 on SW3 and SW4 as a single Layer 3 link; SW3 should be configured with the IP address 34.1.1.3/24, and SW4 should be configured with the IP address 34.1.1.4/24. These ports should not negotiate LACP or PAgP.

This task requires bundling the E6/0-1 interfaces connecting SW3 and SW4 into a Layer 3 port-channel. A layer three port channel requires a different configuration flow compared to the Layer 2 port channels configured earlier in this section. With Layer 2 port channels, the port channel interface was created automatically when the channel-group command was issued on the interfaces. Layer 3 port channels must first be defined by the interface port-channel command and then enabled for Layer 3 operation with the no switchport command. The member interfaces are then associated with the configured Layer 3 port channels after enabling Layer 3 operations on them individually.

Configuration order of Etherchannels

Configuration of Layer 3 Etherchannels should be done in a certain order. Following is my recommendation of the order when setting up Layer 3 etherchannels:

Step 1 : Use the **default interface** command to reset the member interfaces to their default state

Step 2 : Configure the port-channel with the **interface port-channel** *channel-group-number* command. Issue the **no switchport** for this port-channel to disable Layer 2 operations and enable Layer 3 operations. Assign an Ip address to this port-channel

Step 3 : Issue the **no switchport** command under the physical interfaces that are to be bundled up into the Layer 3 port channel. Assign the port-channel created in step 2 to the physical interfaces with the **channel-group** *channel-group-number* **mode** command (the *channel-group-number* should match the *channel-group-number* used to configure the port-channel interface in the **interface port-channel** *channel-group-number* command).

Step 4 : Flap the physical interfaces with the **shutdown** and then a **no shut** command

The above steps can be examined in the below configuration that completes this task by creating a Layer 3 EtherChannel between SW3 and SW4:

On SW3:

Step One:

```
SW3(config)#default interface range e6/0-1
```

Step Two:

```
SW3(config)#interface port-channel 34
SW3(config-if)#no switchport
SW3(config-if)#ip address 34.1.1.3 255.255.255.0
```

Step Three:

```
SW3(config)#interface range e6/0-1
SW3(config-if-range)#no switchport
SW3(config-if-range)#channel-group 34 mode on
```

Step Four:

```
SW3(config)#interface range e6/0-1
SW3(config-if-range)#shut
SW3(config-if-range)#no shut
```

On SW4:

Step One:

```
SW4(config)#default interface range e6/0-1
```

Step Two:

```
SW4(config)#interface port-channel 43
SW4(config-if)#no switchport
SW4(config-if)#ip address 34.1.1.4 255.255.255.0
```

Step Three:

```
SW4(config)#interface range e6/0-1
SW4(config-if-range)#no switchport
SW4(config-if-range)#channel-group 43 mode on
```

Step Four:

```
SW4(config)#interface range e6/0-1
SW4(config-if-range)#shut
SW4(config-if-range)#no shut
```

To verify and test the configuration:

Following the above configuration, the **show etherchannel summary** can be used to verify the Layer 3 port-channel. The example output below from SW4 highlights the relevant parts. Notice the **RU** in parenthesis against the Po43. The **R** indicates that Po43 is a Layer 3 port-channel and is currently in use as indicated with the **U**. The member physical interfaces once again have **P** assigned to them indicating that these interfaces are bundled to form the Layer 3 port-channel 43:

```
SW4#sh etherc 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      N - not in use, no aggregation
        f - failed to allocate aggregator

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

        A - formed by Auto LAG


Number of channel-groups in use: 2
Number of aggregators:           2

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------
42        Po42(SU)        PAgP       Et5/0(P)    Et5/1(P)
43        Po43(RU)         -         Et6/0(P)    Et6/1(P)

SW4#show ip interface brief | exclude unassigned
Interface              IP-Address      OK? Method Status                Protocol
Port-channel43         34.1.1.4        YES manual up                    up

SW4#ping 34.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 34.1.1.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/1 ms
```

Task 6

Erase the startup configuration and vlan.dat files before proceeding to the next lab.

Lab 3
An STP Walk Through

Spanning-tree Technology Introduction Lab

This section is designed to teach basic to advanced concepts of Spanning Tree Protocol-.

It utilizes a common topology over which each version of Spanning Tree Protocol is configured with a given set of requirements and restraints. The requirements and restraints are engineered to explain the behaviors of each version of Spanning Tree Protocol, highlighting the important limitations and enhancements of each feature.

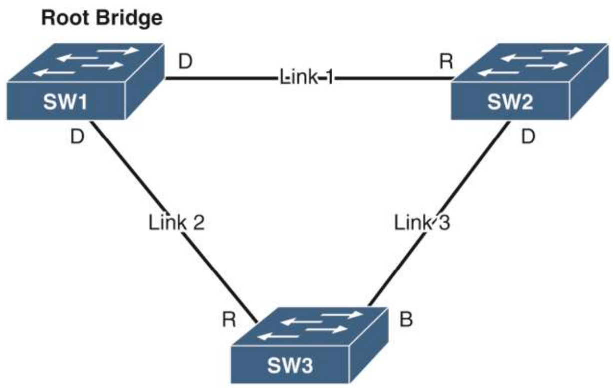
This lab is entirely focused on Spanning Tree Protocol technologies and assumes basic knowledge of the following:

- Trunking configuration
- VLAN configuration and usage
- Layer 2 link aggregation
- IP address configuration
- Basic IP connectivity testing

***NOTE*: The solutions provided in this lab are not all inclusive. There may be many ways to solve each task. All alternate solutions are acceptable, provided that they do not violate previous restraints or tasks.**

802.1D Per-VLAN Spanning Tree Protocol

Initial Lab Setup



Referencing the 802.1D PVST+, 802.1w and the MST topology diagrams, the following is the initial setup for the demonstration labs. Use this same initial set up when starting each lab.

Task 1

Change the hostname on each switch to SW#, where # is the number of the switch (for example, Switch 1 = SW1).

SW1 through SW6 are each assigned hostnames with the hostname command as shown below:

On SW1:

```
Switch(config)#hostname SW1
```

On SW2:

```
Switch(config)#hostname SW2
```

On SW3:

```
Switch(config)#hostname SW3
```

On SW4:

```
Switch(config)#hostname SW4
```

On SW5:

```
Switch(config)#hostname SW5
```

On SW6:

```
Switch(config)#hostname SW6
```

Task 2

Ensure that only the following ports on the switches are in an up/up state:

For each sub task in task 2, the **interface range** command is first used to list all the interfaces on the switches. The **shutdown** command is then issued to shutdown these interfaces. Following this, the **interface range** lists interfaces specified in the task that are brought back up with the **no shut** command:

- SW1
  - E2/1-2
  - E3/1-2

On SW1:

```
SW1(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW1(config-if-range)#shut

SW1(config)#interface range e2/1-2,e3/1-2
SW1(config-if-range)#no shut
```

To verify the configuration:

```
SW1#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status                Protocol
Ethernet2/1            unassigned      YES unset  up                    up
Ethernet2/2            unassigned      YES unset  up                    up
Ethernet3/1            unassigned      YES unset  up                    up
Ethernet3/2            unassigned      YES unset  up                    up
```

- SW2
  - E1/1-2
  - E3/1-2

- E4/1-2
- E5/1-2

On SW2:

```
SW2(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-3,e7/0-3
SW2(config-if-range)#shut

SW2(config)#interface range e1/1-2,e3/1-2,e4/1-2,e5/1-2
SW2(config-if-range)#no shut
```

To verify the configuration:

```
SW2#show ip interface brief | ex down


Interface              IP-Address      OK? Method Status              Protocol
Ethernet1/1            unassigned      YES unset  up                  up
Ethernet1/2            unassigned      YES unset  up                  up
Ethernet3/1            unassigned      YES unset  up                  up
Ethernet3/2            unassigned      YES unset  up                  up
Ethernet4/1            unassigned      YES unset  up                  up
Ethernet4/2            unassigned      YES unset  up                  up
Ethernet5/1            unassigned      YES unset  up                  up
Ethernet5/2            unassigned      YES unset  up                  up
```

- SW3
  - E1/1-2
  - E2/1-2
  - E4/1-2
  - E5/1-2

On SW3:

```
SW3(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-3,e7/0-3
SW3(config-if-range)#shut

SW3(config)#interface range e1/1-2,e2/1-2,e4/1-2,e5/1-2
```

```
SW3(config-if-range)#no shut
```

To verify the configuration:

```
SW3#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status           Protocol
Ethernet1/1            unassigned      YES unset  up               up
Ethernet1/2            unassigned      YES unset  up               up
Ethernet2/1            unassigned      YES unset  up               up
Ethernet2/2            unassigned      YES unset  up               up
Ethernet4/1            unassigned      YES unset  up               up
Ethernet4/2            unassigned      YES unset  up               up
Ethernet5/1            unassigned      YES unset  up               up
Ethernet5/2            unassigned      YES unset  up               up
```

- SW4
  - E2/1-2
  - E3/1-2
  - E6/1-2

On SW4:

```
SW4(config)#interface rang e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW4(config-if-range)#shut

SW4(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW4(config-if-range)#no shut
```

To verify the configuration:

```
SW4#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status           Protocol
Ethernet2/1            unassigned      YES unset  up               up
Ethernet2/2            unassigned      YES unset  up               up
Ethernet3/1            unassigned      YES unset  up               up
Ethernet3/2            unassigned      YES unset  up               up
Ethernet6/1            unassigned      YES unset  up               up
```

```
Ethernet6/2            unassigned      YES unset  up             up
```

- SW5
  - E2/1-2
  - E3/1-2
  - E6/1-2

On SW5:

```
SW5(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW5(config-if-range)#shut

SW5(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW5(config-if-range)#no shut
```

To verify the configuration:

```
SW5#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status          Protocol
Ethernet2/1            unassigned      YES unset  up              up
Ethernet2/2            unassigned      YES unset  up              up
Ethernet3/1            unassigned      YES unset  up              up
Ethernet3/2            unassigned      YES unset  up              up
Ethernet6/1            unassigned      YES unset  up              up
Ethernet6/2            unassigned      YES unset  up              up
```

- SW6
  - E4/1-2
  - E5/1-2

On SW6:

```
SW6(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
```

```
SW6(config-if-range)#shut

SW6(config)#interface range e4/1-2,e5/1-2
SW6(config-if-range)#no shut
```

To verify the configuration:

```
SW6#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status          Protocol
Ethernet4/1            unassigned      YES unset  up              up
Ethernet4/2            unassigned      YES unset  up              up
Ethernet5/1            unassigned      YES unset  up              up
Ethernet5/2            unassigned      YES unset  up              up
```

Task 3

Configure VLANs 10, 20, 30, and 40 on each switch, using any method.

This task requires configuring VLANs 10, 20, 30, 40 on the switches. Since the task does not state any restrictions, for ease of configuration, VTP can also be used to propagate the VLANs to all the switches.

On All Switches:

```
SW1(config)#vlan 10,20,30,40
SW1(config-vlan)#exit
```

To verify the configuration:

On Any Switch:

```
SWx#show vlan brief | include VLAN

VLAN Name                             Status    Ports
10   VLAN0010                         active
20   VLAN0020                         active
30   VLAN0030                         active
40   VLAN0040                         active
```

Task 4

Configure trunk ports on all up/up interfaces.

The commands **switchport trunk encapsulation dot1q** and **switchport mode trunk** is configured on all the interfaces on all the switches that were brought back up in the initial lab setup section of this lab. The combination of these commands manually configures the interfaces to function as 802.1q trunk links.

On SW1:

```
SW1(config)#interface range e2/1-2,e3/1-2
SW1(config-if-range)#switchport trunk encapsulation dot1q
SW1(config-if-range)#switchport mode trunk
```

On SW2:

```
SW2(config)#interface range e1/1-2,e3/1-2,e4/1-2,e5/1-2
SW2(config-if-range)#switchport trunk encapsulation dot1q
SW2(config-if-range)#switchport mode trunk
```

On SW3:

```
SW3(config)#interface range e1/1-2,e2/1-2,e4/1-2,e5/1-2
SW3(config-if-range)#switchport trunk encapsulation dot1q
SW3(config-if-range)#switchport mode trunk
```

On SW4:

```
SW4(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW4(config-if-range)#switchport trunk encapsulation dot1q
SW4(config-if-range)#switchport mode trunk
```

On SW5:

```
SW5(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW5(config-if-range)#switchport trunk encapsulation dot1q
SW5(config-if-range)#switchport mode trunk
```

On SW6:

```
SW6(config)#interface range e4/1-2,e5/1-2
SW6(config-if-range)#switchport trunk encapsulation dot1q
SW6(config-if-range)#switchport mode trunk
```

Configuration Tasks: 802.1D

Before getting into the tasks, it helps to review the basic operation of Spanning Tree Protocol and some of the terms that will be used throughout the solution guide. These concepts will be fleshed out more throughout the guide.

802.1D Spanning Tree Protocol is the protocol that is run between switches used to prevent Layer 2 bridging loops. Loops can form in Layer 2 networks because the Layer 2 Ethernet frame format does not contain any maximum hop count limitation, such as the TTL field in the IP header. This omission of maximum hop count means a single Ethernet frame can be forwarded infinitely throughout a switched network. This is particularly dangerous when the frame being forwarded is a broadcast frame.

Switches are called *transparent bridges* because they abstract the physical network design from the end stations connecting to the LAN. In the past, all stations connected to a Layer 2 LAN were physically connected to the same physical electrical bus via a repeater. Only one station could speak on the segment at a time, and all stations were considered directly connected. This setup was called a *single collision domain* because there was a possibility that two stations would transmit at the same time, and those transmissions could collide.

Later, bridges came about that allowed intelligent learning. An ethernet Bridge would be used to combine two Layer 2 network segments into a single Layer 2 network. Bridges learned MAC addresses and only forwarded traffic between the two network segments when necessary. This operation was transparent to the end stations, because as far as the stations could detect, they were directly connected to the remote stations they were communicating with. The ethernet bridge in this way creates two separate collision domains.

An ethernet switch is basically a multiport bridge. Instead of learning MAC addresses only on two ports, a switch learns MAC addresses on all ports and forwards traffic based on destination MAC addresses to destination ports. If two ethernet frames are destined to the same port, the switch will buffer one frame while forwarding the other. Because not all traffic forwarded across the LAN segment is repeated to all ports, the switch achieves a single collision domain between each port on the switch and the stations connected to those ports.

For the switch to do this, it must first learn which MAC addresses are reachable off its various ports. To do so, the switch reads the source MAC address of all frames it receives and records it associated to the receiving port in a MAC address table, also called a Content **Addressable Memory** or **CAM** table. The switch switches traffic between ports by performing a lookup in the CAM table based on the destination MAC address of the ethernet frames. Through this, known unicast traffic can be forwarded only out the port where the intended station exists.

The problem arises when a switch receives a frame destined to a MAC address it has not learned. The switch cannot drop the frame because that would break the LAN communication. Instead, it floods the frame out all ports in the same VLAN on the switch (including trunk ports that carry those VLANs) except the port on which it initially received the frame. This process, known as **unknown unicast flooding**, can cause a loop condition in the switched network with broadcast frames.

Broadcast frames are frames that are intended to be received by all stations. They are sent to the well-known broadcast destination MAC address FFFF.FFFF.FFFF. When a switch encounters a broadcast frame, it performs unknown-unicast-type flooding for the frame in question. It sends the traffic out all interfaces except the interface on which it was received. The switch does this because the broadcast MAC address FFFF.FFFF.FFFF will never be the source of an ethernet frame. Because it is never the source of an ethernet frame, the switch will never associate the broadcast MAC address with a receiving port in its CAM table, triggering the unknown unicast flooding behavior.

If there are redundant links in a multi-switch environment, where the chain of interconnected links leads back to the switch that originally forwarded the broadcast traffic, a **broadcast storm** occurs. In a broadcast storm, each receiving switch performs the same unknown-unicast-type flooding on the broadcast packet. The broadcast packet is therefore regenerated and looped endlessly throughout the switched network. This leads to high CPU utilization on the switches and can quickly bring down the entire Layer 2 network.

Spanning Tree Protocol converts a switched network into a shortest-path tree. This tree is constructed by designating a switch as the root of the tree, called the **root bridge**. The root bridge is the only bridge in which all of the ports are considered designated ports. A designated port is a port that is responsible for relaying spanning-tree-related messages downstream from the root bridge to other leaf switches. From the root bridge's perspective, all other switches are downstream from it, and thus it should be designated on all of its ports.

All non-root switches elect a single port to become their root port. The root port is the port that the switch uses to reach the root bridge. This port also receives spanning-treerelated messages on the shortest-path tree to be relayed out all other designated ports on the switch.

Finally, redundant links in the network are put into a blocking state. So-called blocking ports are ports that receive BPDUs from a designated port that is not on the shortest-path tree. In other words, they are alternate looped paths to the root bridge that are longer than the path used by the root port. Blocking ports do not send BPDUs in traditional

Spanning Tree Protocol; instead, a blocking port receives a constant flow of BPDUs from the neighboring designated port.

Through this system of designated, root, and blocking ports, STP creates a loop-free network. Broadcast frames and frames undergoing unknown unicast flooding travel the shortest path from leaf switch to root switch and are blocked (dropped) on redundant links.

Task 1

Configure all switches to run 802.1D Spanning Tree Protocol.

Cisco switches run a single instance of spanning tree for every VLAN configured on the switch. This is called **Per-VLAN Spanning-Tree protocol (PVST)**. There are two versions of this: **PVST** and **PVST+**. The original PVST variant was for compatibility with ISL trunks. PVST+ is compatible with 802.1Q trunks. The advantage of running PVST+ on a switch is that a separate Spanning-tree can be calculated per VLAN. This allows the administrator to load balance traffic over redundant links by VLAN. Some VLANs can take a different set of paths while others take the unused paths. The switch calculates the blocking/forwarding state for each individual VLAN running over the trunk link, allowing a single trunk port to be blocking for one VLAN, root for another, and designated for yet another.

802.1D PVST+ is enabled on older Cisco switches by default and can be statically configured using the **spanning-tree mode pvst** command in global configuration mode. The output of the show spanning-tree command indicates which spanning-tree variant the switch is currently using.

On All Switches:

To change the mode to PVST:

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

To verify the configuration:

On SW1:

```
SW1#show spanning-tree | include VLAN|protocol

VLAN0001
  Spanning tree enabled protocol ieee
```

```
VLAN0010
  Spanning tree enabled protocol ieee
VLAN0020
  Spanning tree enabled protocol ieee
VLAN0030
  Spanning tree enabled protocol ieee
VLAN0040
  Spanning tree enabled protocol ieee
```

In the above output, the spanning tree protocol enabled is listed as **ieee.** This is a reference to 802.1D PVST+ which is run as default on some cisco switching platforms.

Task 2

Ensure that SW1 is the root for every VLAN.

Switches running Spanning-tree protocol organize themselves into a shortest-path tree that spans the entire switched network. The tree starts, or has its roots, at a central point in the network. This central point is called the **root bridge** because it is the root of the resulting tree topology. To determine the root bridge, the switches perform a root bridge election. The winner of the election is based on whichever switch has the best **Bridge ID**. The Bridge ID of a switch is composed of the following parts:

1. 16-bit Priority value (includes 12-bits for VLAN using System ID extension)
2. 48-bit MAC address

The Bridge ID is exchanged in STP configuration frames called, **Bridge Protocol Data Units** or **BPDUs,** between switches. A switch receiving a BPDU compares it to its own BPDU. The BPDU with the lowest Bridge ID is considered to have originated from the Root Bridge.This comparison is first made by comparing the configured **Priority Value** (defaults to 32,768 on cisco platforms). If the priority value is a tie, then whichever switch has the lowest MAC address will be elected root. When a switch receives a BPDU from the root, it replaces its own stored BPDU with the Root's and will relay it the Root's BPDU out all non-blocking ports towards other switches in the network. Within the STP network, all switches should agree on which switch is the Root Bridge.

The administrator should be careful in choosing which switch becomes the Root Bridge because once the spanning-tree synchronizes across the switched network, all traffic for the network could pass through the Root. For this reason, the Root bridge should be the most capable or centrally-located switch in the network. Due to the nature of the default selection algorithm, in a production network, if all switches are left at default settings the oldest switch in the network (by production date) will become the root bridge. This is due to how vendors create their products. All vendors have a pool of MAC addresses from which they will assign to their products. This assignment is usually done in ascending order (counting up from 0). Thus, newer created products will have higher MAC addresses than older products in general.

The administrator can select which switch becomes the root bridge by modifying the Priority Value that makes up the Bridge ID. The chosen Root Bridge should have a priority value lower than all other switches in the network. The priority value is configured using the **spanning-tree vlan [vlan-list] priority [priority]** command.

The priority value specified needs to be in multiples of 4096. The value 4096 is set because of the **system extended ID** which uses the last 12 bits of the priority value to encode the VLAN number. Encoding the VLAN number allows each VLAN instance of STP to have a unique Bridge ID for Cisco switches running PVST+, which is a requirement of the STP standard.

Alternatively, the **spanning-tree vlan [vlan-list] root primary** command can be used to automatically set the switch's priority to be lower than the current root by 4096. Keep in mind this command only processes once as a macro. If another switch is configured with a lower priority value, this command does not reset the intended root switch's value lower.

Looking at the task, the administrator is required to set SW1 as the root bridge. For this lab, no extra configuration is needed, since SW1 has already been designated as the root bridge for all VLANs. The decision was made based on SW1's MAC address (lowest) as all switches have a tied priority value of 32,768. The **show spanning-tree vlan 1** example output from SW1 verifies this. SW1 is currently the root for VLAN 1 and will be the root by default for all the VLANs configured for this topology:
NOTE: If SW1 is not the root bridge in your topology, use the spanning-tree vlan 1-4094 root primary command to make it the root bridge.

On SW1:

```
SW1#show spanning-tree vlan 1
```

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    32769

             Address     aabb.cc00.1f00
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     aabb.cc00.1f00
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- ------
Et2/1               Desg FWD 100       128.10   P2p
Et2/2               Desg FWD 100       128.11   P2p
Et3/1               Desg FWD 100       128.14   P2p
Et3/2               Desg FWD 100       128.15   P2p
```

Task 3

Ensure that all switches wait a total of 10 seconds in the listening and learning states before moving a port to forwarding.

Before moving a port from the blocking to a forwarding state, 802.1D transitions between two intermediary states called the **listening** and **learning** states.

During the listening state, the switch listens and transmits configuration BPDUs, but will not transmit any data traffic and will discard any received data traffic. The duration the switch remains in the listening state is equal to the **forward delay time** which is 15 seconds by default. After a duration of the forward delay time, the switch moves the port into the learning state. This state is similar to the listening state, except the switch learns MAC addresses from the data traffic received on a port while still not forwarding them out other ports. The duration the switch remains in the learning state is also equal to the **forward delay timer**, 15 seconds by default, after which the port is transitioned to the forwarding state. These values can be confirmed in the **show spanning-tree** output as shown in the example output from SW2 below:

On SW2:

Notice the two sets of timers in the output below. The first forward delay is the delay set by the Root Bridge and is propagated to all the switches. The second forward delay is the delay set by the local switch and is used if the local switch becomes the root bridge.

```
SW2#show spanning-tree | include VLAN|Forward

VLAN0001
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
VLAN0010
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
VLAN0020
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
VLAN0030
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
VLAN0040
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
```

From this, it can be concluded that by using the default of 15 seconds in each intermediate state, it will take a port a total of 30 seconds to transition from blocking to forwarding.

Because the forward delay timer is used twice in the calculation, the total time spent in the listening and learning states by default will be 30 seconds. To reduce the total time the port stays in the listening or learning states, the forward delay timer must be decreased. This reduction should be taken with care. The default timer was set with the goal of providing enough time for all bridges in the STP network to converge. The shorter the forward delay timer, the more likely it would be to form temporary bridging loops if not all bridges have agreed about the STP topology.

This task requires ports to spend only 10 seconds in the listening and learning states. To do this the forward delay timer must be modified. All switches in the STP network use the root bridge's advertised forward delay time as the agreed upon duration for their own forward delay timers. The **spanning-tree vlan 1-4094 forward-time** command is used on the root bridge SW1 to change the default timer setting. The exact value used requires some thought.

The Forward Delay timer is used twice in the port transition process: to transition from listening to learning and to transition from learning to forwarding. In order to spend a total of 10s in both listening and learning states total, the forward delay timer should be set to 5 seconds. With this change, the port will spend 5 seconds in the listening state and 5 seconds in learning state before forwarding, totaling 10 seconds as the task requires. The **spanning-tree vlan 1-4094 forward-time 5** command implements this change on the root bridge SW1:

On SW1:

```
SW1(config)#spanning-tree vlan 1-4094 forward-time 5
```

The **show spanning-tree** output below verifies the result from the configuration change. Notice now, the forward delay timer now shows 5 seconds for all VLANs:

To verify the configuration:

```
SW1#show spanning-tree | include VLAN|Forward

VLAN0001
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
VLAN0010
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
VLAN0020
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
VLAN0030
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
VLAN0040
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
             Hello Time   2 sec  Max Age 20 sec  Forward Delay  5 sec
```

All switches in the topology will inherit the new forward delay time set at the root bridge from the relayed BPDUs they receive. As an example, the **show spanning-tree root** command below confirms SW2 accepts the new forward delay time of 5s advertised by the root bridge SW1:

On SW2:

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

```
                                        Root      Hello Max Fwd
Vlan                     Root ID        Cost      Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0001         24577 aabb.cc00.1f00       100     2   20   5  Et1/1
VLAN0010         24586 aabb.cc00.1f00       100     2   20   5  Et1/1
VLAN0020         24596 aabb.cc00.1f00       100     2   20   5  Et1/1
VLAN0030         24606 aabb.cc00.1f00       100     2   20   5  Et1/1
VLAN0040         24616 aabb.cc00.1f00       100     2   20   5  Et1/1
```

Task 4

Ensure that if the current root bridge fails, SW2 becomes the new root bridge for all VLANs.

As established previously, spanning-tree elects a root bridge by electing the switch that transmits a BPDU with the lowest Bridge ID to be the root bridge. When the root bridge fails, the switch with the second lowest Bridge ID will become the root bridge. Administrators can influence this decision just as with the original root bridge selection. This can be done by either setting the priority to be higher than the current root but lower than all other switches or using another macro command similar to the **spanning-tree vlan [vlan list] root primary** command.

The **spanning-tree vlan [vlan list] root secondary** command is a one-time macro that sets the bridge priority on the switch to 28,762. With all other switches set to default, the indicated priority value would be lower than all other switches.

To complete the task, execute the **spanning-tree vlan 1-4094 root secondary** command on SW2 to specify that SW2 acts as the root switch should the current root SW1 fail. However, a key point to note here is that this command causes the switch that is configured to be the secondary root using this command to always set its priority to 28,672 as shown below. This could result in SW2 becoming the root since its priority is now lower than that of SW1's if SW1's priority was not set to a much lower value in the second task.

On SW2:

```
SW2(config)#spanning-tree vlan 1-4094 root secondary
```

To verify the configuration:

```
SW2#show spanning-tree

VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    28673
             Address     aabb.cc00.2000
             This bridge is the root  ! SW2 becomes the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    28673  (priority 28672 sys-id-ext 1)
             Address     aabb.cc00.2000
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  15  sec

SW2#show spanning-tree | include VLAN|priority

VLAN0001
  Bridge ID  Priority    28673  (priority 28672 sys-id-ext 1)
VLAN0010
  Bridge ID  Priority    28682  (priority 28672 sys-id-ext 10)
VLAN0020
  Bridge ID  Priority    28692  (priority 28672 sys-id-ext 20)
VLAN0030
  Bridge ID  Priority    28702  (priority 28672 sys-id-ext 30)
VLAN0040
  Bridge ID  Priority    28712  (priority 28672 sys-id-ext 40)
```

This can be remedied by using the **primary** option with the **spanning-tree vlan 1-4094 root primary** command on SW1 after trying to make SW2 the secondary root. As a result of this, the priority on SW1 is now set to 24576 allowing it to regain its root status:

On SW1:

```
SW1(config)#spanning-tree vlan 1-4094 root primary ! Configures SW1 to
regain its root bridge status by setting a lower priority for itself
```

To verify the configuration:

```
SW1#show spanning-tree | include VLAN|priority
```

```
VLAN0001
  Bridge ID  Priority    24577  (priority 24576 sys-id-ext 1)
VLAN0010
  Bridge ID  Priority    24586  (priority 24576 sys-id-ext 10)
VLAN0020
  Bridge ID  Priority    24596  (priority 24576 sys-id-ext 20)
VLAN0030
  Bridge ID  Priority    24606  (priority 24576 sys-id-ext 30)
VLAN0040
  Bridge ID  Priority    24616  (priority 24576 sys-id-ext 40)
```

Task 5

Ensure that SW2's and SW3's E1/2 interface is used as the root port for all VLANs. Do not modify cost to achieve this.

The **show spanning-tree root** output on SW2 and SW3 below shows the E1/1 interface on these switches as the root ports:

On SW2:

```
SW2#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------

VLAN0001         24577 aabb.cc00.1f00       100    2   20  5   Et1/1
VLAN0010         24586 aabb.cc00.1f00       100    2   20  5   Et1/1
VLAN0020         24596 aabb.cc00.1f00       100    2   20  5   Et1/1
VLAN0030         24606 aabb.cc00.1f00       100    2   20  5   Et1/1
VLAN0040         24616 aabb.cc00.1f00       100    2   20  5   Et1/1
```

On SW3:

```
SW3#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0001         24577 aabb.cc00.1f00       100    2   20  5   Et1/1
```

```
VLAN0010         24586 aabb.cc00.1f00       100     2   20  5  Et1/1
VLAN0020         24596 aabb.cc00.1f00       100     2   20  5  Et1/1
VLAN0030         24606 aabb.cc00.1f00       100     2   20  5  Et1/1
VLAN0040         24616 aabb.cc00.1f00       100     2   20  5  Et1/1
```

A root port on a switch is the one that receives the superior BPDU in comparison to the other ports on the same switch. Root ports always point up the spanning tree towards the root bridge. A switch uses the following algorithm to determine which port should become the root port:

1. Port with the least cost to reach the root (RPC)
2. Port that receives the BPDU with the lowest **sender bridge ID** (SBID)
3. Port that receives the BPDU with the lowest **sender port ID** (SPID)
4. Port with the lowest **receiver port ID** (RPID)

Note:
In the above, the **RPC**, **SBID** and the **SPID** values are included in a received configuration BPDU whereas the RPID is not included in a configuration BPDU. This means, when a switch receives a configuration BPDU, in order to determine a root port, the first three steps involve evaluating the values in a received configuration BPDU. The RPID, which is the receiver's port ID is evaluated locally to determine a root port should all other criteria tie.

All ports on switches have default port costs associated with them. These costs vary depending on the port speed. For example, the default port cost for a 10 Mbps link for 802.1.D is 100. The total path cost to the root on non-root switches is calculated by adding the port cost of an interface to the root path cost (RPC) received in the configuration BPDUs.

In the lab topology, SW1 is the root bridge. Since its own cost to reach the root, which is itself, is 0, it sends out configuration BPDUs with a root path cost or the RPC of 0. SW2 and SW3 receive this BPDU on their E1/1-2 interfaces. They accept SW1's BPDU as the superior BPDU making it the root bridge. They send SW1's BPDU out all of their other non-blocking ports. Before sending the BPDU out the other ports, SW2 and SW3 add their own cost to reach the root bridge to the total RPC reported in the received BPDU. In this case, SW2 and SW3's own cost of 100 to reach the root bridge SW1, is added to the cost of 0 advertised by SW1 in its configuration BPDU to these switches. This results in a total cost of 100 (0 + 100). Using SW2 as an example, this BPDU is sent out its E3/1-2, E4/1-2, and E5/1-2 interfaces towards SW3, SW4, and SW5 respectively.

Because SW2 sends the BPDU out its interface towards SW3, SW3 receives two copies of the same BPDU with different RPC values. (SW3 technically can receive a copy of the BPDU from SW4 and SW5, but these BPDUs can be ignored in this explanation because they will be greatly inferior to the one received from SW2 and SW1). SW3 adds its interface cost to the RPC received in the BPDU from SW2, 100 + 100 = 200, and compares it to the BPDU received on all other ports, specifically the one sent from the root SW1 with a cost of 100 (0 RPC advertised from the root plus the interface cost of 100) over its E1/1-2 interfaces. Obviously, SW3 decides that the RPC through its interfaces towards SW1 is the best cost and it is this cost that SW3 advertises in BPDUs towards other switches. At this point SW3 must choose which of those two ports (E1/1 - E1/2) to mark as the root port.

The same process occurs on SW2 with respect to the BPDU received from SW3. Similar to SW3, SW2 must choose between its two interfaces towards SW1 to be its root port.

The **show spanning-tree detail** command can be used to verify the cost to the root over different interfaces. As an example, the **show spanning-tree vlan 1 interface e1/1 detail** and **show spanning-tree vlan 1 interface e3/1 detail** is issued on SW2 below. The port path cost is the cost assigned to the port on SW2. The designated path cost is the RPC received by the switch from its upstream designated port in configuration BPDUs. The value here can be an indicator of whether or not this switch is directly connected to the root bridge. If the root bridge is the directly connected designated port, the value will be 0. Any other value indicates this switch is connected to a non-root switch.

On SW2:

```
SW2#show spanning-tree vlan 1 interface e1/1 detail

Port 6 (Ethernet1/1) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.6.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.10, designated path cost 0
   Timers: message age 2, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 159, received 4679

SW2#show spanning-tree vlan 1 interface e3/1 detail

 Port 14 (Ethernet3/1) of VLAN0001 is designated forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.14.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 28673, address aabb.cc00.2000
   Designated port id is 128.14, designated path cost 100
   Timers: message age 0, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 4888, received 5
```

In the above it was determined that SW2 and SW3 will choose one of their ports towards SW1 as root. The Spanning-tree specification only allows a single root port to exist on a switch for any given instance of spanning-tree. Thus, SW2 and SW3 need to determine which of the two ports connected to SW1 is receiving the better BPDU.

As seen with the **show spanning-tree detail** output from SW2 and SW3 below, the cost to reach the root over their E1/1 and E1/2 interfaces are both equal to 100 (Cost to the root = Port path cost + Designate path Cost):

On SW2:

```
SW2#show spanning-tree vlan 1 interface e1/1 detail

 Port 6 (Ethernet1/1) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.6.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.10, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 159, received 4828

SW2#show spanning-tree vlan 1 interface e1/2 detail

 Port 7 (Ethernet1/2) of VLAN0001 is alternate blocking

   Port path cost 100, Port priority 128, Port Identifier 128.7.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.11, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 148, received 4853
```

On SW3:

```
SW3#show spanning-tree vlan 1 interface e1/1 detail

Port 6 (Ethernet1/1) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.6.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.14, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 2
   Link type is point-to-point by default
   BPDU: sent 12, received 5046

SW3#show spanning-tree vlan 1 interface e1/2 detail

Port 7 (Ethernet1/2) of VLAN0001 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.7.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.15, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 0
   Link type is point-to-point by default
   BPDU: sent 2, received 5062
```

With the cost values being equal over the E1/1 and E1/2 interfaces, SW2 and SW3 need to move on to the next tiebreaker to determine the root port between these interfaces. The next tie breaker calls to evaluate the senders bridge ID, SBID in the received configuration BPDUs. In this case, the SBID in the BPDUs received on the E1/1 and E1/2 interfaces are identical, meaning SW2 and SW3 will now have to use the third tie breaker to determine the root port.

The third tie breaker compares the **sender port ID**, **SPID**, in the received configuration BPDUs. All ports on a switch are assigned a port ID. The port ID is made up of two parts: **Port priority** which is a configurable value, and **port index** which is the same as the SNMP ifindex value. When all else ties, the receiving switch compares the SPIDs of the remote designated port that is included in the received configuration BPDUs. The port that receives the lowest SPID is designated as the root port.

The **show spanning-tree detail** output can be used to verify the SPID. The output of this command refers to SPID as the **Designated port ID**. The output below shows these values to be lower on the E1/1 interface on

SW2 and SW3 when compared to the E1/2 interfaces. With lower values being preferred with STP, the E1/1 interface on SW2 and SW3 finally get elected as the root port:

On SW2:

```
SW2#show spanning-tree vlan 1 interface e1/1 detail

Port 6 (Ethernet1/1) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.6.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.10, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 159, received 5177

SW2#show spanning-tree vlan 1 interface e1/2 detail

Port 7 (Ethernet1/2) of VLAN0001 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.7.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.11, designated path cost 0
   Timers: message age 2, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 148, received 5187
```

As we can see above, the Port ID over the E1/1 interface is 128.10 which is lower than the port ID of 128.11 over the E1/2 interface.

On SW3:

```
SW3#show spanning-tree vlan 1 interface e1/1 detail

 Port 6 (Ethernet1/1) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.6.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.14, designated path cost 0
   Timers: message age 2, forward delay 0, hold 0
   Number of transitions to forwarding state: 2
   Link type is point-to-point by default
   BPDU: sent 12, received 5437

SW3#show spanning-tree vlan 1 interface e1/2 detail

Port 7 (Ethernet1/2) of VLAN0001 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.7.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.15, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 0
   Link type is point-to-point by default
   BPDU: sent 2, received 5460
```

Port ID over the E1/1 interface is 128.14 which is lower than the port ID of 128.15 over the E1/2 interface.

The task requires SW2 and SW3 to use their E1/2 ports as root ports. One of the ways to achieve this is to modify the cost values so as to make the path cost via E1/2 lower than that over E1/1. The task however restricts modifying the cost. The next option would be to modify the configurable part of the SPID on the root bridge SW1 so that the SPID in the BPDU received over the E1/2 interfaces on SW2 and SW3 is lower. This is done with the spanning-tree vlan 1-4094 port-priority 64 command on the E2/2 and E3/2 interfaces on SW1. As a result, SW2 and SW3 now designate their E1/2 interfaces as the root port and the E1/1 interfaces as alternate blocking:

Note: The spanning-tree port-priority value can be changed in increments of 16

On SW1:

```
SW1(config)#interface e2/2
SW1(config-if)#spanning-tree vlan 1-4094 port-priority 64

SW1(config)#interface e3/2
SW1(config-if)#spanning-tree vlan 1-4094 port-priority 64
```

On SW1:

```
SW1(config)#interface e2/2
SW1(config-if)#spanning-tree vlan 1-4094 port-priority 64

SW1(config)#interface e3/2
SW1(config-if)#spanning-tree vlan 1-4094 port-priority 64
```

On SW2:

After the above configurations on SW1, the port ID over the E1/2 interface is 64.11 which is lower than the port ID of 128.10 over the E1/1 interface on SW2

```
SW2#show spanning-tree vlan 1 interface e1/1 detail

Port 6 (Ethernet1/1) of VLAN0001 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.6.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.10, designated path cost 0
   Timers: message age 2, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 159, received 5484

SW2#show spanning-tree vlan 1 interface e1/2 detail

Port 7 (Ethernet1/2) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.7.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 64.11, designated path cost 0
   Timers: message age 2, forward delay 0, hold 0
   Number of transitions to forwarding state: 2
   Link type is point-to-point by default
   BPDU: sent 150, received 5557
```

On SW3:

After the above configurations on SW1, the port ID over the E1/2 interface is 64.15 which is lower than the port ID of 128.14 over the E1/1 interface

```
SW3#show spanning-tree vlan 1 interface e1/1 detail

Port 6 (Ethernet1/1) of VLAN0001 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.6.

   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 128.14, designated path cost 0
   Timers: message age 1, forward delay 0, hold 0
   Number of transitions to forwarding state: 2
   Link type is point-to-point by default
   BPDU: sent 12, received 5755
```

```
SW3#show spanning-tree vlan 1 interface e1/2 detail

Port 7 (Ethernet1/2) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.7.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 24577, address aabb.cc00.1f00
   Designated port id is 64.15, designated path cost 0
   Timers: message age 2, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 4, received 5779
```

Task 6

Ensure that SW5 uses its E3/1 port as the root port for VLANs 10 and 30. Do not modify SW2 to achieve this.

As the topology stands now, SW5 designates its E2/1 interface as the root port for VLAN 10 and 30 towards the root switch SW1. The E3/1 interface is in a blocking state:

On SW5:

```
SW5#show spanning-tree vlan 10 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et2/1               Root FWD 100       128.10   P2p
Et2/2               Altn BLK 100       128.11   P2p
Et3/1               Altn BLK 100       128.14   P2p
Et3/2               Altn BLK 100       128.15   P2p
Et6/1               Desg FWD 100       128.26   P2p
Et6/2               Desg FWD 100       128.27   P2p

SW5#show spanning-tree vlan 30 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et2/1               Root FWD 100       128.10   P2p
Et2/2               Altn BLK 100       128.11   P2p
Et3/1               Altn BLK 100       128.14   P2p
```

```
Et3/2               Altn BLK 100       128.15   P2p
Et6/1               Desg FWD 100       128.26   P2p
Et6/2               Desg FWD 100       128.27   P2p
```

Using certain **show** commands helps understand why SW5 chooses E2/1 as its root port. First, using the **show spanning-tree vlan 10,30 interface e2/1** and **show spanning-tree vlan 10,20 interface e3/1** outputs, the cost to reach the root SW1 over these interfaces can be determined. The cumulative root cost is calculated by adding the **port path cost** of the interface and the **designated path cost** over that interface. Recall, the port path cost is the cost assigned to the interface whereas the designated path cost is the RPC received by the switch from its upstream designated port in configuration BPDUs.

NOTE: SW5 may also receive BPDUs relayed from SW6, but this BPDU will be inferior to all other BPDUs and as such is ignored in this explanation.

On SW5:

```
SW5#show spanning-tree vlan 10,30 interface e2/1 detail

Port 10 (Ethernet2/1) of VLAN0010 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.10.
   Designated root has priority 24586, address aabb.cc00.1f00
   Designated bridge has priority 28682, address aabb.cc00.2000
   Designated port id is 128.22, designated path cost 100
   Timers: message age 3, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 5, received 5747

Port 10 (Ethernet2/1) of VLAN0030 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.10.
   Designated root has priority 24606, address aabb.cc00.1f00
   Designated bridge has priority 28702, address aabb.cc00.2000
   Designated port id is 128.22, designated path cost 100
   Timers: message age 3, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   Link type is point-to-point by default
   BPDU: sent 3, received 4988

SW5#show spanning-tree vlan 10,30 interface e3/1 detail

 Port 14 (Ethernet3/1) of VLAN0010 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.14.
   Designated root has priority 24586, address aabb.cc00.1f00
   Designated bridge has priority 32778, address aabb.cc00.2100
   Designated port id is 128.22, designated path cost 100
   Timers: message age 5, forward delay 0, hold 0
   Number of transitions to forwarding state: 0
   Link type is point-to-point by default
   BPDU: sent 2, received 6043

Port 14 (Ethernet3/1) of VLAN0030 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.14.
   Designated root has priority 24606, address aabb.cc00.1f00
   Designated bridge has priority 32798, address aabb.cc00.2100
   Designated port id is 128.22, designated path cost 100
   Timers: message age 9, forward delay 0, hold 0
   Number of transitions to forwarding state: 0
   Link type is point-to-point by default
   BPDU: sent 3, received 4847
```

Using this method of calculation, the RPC for VLANs 10 and 30 is 200 over the e2/1 and e3/1 interfaces on SW5. With these cost values equal, SW5 uses the next tiebreaker to determine the root port, comparing the sending switchs' Bridge ID (SBIDs) in the received configuration BPDUs.

SW5 receives configuration BPDUs from SW2 and SW3. The BPDUs contain the SBID, which consists of the priority and the MAC address, along with the Bridge ID of the current root bridge. Once again, because the path costs tie between the two received BPDUs, SW5 will choose the port receiving the BPDU with the lowest SBID.

In the above output, SW5 receives a BPDU on its E2/1 interface on VLAN 10 and 30 with the SBID priority as 28682 from SW2. On its E3/1 interface, SW5 receives a BPDU with a SBID priority as 32798 from SW3. These values are listed in the "Designated bridge has priority" section in the output. Because SW2's BPDU has a lower priority than SW3's, SW5 places its E3/1 interface in blocking mode for both VLANs in question.

NOTE: The ultimate determination for root port in this case is going to be the SPID. SW5 will choose the BPDU received on its E2/1 interface over its E2/2 interface because SW2's SPID over E2/1 is lower than E2/2. The main focus on this task, however, is pointing out how the Bridge ID affects the decision-making between the BPDU received from SW2 and the BPDU received from SW3.

The task requires that SW5 designates its E3/1 interface as the root port for VLAN 10 and VLAN 30. One way to do this would be to adjust SW3's bridge priority to be lower than SW2. Doing so, however, would violate the 4th task specifying that SW2 should be the secondary root should SW1 fail. In such a scenario, SW3 with the lower bridge priority would be elected root bridge instead of SW2.

The easiest way to accomplish the task is to modify the costs such that the cost to reach the root over the E3/1 interface is lower than that over the E2/1 interface for VLANs 10 and 30. The following configuration shows the port cost is set to 50 (default 100) on the E3/1 interface on SW5 with the **spanning-tree vlan 10,30 cost 50** command. On doing so, SW5 transitions the E3/1 interface to root and E2/1 to blocking as evidenced by the debug spanning-tree events below.

On SW5:

Before changing the configuration:

```
SW5#debug spanning-tree events
Spanning Tree event debugging is on

SW5(config)#interface e3/1
SW5(config-if)#spanning-tree vlan 10,30 cost 50

STP: VLAN0010 new root port Et3/1, cost 150
STP: VLAN0010 Et3/1 -> listening
STP: VLAN0010 sent Topology Change Notice on Et3/1
STP[10]: Generating TC trap for port Ethernet2/1
STP: VLAN0010 Et2/1 -> blocking
STP: VLAN0030 new root port Et3/1, cost 150
STP: VLAN0030 Et3/1 -> listening
STP: VLAN0030 sent Topology Change Notice on Et3/1
STP[30]: Generating TC trap for port Ethernet2/1
STP: VLAN0030 Et2/1 -> blocking

SW5#show spanning-tree vlan 10,30 blockedports

Name                 Blocked Interfaces List
-------------------- ------------------------------------
VLAN0010             Et2/1, Et2/2, Et3/2
VLAN0030             Et2/1, Et2/2, Et3/2

Number of blocked ports (segments) in vlan 10,30: 6

SW5#show spanning-tree vlan 10,30 root
```

```
Root Hello Max Fwd
Vlan                     Root ID             Cost  Time Age Dly Root Port
---------------- -------------------- ---------- ----- --- ---  ------------
VLAN0010         24586 aabb.cc00.1f00       150     2   20  15      Et3/1
VLAN0030         24606 aabb.cc00.1f00       150     2   20  15      Et3/1
! E3/1 now designated as the root port


SW5#u all
All possible debugging has been turned off
```

Task 7

Ensure that SW4 uses its E3/1 port as root port for VLANs 20 and 40. Do not modify SW3 to achieve this.

The **show spanning-tree vlan 20,40** output below shows the current root port for VLAN 20 and 40 on SW4 is its E2/1 interface:

On SW4:

```
SW4#show spanning-tree vlan 20 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et2/1               Root FWD 100       128.10   P2p
Et2/2               Altn BLK 100       128.11   P2p
Et3/1               Altn BLK 100       128.14   P2p
Et3/2               Altn BLK 100       128.15   P2p
Et6/1               Desg FWD 100       128.26   P2p
Et6/2               Desg FWD 100       128.27   P2p

SW4#show spanning-tree vlan 40 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et2/1               Root FWD 100       128.10   P2p
Et2/2               Altn BLK 100       128.11   P2p
Et3/1               Altn BLK 100       128.14   P2p
Et3/2               Altn BLK 100       128.15   P2p
Et6/1               Desg FWD 100       128.26   P2p
Et6/2               Desg FWD 100       128.27   P2p
```

The following determines SW4's current root port selection. SW4 receives configuration BPDUs relayed from SW2, SW3 and SW6. Among all the BPDUs received, the BPDUs from SW6 will be the most inferior one due to higher cost to reach the root through SW6. The configuration BPDUs SW4 receives from SW2 and SW3 both result in an equal cost of 200 to the root SW1. With the cost tied, SW4 then uses the next tie breaker of comparing the SBID to determine which BPDU is the most superior. Since the SBID of SW2 is lower than the SBID of SW3, SW4 has to now consider which of the direct ports connected to SW2 it can designate as the root port. At this point, SW4 uses the next tie breaker of comparing the SPID and designates its E2/1 interface as the root port due to a lower SPID of SW2's E4/1 interface.

The task requires ensuring that SW4's E3/1 interface is designated as the root port. One of the ways this can be achieved is to lower the BID of SW3, however the task restricts any changes from being made on SW3 and similar to the previous task, doing so would violate the 4th task in the lab. So once again, the easiest way to complete this task is by modifying the cost values such that the cost to reach the root through SW4 over its E3/1 interface is lower than the cost to reach the root over its E2/1 interface. So, to solve this task the interface level command **spanning-tree vlan 20,40 cost 50** is configured on SW4's E3/1 interface:

<u>On SW4:</u>

```
SW4(config)#interface e3/1
SW4(config-if)#spanning-tree vlan 20,40 cost 50
```

With the **debug spanning-tree events** turned on below, SW4 transitions the previously blocked E3/1 interface to a forwarding state for vlans 20 and 40. Notice the time taken from SW4 to transition the port into a forwarding state from the blocking state is 10 seconds. This is a result of setting the forward delay time to 5 seconds in task 3 of this section. The switch spends 5 seconds in the listening state, 5 seconds in the learning state and then transitions to the forwarding state.

Note : The default settings would cause the switch to spend 15 seconds in each transitory state, resulting in the switch taking 30 seconds to move from the blocking state to the forwarding state.

The **show spanning-tree vlan 20 root** and **show spanning-tree vlan 40 root** output further verifies E3/1 as the new root port for VLAN 20 and 40:

On SW4:

Port transitions from listening to forwarding in 10 seconds for VLAN 20

```
*Apr 13 11:28:14.376: STP: VLAN0020 new root port Et3/1, cost 150
*Apr 13 11:28:14.376: STP: VLAN0020 Et3/1 -> listening
*Apr 13 11:28:19.381: STP: VLAN0020 Et3/1 -> learning
*Apr 13 11:28:24.386: STP: VLAN0020 Et3/1 -> forwarding
```

For VLAN 40

```
*Apr 13 11:28:14.376: STP: VLAN0040 new root port Et3/1, cost 150
*Apr 13 11:28:14.376: STP: VLAN0040 Et3/1 -> listening
*Apr 13 11:28:19.381: STP: VLAN0040 Et3/1 -> learning
*Apr 13 11:28:24.386: STP: VLAN0040 Et3/1 -> forwarding
```

```
SW4#show spanning-tree vlan 20 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et2/1               Altn BLK 100       128.10   P2p
Et2/2               Altn BLK 100       128.11   P2p
Et3/1               Root FWD 50        128.14   P2p
Et3/2               Altn BLK 100       128.15   P2p
Et6/1               Desg FWD 100       128.26   P2p
Et6/2               Desg FWD 100       128.27   P2p

SW4#show spanning-tree vlan 40 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et2/1               Altn BLK 100       128.10   P2p
Et2/2               Altn BLK 100       128.11   P2p
Et3/1               Root FWD 50        128.14   P2p
Et3/2               Altn BLK 100       128.15   P2p
Et6/1               Desg FWD 100       128.26   P2p
Et6/2               Desg FWD 100       128.27   P2p

SW4#show spanning-tree vlan 20 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0020          24596 aabb.cc00.1f00       150    2   20   5  Et3/1
```

```
SW4#show spanning-tree vlan 40 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0040         24616 aabb.cc00.1f00       150    2   20   5  Et3/1
```

This example showed how it can take a port up to 30s to transition from blocking to forwarding with default timer settings. In the lab, this time period is reduced to 10s through forward delay timer modifications. It should still be noted that, even at 10s, 802.1D is not a fast converging protocol, 10 seconds is a significant amount of time before restoring functionality and is the main drawback to timer-based convergence algorithms such as the one employed by traditional 802.1D Spanning-tree protocol. Later on, in the lab, this deficiency is examined again when transitioning to 802.1w Rapid Spanning-tree protocol, which utilizes a more event-based convergence system.

Task 8

Ensure that interfaces connected to non-switch hosts come online immediately.

Spanning-tree protocol, as mentioned early on, was invented to deal with the possibility of loops in the switched network when multiple switches were connected by redundant links. To do so, spanning-tree blocks redundant links based on a predetermined algorithm. This algorithm not only governs how switches determine which redundant links to block but also what to do with links that are allowed to forward.

In 802.1D spanning-tree, ports that are deemed suitable for forwarding (ports that will not cause a loop), transition from **blocking → listening → learning → forwarding** states. As stated in a task above, the port will spend Forward Delay seconds in both the listening and learning states. The reasoning behind this progression is that before a port moves to forwarding, it must be determined that doing so would not cause a loop.

In the blocking state all traffic is discarded and only BPDUs are accepted (but not sent). The port then transitions to the listening state where it is evaluating BPDUs that could possibly be received from another switch. Next, the switch enters the learning phase where it learns MAC addresses from received data traffic (still not forwarding the traffic) on the port. Finally, the port enters the forwarding state and normal operation can begin.

These mechanisms work well for links connected between switches mainly because other switches are the devices that can form the loops. The mechanisms are in place to allow time for BPDUs to progress throughout the network and for all of the switches to synchronize the spanning-tree topology. The mechanisms do not work well for the end hosts that are connected to the switches, nor are they necessarily needed.

A switch will have two kinds of links, one to another switch and one to an end host. For the sake of this explanation, an end host can be a router, PC, firewall, wireless access point, server, or any other device that does not extend the Layer 2 broadcast domain between multiple switches. End hosts are the final destinations for all inter-switch data traffic. Because end hosts do not extend the broadcast domain and they are the destination for all inter-switch data traffic (they do not transit Layer 2 traffic), there is a low probability that such ports will cause a loop in the forwarding state.

This low probability means the switch does not need to transition such ports from blocking to listening because end hosts typically do not send BPDUs. It does not need to spend time in the learning state either because the port will only learn the MAC address of the directly-connected end host. Logically, it is feasible to transition such ports directly to forwarding instead of progressing through the **blocking → listening → learning → forwarding** states.

In actuality, if the host has to wait the default 30 seconds before being allowed to forward traffic on the LAN, it can delay or even break certain pre-boot tasks performed by the NICs of most end-hosts. The most notable task being DHCP. DHCP is the protocol used by a host to acquire an IP address it can use on the LAN for communication. If the switch port does not move from blocking to forwarding in enough time for the host to receive a DHCP reply, some hosts will cease trying to obtain an IP address.

So, for such cases, a Cisco-proprietary STP optimization feature called **portfast** can be enabled on these ports and this is what this task hints towards configuring. The portfast feature on SW4, SW5 and SW6 will allow them to bypass the listening and learning states and transition their access ports immediately to a forwarding state. This would result in providing immediate network connectivity to end devices for external services such as DHCP. Without the portfast feature, end devices would have to wait for up to 30 seconds, the time taken for the access port to transition to the forwarding state from the blocking state.

The port fast feature can be enabled directly on an interface with the **spanning-tree portfast edge** command or globally with the **spanning-tree portfast edge default** command. The difference between these commands is that the interface-level command should be executed on all access ports by the administrator. The global way of enabling the port fast feature automatically activates the port fast feature on all access ports. Portfast can also be enabled on trunk links connected to end hosts such as wireless autonomous or hybrid APs, datacenter hypervisors, or any other device that needs to receive traffic from multiple VLANs using the **spanning-tree portfast edge trunk** command on the trunk port. Care should be taken to ensure another switch is never mistakenly connected to the trunk port with portfast as temporary bridging loops can occur.

Configuring host ports as portfast interfaces also carries another benefit. Once portfast is enabled, the interface will not generate topology change notifications whenever the interface goes down. This is important because host ports may shutdown repeatedly during the day as hosts come online and go offline. Without portfast, each event would cause a topology change in the network, forcing all switches to age out their MAC address tables prematurely, flooding unknown unicast traffic until MAC addresses are relearned. Secondly, portfast-enabled interfaces do not age their MAC address entries in the CAM table during a topology change event. The reasoning is since these ports connect to end hosts, those MAC addresses will always be learned on the port. There is no need to flush the table only to relearn the exact same information.

Since the lab topology does not explicitly ask to configure any access port, this task is solved by enabling the portfast feature globally using the **spanning-tree portfast edge default** command on SW4, SW5 and SW6. The **show spanning-tree summary** output can be used to verify the status of the feature as seen below:

On SW4, SW5, and SW6:

```
SW4(config)#spanning-tree portfast edge default

%Warning: this command enables portfast by default on all interfaces.
```

```
You should now disable portfast explicitly on switched ports leading to
hubs, switches and bridges as they may create temporary bridging loops.
```

To verify the configuration:

```
SWx#show spanning-tree summ | include Port

Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is disabled
Portfast Edge BPDU Filter Default       is disabled
```

a. If SW4 or SW5 detects a switch on these ports when they first come online, they should process the BPDUs normally. Otherwise, it should not process received BPDUs.

Ports configured with the portfast feature are supposed to be connected to end hosts. This is because it is safe for interfaces connected to end hosts to immediately come online rather than waiting through the listening and learning states. There are times, however, where portfast-enabled interfaces are connected to another switch either by accident or on purpose. In such an event, the switch port will move directly into forwarding state without transitioning properly to ensure there are no loops in the switched network. This means temporary Layer 2 loops could exist in the network. In this case, it is desirable for the interface to participate in STP if the portfast-enabled interface is connected to another switch.

The global **BPDU Filter** feature was created just for this purpose. The BPDU filter itself effectively disables spanning-tree operation on a switch port. No BPDUs will be sent or processed on a BPDU-filter-enabled port. BPDU filter is enabled on a per-port basis by using the **spanning-tree bpdufilter enable** command in interface configuration mode. Enabled globally using the **spanning-tree portfast edge bpdufilter default** command in global configuration mode, however, BPDU filter changes its operation in the following ways:

1. It is only enabled on portfast-enabled interfaces
2. It first waits a period of time before activating on the interface
    a. If a BPDU is received on the portfast-enabled interface during the waiting period, the BPDU filter is disabled and the port continues to process BPDUs.

b. If a BPDU is not received on the portfast-enabled interface during the waiting period, the port is allowed to transition directly to forwarding state. BPDUs will cease to be sent out of the port unless one is received.

When enabled globally, BPDU filter only operates on portfast-enabled interfaces. Once a portfast-enabled interface first comes online, BPDU filter allows the port to continue sending and receiving BPDUs for 10 hello intervals. The result is the switch will send 11 hello packets in total, one initially and then one for each hello interval BPDU filter waits before activating. If no BPDUs are received, the port is allowed to jump directly to the forwarding state. The switch port also ceases to send any new BPDUs out of the BPDU-filter-enabled port unless a BPDU is first received. If a BPDU is received during the initial 11 BPDU timespan, the BPDU filter and portfast is disabled. The port evaluates the BPDUs and transitions to the listening and learning states before moving to forwarding based on the exchange of BPDUs like normal.

This operation works because a properly functioning spanning-tree-enabled switch will always send out BPDUs when its port first comes online.

To solve subtask A, the BPDU filter feature should be enabled globally using the **spanning-tree portfast edge bpdufilter default** command on SW4 and SW5. Once enabled, BPDU filter operation will take place as described above. BPDU filter default configuration can be verified using the **show spanning-tree summary** command as shown below:

On SW4 and SW5:

```
SWx(config)#spanning-tree portfast edge bpdufilter default

SW4#show spanning-tree summary | include Port

Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is disabled
Portfast Edge BPDU Filter Default       is enabled

SW5#show spanning-tree summary | include Port

Portfast Default                        is edge
Portfast Edge BPDU Guard Default        is disabled
Portfast Edge BPDU Filter Default       is enabled
```

b. If SW6 detects a switch on one of these ports, it should disable the port.

The issue with configuring portfast alone on a switch is that the portfast-enabled interface comes online immediately. These ports are supposed to lead to end hosts and not other switches in the network. The ports will, however, still process received BPDUs. Such behavior can allow rogue switches or malicious users to hijack the spanning-tree topology by crafting spoofed BPDUs claiming their device is the root bridge.

The BPDU filter feature helps mitigate it by telling the port to ignore all BPDUs and not to send out its own BPDUs if a BPDU is received on an interface. This comes at an inherent risk. If the intended portfast interface receives a BPDU, that means there is another spanning-tree-enabled device connected to the port. This could be either another switch, a malicious user, or a host with two NICs connected to separate wall jacks that is either intentionally or unintentionally bridging traffic between the two ports. In either case, there is a problem with the network since that particular port has been designated as a port towards an end host that should not cause loops in the network. If this point is not true, a loop can be formed in the network.

Cisco has another feature designed to take a more aggressive approach to solving this problem. That feature is known as BPDU guard. When BPDU guard is enabled on a port, if the configured port receives any BPDUs, that port is shut down and placed into an err-disabled state. This state does not clear unless the port is shut down and brought back up again.

Taking this approach protects the network from issues and also can serve as an alert of a misconfiguration in the network. BPDU guard is enabled using the interface-level command **spanning-tree bpduguard enable**. Like the BPDU filter, BPDU guard can also be enabled globally on the switch port. Once enabled globally using the **spanning-tree portfast edge bpduguard default** command, BPDU guard only takes effect on portfast-enabled interfaces. If a BPDU is received on any portfast enabled interface, that port is put in **err-disabled state** and is immediately shut down.

It is worth mentioning the interaction between BPDU guard and BPDU filter when both are enabled on an interface. If port-level BPDU guard and BPDU filter are configured on an interface, BPDU filter will take precedence over BPDU guard. The interface will not process received BPDUs, thus BPDU guard will not shut down the interface. The BPDU guard configuration is effectively rendered unnecessary.

If both features are enabled globally, however, the two features mesh very well. BPDU filter will prevent the port from sending BPDUs on the interface, however if a BPDU is received, BPDU guard will shut it down. This configuration is good for interfaces that should lead to end hosts where it is not desirable to send out BPDUs but if one is received the port should be deactivated for investigation.

This task requires that when a portfast enabled port on SW6 receives a BPDU, the port should be disabled instead of functioning as a non edge port. BPDU guard will be enabled globally on SW6 to meet this requirement with the **spanning-tree portfast edge bpduguard default** command shown below:

On SW6:

```
SW6(config)#spanning-tree portfast edge bpduguard default

SW6#show spanning-tree summary | include Port

Portfast Default                          is edge
Portfast Edge BPDU Guard Default          is enabled
Portfast Edge BPDU Filter Default         is disabled
```

NOTE: The **show interface status err-disabled** command can be used to display any interface that has an err-disabled state.

Task 9

Ensure that SW6 is able to fully utilize redundant links between neighboring switches.
Use a Cisco-specific approach to solve this.

This task aims to configure the mentioned switches such that multiple redundant links between them can be utilized. This is a hint towards using **EtherChannel bundling**. In an etherchannel, the switch takes multiple links and bundles them into one logical link. The logical link is called the **port channel interface**. Once bundled, the switch will send traffic outbound across the individually bundled links based on a load-balancing method.

When configuring etherchannels it is important that both devices on either side of the link agree that the links should be bundled. The reason is each device may send frames from the same ethernet source MAC address across different links. If the far-end switch has not properly bundled the link, it may cause MAC address flapping on the far-end switch. Additionally, loops could form because STP BPDUs are only sent out of one link in the bundle to the far-end switch. This means the far-end switch would only receive a BPDU on one of its ports, making it assume the rest of the bundled ports are not connected to a neighboring bridge. This assumption causes the far-end switch to set all of those same ports into designated state. In this case, traffic can be looped from the local switch (configured with an etherchannel bundle) and the far-end switch (not configured with an etherchannel bundle).

To help keep EtherChannel configurations consistent, two main protocols exist, standards-based **LACP** and cisco-proprietary **PAgP**. If the EtherChannel is configured to run one of these protocols, messages are exchanged between the switches that ensure consistent configuration and prevent the scenario above. If misconfiguration is detected, the links will not be bundled and are kept as independent links.

Port channels will be created between SW4 - SW6 and SW5 - SW6 using PAgP since the task specifically asks to use a Cisco specific approach. PAgP can be enabled on the links using the **channel-group {number} mode { desirable | auto }** command. If enabled with the **desirable** option, the switch will send PAgP frames to the far-end switch in an attempt to bring up the etherchannel. If enabled with the **auto** option, the switch will wait to receive PAgP frames from the far-end switch before forming the bundle. After entering the configuration commands, the switch creates a logical Port-Channel interface. The port-channel interface will replicate all future configurations applied to it to the member interfaces.

When configuring etherchannels it is best to follow the following procedures:

Step 1: Use the **default interface** command for all the interfaces that are to be bundled up into the same port channel. The default interface command resets the interface back to their default values. Additionally, for ease of configuration, the **default interface range** command can be used to reset multiple interfaces to their default state.

Step 2: Configure the EtherChannel by issuing the **channel-group** *channel-number* **mode** command to the physical interfaces. This will result in automatic port-channel interface creation. The channel-number value does not have to match on both sides.

Step 3: Configure the trunk related parameters directly on the port-channel interface configuration mode

Step 4: Reset the ports in the group by first issuing the **shutdown** command followed by the **no shutdown** command.

Because the trunking ports have already been configured in earlier in the lab, only step 2 from the above is configured in the output below:

On SW6:

```
SW6(config)#interface range e4/1-2
SW6(config-if-range)#channel-group 4 mode desirable

SW6(config)#interface range e5/1-2
SW6(config-if-range)#channel-group 5 mode desirable
```

On SW4:

```
SW4(config)#interface range e6/1-2
SW4(config-if-range)#channel-group 4 mode desirable
```

On SW5:

```
SW5(config)#interface range e6/1-2
SW5(config-if-range)#channel-group 5 mode desirable
```

The **show EtherChannel summary** output on the switches can be used to confirm the status of the port channel with the **SU** indicating a in use Layer 2 bundle:

On SW6:

```
SW6#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      N - not in use, no aggregation
       f - failed to allocate aggregator

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

        A - formed by Auto LAG

Number of channel-groups in use: 2
Number of aggregators:           2

Group  Port-channel  Protocol    Ports
------+-------------+-----------+--------------------
4      Po4(SU)         PAgP      Et4/1(P)    Et4/2(P)
5      Po5(SU)         PAgP      Et5/1(P)    Et5/2(P)

SW4#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      N - not in use, no aggregation
       f - failed to allocate aggregator

       M - not in use, minimum links not met
       m - not in use, port not aggregated due to minimum links not met
       u - unsuitable for bundling
       w - waiting to be aggregated
       d - default port
       A - formed by Auto LAG

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

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

4      Po4(SU)         PAgP      Et6/1(P)    Et6/2(P)
```

<u>On SW5:</u>

```
SW5#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      N - not in use, no aggregation
       f - failed to allocate aggregator

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

       A - formed by Auto LAG

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

Group  Port-channel  Protocol    Ports
------+-------------+-----------+---------------------------------------
5      Po5(SU)         PAgP      Et6/1(P)    Et6/2(P)
```

Task 10

Ensure that SW6's interface toward SW5 is used as the root port for all VLANs. If the link between SW5 and SW6 goes down, SW6 should immediately switch to using its link toward SW4.

After creating the port channel interfaces in the above task, the **show spanning-tree root port** command output is shown below. As seen, port channel 4 connected to SW4 is designated as the root port by SW6 for VLANs 1, 20 and 40. Port channel 5 is designated as the root port for all other VLANs:

On SW6:

```
SW6#show spanning-tree root port

VLAN0001                 Port-channel4
VLAN0010                 Port-channel5
VLAN0020                 Port-channel4
VLAN0030                 Port-channel5
VLAN0040                 Port-channel4
```

SW6 designates its port-channel 4 interface as the root port because of a lower root path cost over this interface. This can be calculated by observing the show spanning-tree detail output.

This task requires the EtherChannel link to SW5 to be designated as the root port. This can be achieved by ensuring that the cost to reach the root over the port-channel 5 is lower than that cost to reach the root over the port-channel 4 with the **spanning-tree vlan 1-4094 cost 5** command on SW6:

On SW6:

```
SW6(config)#interface port-channel5
SW6(config-if)#spanning-tree vlan 1-4094 cost 5

SW6#show spanning-tree root port

VLAN0001         Port-channel5
VLAN0010         Port-channel5
VLAN0020         Port-channel5
VLAN0030         Port-channel5
VLAN0040         Port-channel5
```

The **show spanning-tree root port** command verifies the same as seen above. Port channel 5 is now the root port for all VLANs as the task desires.

The task then hints towards configuring the uplinkfast feature on SW6. In the event of a link failure on the root port, the switch must transition one of its blocking ports to be the new root port through the Listening and Learning states before it can become fully operational and move to forwarding. This process can take up to 30 seconds (with default timers).

If the switch is guaranteed to be a leaf node switch, meaning no other switch uses it to transit to the root, there is little reason to move the new root port through the listening and learning states. The switch can bring its alternate connection to the root bridge up immediately.

Uplinkfast is a feature that can speed up this process by making note of all blocking ports on the switch that can be used as alternatives to the root bridge. This means the blocking ports are the redundant uplink ports leading to the root bridge. It calculates the best of these ports and uses it as a spare in case the current root port fails. When the root port fails, uplinkfast can immediately bring the new root port up without transitioning it from the blocking to the listening and then the learning states that takes 30 seconds.

Uplinkfast is enabled globally for the entire switch with the **spanning-tree uplinkfast** command. After doing so, the switch will automatically set its priority and port cost values artificially high in an attempt to discourage any other switch in the network from using it as a transit switch. The s**how spanning-tree summary | include Uplink** command can be used to verify the uplink status:

On SW6:

```
SW6(config)#spanning-tree uplinkfast

SW6#show spanning-tree summary | include Uplink

UplinkFast                                  is enabled
```

Task 11

Ensure that SW3's interfaces is designated on the SW2/SW3 link for all VLANs.

As we've seen so far, after electing the root bridge, each non-root switch must determine its root port, the port that leads directly to the root bridge. This is done by comparing the path costs for all links in the STP network leading to the root. The Root bridge originates BPDUs with a cost of 0 out of all of its ports and this cascades down to the non-root switches. They receive this BPDU and relay it to other switches, adding their own cost to the root to the BPDU. The switch gathers all received BPDUs on all ports and selects the port receiving the lowest cost as Root Port.

After electing the root bridge and root ports, the remaining switches need to determine on which non-root ports they should act as the Designated Bridge. The designated bridge in spanning-tree is the switch that is designated to relay BPDUs received from the root, downstream towards leaf node switches. They also are responsible for forwarding data traffic upstream towards the root and downstream from the root to leaf node switches. These ports are called Designated Ports.

NOTE: In a normal 802.1D STP implementation, only Designated Ports forward BPDUs.

The Designated Bridge is elected using the following criteria:

1. Lowest Root Path Cost to the Root Bridge
2. Lowest Sender Bridge ID
3. Lowest Sender Port ID

Observing just the **show spanning-tree vlan 1** output on SW2 and SW3 as an example, the current designated ports on the link between SW2 - SW3 is SW2's E3/1 - 2 interfaces. The E2/1 - 2 interfaces on

SW3 are in a blocking state. This simply means, SW2 is the designated bridge on the SW2 - SW3 segment for VLAN 1 and for all other VLANs.

Both SW2 and SW3 have a cost of 100 to reach the root bridge SW1. With the cost value equal, SW2 and SW3 use the second tie breaker of the election criteria to determine the designated bridge. Since SW2 has a lower bridge ID, it is elected as the designated bridge on this segment and designates its interfaces connected to SW3 as designated ports:

On SW2:

```
SW2#show spanning-tree vlan 1 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et1/1               Altn BLK 100       128.6    P2p
Et1/2               Root FWD 100       128.7    P2p
Et3/1               Desg FWD 100       128.14   P2p
Et3/2               Desg FWD 100       128.15   P2p
Et4/1               Desg FWD 100       128.18   P2p
Et4/2               Desg FWD 100       128.19   P2p
Et5/1               Desg FWD 100       128.22   P2p
Et5/2               Desg FWD 100       128.23   P2p
```

On SW3:

```
SW3#show spanning-tree vlan 1 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et1/1               Altn BLK 100       128.6    P2p
Et1/2               Root FWD 100       128.7    P2p
Et2/1               Altn BLK 100       128.10   P2p
Et2/2               Altn BLK 100       128.11   P2p
Et4/1               Desg FWD 100       128.18   P2p
Et4/2               Desg FWD 100       128.19   P2p
Et5/1               Desg FWD 100       128.22   P2p
Et5/2               Desg FWD 100       128.23   P2p
```

The task specifies to ensure that SW3 is elected as the designated bridge on the segment for all VLANs.

Since both SW2 and SW3 have the same cost to the root switch, this task can be solved by modifying the root cost on SW3 such that it is lower than the root cost on SW2 making SW3 the designated bridge on the segment. This can be done by modifying the port cost on SW3's current root port, E1/2, with the **spanning-tree cost 50**. The output below shows the port cost modification on SW3 and the "**show spanning-tree vlan 1"** output now confirms that SW3 is the designated bridge on the segment. Omitting the vlan number in the **spanning-tree cost** command sets the port cost to 50 for all VLANs on the switch. This results in SW3 becoming the designated bridge for the SW2 - SW3 segment for ALL VLANs as the task desires:

On SW3:

```
SW3(config)#interface e1/2
SW3(config-if)#spanning-tree cost 50

SW3#show spanning-tree vlan 1 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et1/1               Altn BLK 100       128.6    P2p
Et1/2               Root FWD 50        128.7    P2p
Et2/1               Desg FWD 100       128.10   P2p
Et2/2               Desg FWD 100       128.11   P2p
Et4/1               Desg FWD 100       128.18   P2p
Et4/2               Desg FWD 100       128.19   P2p
Et5/1               Desg FWD 100       128.22   P2p
Et5/2               Desg FWD 100       128.23   P2p
```

On SW2:

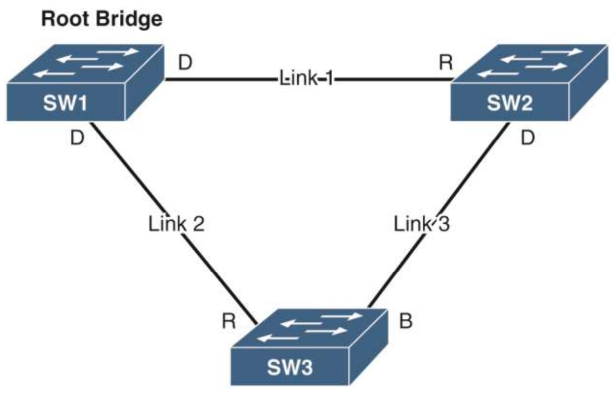
```
SW2#show spanning-tree vlan 1 | begin Interface

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
Et1/1               Altn BLK 100       128.6    P2p
Et1/2               Root FWD 100       128.7    P2p
Et3/1               Altn BLK 100       128.14   P2p
Et3/2               Altn BLK 100       128.15   P2p
Et4/1               Desg FWD 100       128.18   P2p
Et4/2               Desg FWD 100       128.19   P2p
Et5/1               Desg FWD 100       128.22   P2p
Et5/2               Desg FWD 100       128.23   P2p
```

It's important to mention at this point that this task could also have been solved by setting the priority value on SW3 to something lower than SW2's priority. By resetting the priority on SW3 to a value lower than SW2, SW2 would no longer be the secondary root bridge. This however would violate task 4 that requires SW2 to always be the secondary root bridge and take over as the root if the current root SW1 were to fail.

Task 12

Ensure that SW4 and SW5 can recover from an indirect link failure within 10 seconds.
Do not modify spanning-tree timers.

An indirect failure in STP is the loss of a root port for another switch that is not the local switch. Such a failure is detected whenever STP receives an inferior BPDU on any port. This is a result of the process that occurs whenever a switch loses connectivity to the root bridge. The switch experiencing the failure will begin to announce itself as the root bridge out of all of its ports. Consider the following topology for a better understanding of this process:



In the topology above, SW1 is the current root bridge. Link 1 on SW2 and Link 2 on SW3 are their current root ports respectively. Following details on what happens when SW2 loses its current root port, link 1:

1. On losing its root port towards SW1, SW2 starts considering itself as the root. This is because it does not receive any BPDUs relayed from SW3. SW3's port towards SW2 is blocking and blocking ports do not send any BPDUs
2. Claiming itself to be the root, SW2 now begins to send its BPDU to SW3. SW3 considers this to be an inferior BPDU to the BPDU it has stored on its blocking port. This causes SW3 to ignore the inferior BPDU from SW2 till the stored BPDU on the blocking port ages out. BPDUs age out in the Max Age - Message Age seconds in STP.

3. After the BPDU expires in the Max Age - Message Age seconds, the blocking port (link 3) on SW3 transitions to the listening state.
4. SW3 can now start sending the superior BPDUs from SW1 to SW2 on its link 3
5. SW2 receives this BPDU from SW3. Since the received BPDU is superior than the one SW2 generates, SW2 stops sending its own BPDU and moves its port towards SW3 from **listening → learning → forwarding** state.

As we see in the above, the convergence after an indirect link failure took Max Age - Message Age seconds (20 seconds at the first non-root switch and then decremented by 1 on every non-root switch downstream) plus the forward delay time (30 seconds). During this period, the end devices face network outage.

So using traditional STP, the switch receiving the inferior BPDU would have to wait for roughly 20 seconds before transitioning its port or recalculating the STP topology. The **Backbonefast** functionality allows a switch to bypass this delay in case of indirect link failures as in the example above. This is what the task expects to be configured.

The backbonefast functionality is able to bypass the max age delay by:

1. Detecting a link failure as soon as possible by tracking inferior BPDUs received on a blocking port
2. Immediately checking the validity of the current superior BPDU

The above is made possible by implementing another Cisco proprietary process called the **Root Link Query (RLQ)**. Referring back to the example, the following happens with the backbonefast feature activated on all the switches in the switched network:

1. On losing its root port on link 1 towards SW1, SW2 considers itself to be the root, generates BPDUs and forwards them out its designated port to SW3
2. On receiving this inferior BPDU, the backbonefast functionality on SW3 is triggered. SW3 needs to determine if the superior BPDU on its blocking port is still valid. If so, then SW3 can proceed to bypass the max age delay time and immediately age out the stored BPDU on its blocking port.
3. For this purpose, SW3 sends an RLQ request out of all its non-designated ports. This means, SW3 sends the RLQ request to SW1 out its root port on link 2. SW1 receives this and since it is still the current root bridge, it responds with an RLQ response.

4. On receiving the RLQ response, SW3 is now able to confirm that the inferior BPDU received from SW2 is wrong and invalidates it. It now immediately ages out the BPDU on the port and relays the superior BPDU to SW2.
5. SW2 receives this BPDU from SW3. Since the received BPDU is superior than the one SW2 generates, SW2 stops relaying its own BPDU and moves its port towards SW3 from **listening → learning → forwarding** state.

The RLQ request is basically sent to check the connectivity to the root bridge. It contains the BID of the switch the sending switch believes to be the root. If the receiving switch is the root, and the owner of the BID indicated in the RLQ request, it responds with a RLQ response. If the receiving switch is not the root, the roots BID it knows of is the same as the one indicated in the RLQ request, this switch forwards the query to the root.

Backbonefast is configured using the **spanning-tree backbonefast** and should be enabled on all the switches because backbonefast requires the use of RLQ request and reply mechanisms. This is shown in the configuration below. The **show spanning-tree summary** command output can be used to verify the same:

Note: Task 3 of this section reduced the forward delay time from 30 seconds to 10 seconds. So, for this lab, when the backbonefast feature is enabled, the switches will only spend 10 seconds (5 seconds in listening and 5 seconds in learning) to recover from an indirect failure:

On All Switches:

```
SW6(config)#spanning-tree backbonefast
```

To verify the configuration:

```
SWx#show spanning-tree summary | include Back

BackboneFast                             is enabled
```

Task 13

Ensure that SW2 only allows its ports toward SW1 to become root ports.

This task requires configuring the Root Guard feature on SW2's non-root ports.

In the same way a non-designated port receiving an inferior BPDU causes a topology change event, if any port receives a superior BPDU, a topology change event will occur. In particular, this will cause the switch to re-evaluate the location of its root port. This can be extremely devastating to the STP environment in certain situations.

The Root Guard feature is designed to mitigate this threat. When a port is configured with Root Guard it is automatically put in a **root inconsistent state** whenever it begins to receive superior BPDUs. This is similar to the listening state where no traffic is forwarded through the port but BPDUs are still processed and recieved. When the superior BPDUs cease, the port is put back into normal forwarding state by transitioning it through the transient listening and learning states.

This feature is best deployed on designated ports on the switch on a per port basis. Designated ports are ports that face away from the root bridge and as such should never become root ports.

To complete this task, configure spanning-tree guard root on all designated ports on SW2 as follows:

On SW2:

```
SW2(config)#interface range e4/1-2,e5/1-2
SW2(config-if-range)#spanning-tree guard root
```

***NOTE* Do not forget to reset the switch configurations to initial configurations upon completing this exercise.**

802.1w Per-VLAN Rapid Spanning Tree Protocol

So far, we have explored the processes and functionality of traditional Spanning Tree Protocol (also known as 802.1D). The original intent of 802.1D was to ensure that loopfree paths exist in a Layer 2 switched network where redundant links are utilized. 802.1D in its base form accomplishes this goal, but it does so at a price: convergence time. 802.1D utilizes timer-based convergence mechanics, which means ports must receive and evaluate BPDUs to determine the root bridge.

The winning BPDU is stored by each port and relayed out all designated ports on the switch until it is refreshed by the reception of the same BPDU on one of the switch's non-designated (blocking or root) ports. If a port does not receive a BPDU within the Max Age time, the BPDU is aged out, and the topology reconverges. Furthermore, a port that transitions from blocking to forwarding must first pass through listening and learning states.

The entire convergence process for 802.1D, with default timers, can take up to 50 seconds (20 seconds for Max Age time and 30 seconds for transitioning between listening and learning states). As mentioned earlier in the lab, this is a considerable amount of downtime for a modern network. It interferes with host operations such as acquiring a network address to use for data communication.

Recognizing this inefficiency, the IEEE developed the **802.1w** standard, which is also called the **Rapid Spanning Tree Protocol**, or **RSTP**. The goal of RSTP is to drastically reduce the amount of time it takes a network to converge during a convergence event and whenever a switch is newly joined to a network. To do so, RSTP makes a few changes to the 802.1D mechanics:

- BPDUs are sent by all switches independently of reception of the root bridge's BPDU.
- Listening and learning states are combined into a single **learning state**.
- Port roles more clearly define what function a specific port plays in the network.
- Timer-based convergence is replaced by a proposal/agreement process.

In 802.1D, switches do not originate BPDUs. Instead, they relay the received superior BPDU from their root port out their designated ports. This superior BPDU is first originated by the root bridge and acts as the heartbeat of the spanning tree. 802.1w modifies this behavior. All RSTP-compliant switches generate their best stored BPDUs out all of their designated ports at each hello interval, regardless of whether or not one was received on the root ports. This transforms the BPDU from being a measurement of the activity of the root bridge to being a keepalive between two bridges.

This modification of BPDU generation means RSTP can determine if a neighboring switch is active, based on when it last received a BPDU from the neighboring switch. If three Hello intervals of BPDUs are missed (2-second intervals, for 6 seconds total), the switch can immediately act. For blocking ports, that action is to become designated and send its own BPDU. Thus, in order for a blocking port to remain in the blocking state, it must continue to receive superior BPDUs from its upstream designated port.

In addition, the port states were revised. As mentioned earlier in 802.1D, there is little difference
between a port that is blocking and a port that is listening or learning. In each of these states, one of three actions is being performed:

- The port is not forwarding traffic (that is, it is discarding traffic).
- The port is learning about the STP topology while not forwarding traffic.
- The port is learning MAC addresses while not forwarding traffic.

The first two actions correspond to the port refusing to process data frames even to the point where MAC addresses are not being learned over the ports. Instead, state information about the spanning-tree topology is being evaluated. These two functions fall within the purview of the blocking and listening states of the original 802.1D. In 802.1w, they are combined into a simple discarding state to signify that data traffic is being discarded while spanning-tree BPDUs are still being processed. The last point corresponds to the switch processing MAC address information to build MAC address tables on the interfaces—a function of the learning state. This function was deemed necessary and has been retained as the learning state in RSTP.

With these modifications, RSTP possesses only three states: **discarding**, **learning**, and **forwarding**. These states describe what the port is actively doing but do not describe what function in the spanning-tree topology these ports serve. This distinction is necessary to allow rapid convergence in special circumstances. For this, RSTP utilizes unused fields in the 802.1D BPDU (the flags field) to carry the port state and new port roles describing both what action the port is taking and what role that port has in the spanning-tree topology.

The port roles are:

- **Root**: The port that receives the best BPDU of all BPDUs received by the local switch
- **Designated**: The port that sends the best BPDU on its LAN segment
- **Alternate**: The port that receives a superior BPDU on the LAN segment; it is a potential replacement for the root port
- **Backup**: The port that receives the superior BPDU of the local switch's own designated

port; it is a potential replacement for the switch's own designated port

Of all of the states, root and designated are the same and correspond to the root and designated states in 802.1D. Alternate and backup ports are synonymous with blocking ports in 802.1D, but their roles more clearly define where the port lies in the spanning tree. An alternate port is a port that receives a superior BPDU from another bridge that is not the best BPDU the switch has heard. Such ports provide alternative paths to the current root bridge.

A backup port is a port that is self-looped back to the sending local switch. This port could be connected to the same Layer 2 segment that does not speak Spanning Tree Protocol. For example, if a switch is connected to a set of hosts through a hub, the hub echoes all received frames out all ports except the port on which the frames were originally received. For redundancy, the switch could connect to the hub through two ports. If such a situation occurs, the BPDU sent by the switch on port A connected to the hub would be echoed and received on port B, connected to the same hub. If port A is determined to have the superior BPDU, port B would have the backup role because it provides a redundant path to a LAN segment that does not lead back to the root bridge.

A port in 802.1w can be in any mixture of states and roles. For instance, a port can be in the designated discarding role/state, which means it is a port that the switch believes should be designated but has not transitioned to forwarding. This fact is important for RSTP's convergence algorithm.

In 802.1D, convergence is based on a timer-driven state machine. When a switch comes online, it sends BPDUs claiming to be root. When it receives a superior BPDU, the ports must transition from blocking, to listening, to learning, and finally to forwarding, based on the Forward Delay timer (which defaults to 15 seconds, for 30 seconds total). If a switch were to lose connection to the root port, it would announce itself as root toward its neighbors. The neighbors would ignore this information for the Max Age period (which is 20 seconds by default) before reacting to the topology change. The goal is to allow the network to converge before a port is placed into the forwarding state.

802.1w does not use the same process as 802.1D but uses a newer proposal and agreement process that goes as such:

1. A new link port on a switch tries to move from the blocking state to the forwarding state.
2. The port receives a superior BPDU from the root bridge.
3. All other non-edge ports are blocked on the local switch.
4. Once all other switch ports are blocked, the local switch authorizes the root switch to put its port into the forwarding state.
5. The same process occurs on all of the local switch's remaining non-edge ports.

In **step 1**, the new link initializes in the **designated discarding** state. It exchanges BPDUs with the current root's port (which is also in the designated discarding state). During this time, both switches send a BPDU with the proposal bit set as an indication that they want their ports to become the designated port on the segment. Upon receipt of the superior BPDU at step 2, the local switch knows where its root port lies.

In **step 3**, the switch must ensure no loops can occur in the network based on this new information. To do so, it places all of its non-edge ports into the discarding state; this is called synchronization. Throughout this process, the root switch's port is still in the designated
discarding state.

At **step 4**, the local switch tells the root switch, through a BPDU with the agreement flag set, that the root switch's port can be moved to the designated forwarding state. This happens because the local switch has blocked all of its other non-edge ports, preventing any bridging loops.

At **step 5**, the local switch sends a BPDU with the proposal bit set out all of its remaining designated discarding ports to start the rapid convergence process with other potential spanning-tree bridges downstream from the root. Step 5 repeats the proposal/agreement process with each switch to which the local switch has a direct connection. The same process occurs: The switches exchange proposal BPDUs, the losing switch enters the synchronization state, and it informs the designated switch it can move its port to the designated forwarding state. In this way, the synchronization process flows downstream from the root to the edge of the network.

This process relies heavily on the switch determining which ports are edge ports and which are non-edge ports. RSTP keeps track of this by assigning an edge variable and link type status to each switch port. The edge variable indicates whether or not the port leads to an end host or sits at the edge of the network. Such ports do not connect to other switches and should not receive BPDUs. If an edge port receives a BPDU, it loses its edge status. Edge ports are allowed to immediately transition to a forwarding state; this is similar to the spanning-tree PortFast feature.

Link type refers to whether or not the switch port can use rapid transitions, as previously indicated. There are two link types: point-to-point and shared. A point-to-point link is connected to exactly one other RSTP-compliant switch. A shared port is connected to a shared LAN segment that utilizes hubs or repeaters and cannot transition rapidly, as in the previous proposal agreement process.

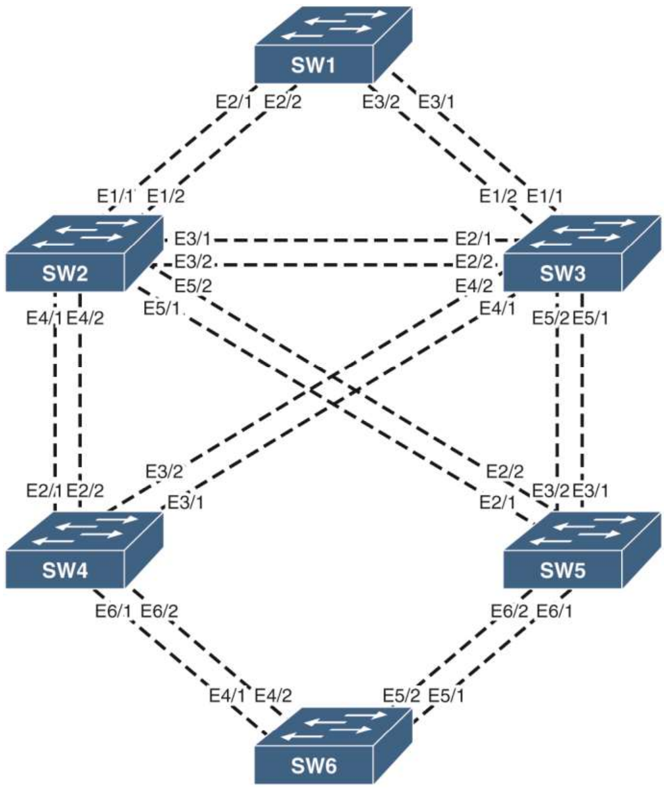
Switches attempt to detect link type by using the duplex setting of the interface: Halfduplex is considered shared, and full-duplex is considered point-to-point.

802.1w also achieves rapid convergence by modifying what constitutes a topology change in the network. In 802.1D, loss of a port or a port transitioning to blocking is considered a topology change event. In 802.1w, only a non-edge (blocking or alternate) port transitioning to the forwarding state generates a topology change event. The reason is that the new non-edge port offers a new path in the network, and the remaining switches should synchronize their MAC address tables accordingly to reflect the change. The TC-while timer is started on the switch initiating the topology change. During this time, the switch sends BPDUs with the TC flag set out all of its designated and root ports.

Switches that receive these BPDUs immediately age out all MAC addresses on all ports except where the TC BPDU was received. This mechanism allows rapid transition of the topology because TC BPDUs are originated by the switch experiencing the topology change event and are not initiated by the root bridge, as is the case in 802.1D.
Finally, because BPDUs are necessary for a blocking port to remain blocking, if a blocking port receives an inferior BPDU, it can react immediately to the information. This is in contrast to the 802.1D specification, which requires the stored BPDU on a port to age out before the switch converges the topology. In 802.1w, the switch receiving the inferior BPDU can infer that a topology change event has occurred somewhere in the network.

If a functioning root port exists on the switch receiving an inferior BPDU, it can simply respond with a proposal BPDU on its formerly blocking port, asking to be set to designated. This allows the failed switch to recover rapidly. If there is no functioning root port (meaning the inferior BPDU was received on a root port), the switch can assume that it should be the new root switch and indicates that with all of its remaining, now downstream, neighbors.

These are the key enhancements to 802.1D built into 802.1w. Some of them may sound familiar as they relate to many of the Cisco enhancement features to 802.1D, such as PortFast and Backbone Fast. The following lab demonstrates these enhancement features of 802.1w and contrasts them with their 802.1D equivalents.



Initial Configuration:

Task 1

Change the hostname on each switch to SW#, where # is the number of the switch (for example, Switch 1 = SW1).

On SW1:

```
Switch(config)#hostname SW1
```

On SW2:

```
Switch(config)#hostname SW2
```

On SW3:

```
Switch(config)#hostname SW3
```

On SW4:

```
Switch(config)#hostname SW4
```

On SW5:

```
Switch(config)#hostname SW5
```

On SW6:

```
Switch(config)#hostname SW6
```

Task 2

a. Ensure that only the following ports on the switches are in an up/up state:

- SW1
  - E2/1-2
  - E3/1-2

On SW1:

```
SW1(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW1(config-if-range)#shut

SW1(config)#interface range e2/1-2,e3/1-2
SW1(config-if-range)#no shut
```

To verify the configuration:

```
SW1#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status         Protocol
Ethernet2/1            unassigned      YES unset  up             up
Ethernet2/2            unassigned      YES unset  up             up
Ethernet3/1            unassigned      YES unset  up             up
Ethernet3/2            unassigned      YES unset  up             up
```

- SW2
  - E1/1-2
  - E3/1-2
  - E4/1-2
  - E5/1-2

On SW2:

```
SW2(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW2(config-if-range)#shut

SW2(config)#interface range e1/1-2,e3/1-2,e4/1-2,e5/1-2
SW2(config-if-range)#no shut
```

To verify the configuration:

```
SW2#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status                Protocol
Ethernet1/1            unassigned      YES unset  up                    up
Ethernet1/2            unassigned      YES unset  up                    up
Ethernet3/1            unassigned      YES unset  up                    up
Ethernet3/2            unassigned      YES unset  up                    up
Ethernet4/1            unassigned      YES unset  up                    up
Ethernet4/2            unassigned      YES unset  up                    up
Ethernet5/1            unassigned      YES unset  up                    up
Ethernet5/2            unassigned      YES unset  up                    up
```

- SW3
  - E1/1-2
  - E2/1-2
  - E4/1-2
  - E5/1-2

On SW3:

```
SW3(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW3(config-if-range)#shut
```

```
SW3(config)#interface range e1/1-2,e2/1-2,e4/1-2,e5/1-2
SW3(config-if-range)#no shut
```

To verify the configuration:

```
SW3#show ip interface brief | ex down

Interface              IP-Address      OK? Method Status       Protocol
Ethernet1/1            unassigned      YES unset  up           up
Ethernet1/2            unassigned      YES unset  up           up
Ethernet2/1            unassigned      YES unset  up           up
Ethernet2/2            unassigned      YES unset  up           up
Ethernet4/1            unassigned      YES unset  up           up
Ethernet4/2            unassigned      YES unset  up           up
Ethernet5/1            unassigned      YES unset  up           up
Ethernet5/2            unassigned      YES unset  up           up
```

- SW4
  - E2/1-2
  - E3/1-2
  - E6/1-2

On SW4:

```
SW4(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW4(config-if-range)#shut

SW4(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW4(config-if-range)#no shut
```

To verify the configuration:

```
SW4#show ip interface brief | ex down

Interface              IP-Address      OK? Method Status       Protocol
Ethernet2/1            unassigned      YES unset  up           up
Ethernet2/2            unassigned      YES unset  up           up
Ethernet3/1            unassigned      YES unset  up           up
```

```
Ethernet3/2            unassigned      YES unset  up            up
Ethernet6/1            unassigned      YES unset  up            up
Ethernet6/2            unassigned      YES unset  up            up
```

- SW5
  - E2/1-2
  - E3/1-2
  - E6/1-2

On SW5:

```
SW5(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW5(config-if-range)#shut

SW5(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW5(config-if-range)#no shut
```

To verify the configuration:

```
SW5#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status        Protocol
Ethernet2/1            unassigned      YES unset  up            up
Ethernet2/2            unassigned      YES unset  up            up
Ethernet3/1            unassigned      YES unset  up            up
Ethernet3/2            unassigned      YES unset  up            up
Ethernet6/1            unassigned      YES unset  up            up
Ethernet6/2            unassigned      YES unset  up            up
```

- SW6
  - E4/1-2
  - E5/1-2

On SW6:

```
SW6(config)#interface range e0/0-3,e1/0-3,e2/0-3,e3/0-3,e4/0-3,e5/0-3,e6/0-
3,e7/0-3
SW6(config-if-range)#shut

SW6(config)#interface range e4/1-2,e5/1-2
SW6(config-if-range)#no shut
```

To verify the configuration:

```
SW6#show ip interface brief | exclude down

Interface              IP-Address      OK? Method Status         Protocol
Ethernet4/1            unassigned      YES unset  up             up
Ethernet4/2            unassigned      YES unset  up             up
Ethernet5/1            unassigned      YES unset  up             up
Ethernet5/2            unassigned      YES unset  up             up
```

Task 3

Configure VLANs 10, 20, 30, and 40 on each switch, using any method.

The task requires VLANs 10, 20, 30 and 40 to be configured on all the switches in the topology. As seen below, these VLANs are configured on all the switches with the **vlan** command. However since the task does not place any restriction, VTP can be used to propagate the VLANs between switches. The **show vlan brief | include VLAN** command output can then be used to verify this configuration:

On All Switches:

```
SW1(config)#vlan 10,20,30,40
SW1(config-vlan)#exit
```

To verify the configuration:

On Any Switch:

```
SWx#show vlan brief | include VLAN
```

```
VLAN Name                             Status    Ports
10   VLAN0010                         active
20   VLAN0020                         active
30   VLAN0030                         active
40   VLAN0040                         active
```

Task 4

Configure trunk ports on all up/up interfaces.

The commands **switchport trunk encapsulation dot1q** and **switchport mode trunk** is configured on all the interfaces on all the switches that were brought back up earlier. The combination of these commands manually configure the interfaces to function as 802.1q trunk links.

On SW1:

```
SW1(config)#interface range e2/1-2,e3/1-2
SW1(config-if-range)#switchport trunk encapsulation dot1q
SW1(config-if-range)#switchport mode trunk
```

On SW2:

```
SW2(config)#interface range e1/1-2,e3/1-2,e4/1-2,e5/1-2
SW2(config-if-range)#switchport trunk encapsulation dot1q
SW2(config-if-range)#switchport mode trunk
```

On SW3:

```
SW3(config)#interface range e1/1-2,e2/1-2,e4/1-2,e5/1-2
SW3(config-if-range)#switchport trunk encapsulation dot1q
SW3(config-if-range)#switchport mode trunk
```

On SW4:

```
SW4(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW4(config-if-range)#switchport trunk encapsulation dot1q
SW4(config-if-range)#switchport mode trunk
```

On SW5:

```
SW4(config)#interface range e2/1-2,e3/1-2,e6/1-2
SW4(config-if-range)#switchport trunk encapsulation dot1q
SW4(config-if-range)#switchport mode trunk
```

<u>On SW6:</u>

```
SW5(config)#interface range e4/1-2,e5/1-2
SW5(config-if-range)#switchport trunk encapsulation dot1q
SW5(config-if-range)#switchport mode trunk
```

802.1w – Configuration Tasks

Task 1

Configure all switches to run 802.1w Rapid Spanning Tree Protocol.

The default spanning-tree mode on most switching platforms is 802.1D Spanning-Tree. In order to activate 802.1w Rapid Spanning-Tree, issue the command **spanning-tree mode rapid-pvst** command in global configuration mode as follows. The **show spanning-tree** output is then used to verify this change as seen below:

On All Switches:

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

To verify the configuration:

On All Switches:

```
SWx#show spanning-tree  | include VLAN|protocol

VLAN0001
  Spanning tree enabled protocol rstp
VLAN0010
  Spanning tree enabled protocol rstp
VLAN0020
  Spanning tree enabled protocol rstp
VLAN0030
  Spanning tree enabled protocol rstp
VLAN0040
  Spanning tree enabled protocol rstp
```

Task 2

Ensure that SW1 is the root for all VLANs.

RSTP uses the same logic as 802.1D spanning-tree to elect a root switch, choosing the switch with the lowest BID in the switched network as the root bridge.

Similar to the same task in the PVST+ section, this task requires no extra configuration since SW1 has already been designated as the root bridge for all VLANs. This decision was made based on SW1's MAC address as all switches have a tied priority value of 32,768.

NOTE: If SW1 is not the root bridge in your topology, use the spanning-tree vlan 1-4094 root primary command to make it the root bridge.

A quick way to determine if the local switch is the root bridge for all VLANs is to use the **show spanning-tree root** command. There is no need to know the MAC address of the local switch when using this command. Because the root switch advertises a cost of 0 to reach itself out of all of its Designated ports, all VLANs in the **show spanning-tree root** command output that have 0 listed in the "Root Cost" column are VLANs for which the local switch is the root bridge as seen below:

On SW1:

```
SW1#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root
Port
---------------- -------------------- --------- ----- --- ---  ------
VLAN0001         32769 aabb.cc00.1f00         0    2   20  15
VLAN0010         32778 aabb.cc00.1f00         0    2   20  15
VLAN0020         32788 aabb.cc00.1f00         0    2   20  15
VLAN0030         32798 aabb.cc00.1f00         0    2   20  15
VLAN0040         32808 aabb.cc00.1f00         0    2   20  15
```

Task 3

Ensure that if the current root bridge fails, SW2 becomes the new root for all VLANs.

Once again, similar to Task 4 in the PVST+ section, SW2 can be configured to be the secondary bridge with the **spanning-tree vlan [vlan list] root secondary** command for all VLANs. Following this, to ensure that SW1 retains its root status, the command **spanning-tree vlan [vlan list] root primary** should be configured on SW1 for all VLANs:

On SW2:

```
SW2(config)#spanning-tree vlan 1-4094 root secondary
```

On SW1:

```
SW1(config)#spanning-tree vlan 1-4094 root primary
```

Task 4

Ensure that SW2's and SW3's E1/2 interface is used as the root port for all VLANs. Do not modify cost to achieve this.

As explained in task 5 of the PVST+ section, SW2 and SW3 use the SPID value as the tiebreaker to elect their E1/1 interface as the root port towards the root bridge. The **show spanning-tree root** command from SW2 and SW3 verifies this:

On SW2:

```
SW2#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0001         24577 aabb.cc00.1f00       100    2   20  15  Et1/1
VLAN0010         24586 aabb.cc00.1f00       100    2   20  15  Et1/1
VLAN0020         24596 aabb.cc00.1f00       100    2   20  15  Et1/1
VLAN0030         24606 aabb.cc00.1f00       100    2   20  15  Et1/1
VLAN0040         24616 aabb.cc00.1f00       100    2   20  15  Et1/1
```

On SW3:

```
SW3#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0001         24577 aabb.cc00.1f00       100    2   20  15  Et1/1
```

```
VLAN0010         24586 aabb.cc00.1f00       100    2   20  15  Et1/1
VLAN0020         24596 aabb.cc00.1f00       100    2   20  15  Et1/1
VLAN0030         24606 aabb.cc00.1f00       100    2   20  15  Et1/1
VLAN0040         24616 aabb.cc00.1f00       100    2   20  15  Et1/1
```

This task requires making configuration changes such that SW2 and SW3 designate their E1/2 interfaces as the root port. However, unlike the PVST+ section, this task restricts modifying the cost to make the E1/2 interfaces on SW2 and SW3 the root port. The task, however, places no restriction on modifying the SPIDs.

The **spanning-tree vlan 1-4094 port-priority 64** command on the E2/2 and E3/2 interfaces on SW1 will cause it to assign 64 as the port-priority for these interfaces. Since this is lower than the default port-priority value of 128, when comparing the SPIDs on the BPDUs received from SW1 on their E1/1 and E1/2 interfaces, both SW2 and SW3 will designate their E1/2 interfaces as the root port for all VLANs. This is verified with the **show spanning-tree root** command output below:

On SW1:

```
SW1(config)#interface e1/2
SW1(config-if)#spanning-tree vlan 1-4094 port-priority 64
```

On SW2:

```
SW2#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
-
VLAN0001         24577 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0010         24586 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0020         24596 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0030         24606 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0040         24616 aabb.cc00.1f00       100    2   20  15  Et1/2
```

On SW3:

```
SW3#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
-
```

```
VLAN0001         24577 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0010         24586 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0020         24596 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0030         24606 aabb.cc00.1f00       100    2   20  15  Et1/2
VLAN0040         24616 aabb.cc00.1f00       100    2   20  15  Et1/2
```

Task 5

Ensure that SW5 uses the following ports as root:

- E2/1 for VLAN 10
- E2/2 for VLAN 20
- E3/1 for VLAN 30
- E3/2 for VLAN 40

Do not modify SW5 to achieve any of this.

By default, the E2/1 interface on SW5 is designated as the root port for all VLANs as seen in the **show spanning-tree output** below:

On SW5:

```
SW5#show spanning-tree root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0001         24577 aabb.cc00.1f00       200    2   20  15  Et2/1
VLAN0010         24586 aabb.cc00.1f00       200    2   20  15  Et2/1
VLAN0020         24596 aabb.cc00.1f00       200    2   20  15  Et2/1
VLAN0030         24606 aabb.cc00.1f00       200    2   20  15  Et2/1
VLAN0040         24616 aabb.cc00.1f00       200    2   20  15  Et2/1
```

The reasoning for electing the E2/1 interface as the root port is as follows: SW5 receives BPDUs from SW2 and SW3 over its E2/1-2 and E3/1-2 interfaces. BPDUs from SW6 are the most inferior, so will not be considered in the following explanation.

The cost to reach the root from both SW2 and SW3 equals to 100 from SW5. As a result of this tie in cost, SW5 uses the next tie breaker, and chooses the switch with lower SBID which is SW2. SW5 then uses the next tiebreaker, the lower SPID to determine which of its ports towards SW2 (E2/1, E2/2) should be designated as the root. Since the SPID of SW2's E5/1 interface is lower than the SPID of its E5/2 interface, SW5 designates its E2/1 interface as the root port for all VLANs. The E2/2 interface is put in an alternate blocking state. The example output for VLAN 1 details and confirms the lower SPID of SW2's E5/1 interface.

```
SW5#show spanning-tree vlan 1 detail | section Ethernet2/1

  Root port is 10 (Ethernet2/1), cost of root path is 200
          from Ethernet2/1
 Port 10 (Ethernet2/1) of VLAN0001 is root forwarding
   Port path cost 100, Port priority 128, Port Identifier 128.10.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 28673, address aabb.cc00.2000
   Designated port id is 128.22, designated path cost 100
   Timers: message age 15, forward delay 0, hold 0
   Number of transitions to forwarding state: 3
   Link type is point-to-point by default
   BPDU: sent 25, received 48781

 Port 11 (Ethernet2/2) of VLAN0001 is alternate blocking
   Port path cost 100, Port priority 128, Port Identifier 128.11.
   Designated root has priority 24577, address aabb.cc00.1f00
   Designated bridge has priority 28673, address aabb.cc00.2000
   Designated port id is 128.23, designated path cost 100
   Timers: message age 15, forward delay 0, hold 0
   Number of transitions to forwarding state: 0
   Link type is point-to-point by default
   BPDU: sent 29, received 47912
```

The following sub tasks require changing the root ports for various VLANs on SW5:

1. e2/1 for VLAN 10

This task requires no extra configuration as the E2/1 interface is already being used as the root port for VLAN 10. The **show spanning-tree vlan 10** output confirms this:

On SW5:

```
SW5#show spanning-tree vlan 10 root

                                       Root    Hello Max Fwd
Vlan                    Root ID        Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0010          24586 aabb.cc00.1f00       200    2   20  15  Et2/1
```

2. e2/2 for VLAN 20

The task restricts making any modifications on SW5. So one of the ways to ensure E2/2 is designated as the root port for VLAN 20 on SW5, is to lower the port-priority value of SW2's E5/2 interface to a value lower than the default value of 128 on its E5/1 interface.This is achieved with the interface-level command **spanning-tree vlan 20 port-priority 64** as seen below:

On SW2:

Port Priority is set to 64 for SW2's E5/2 interface

```
SW2(config)#interface e5/2
SW2(config-if)#spanning-tree vlan 20 port-priority 64
```

On SW5:

```
SW5#show spanning-tree vlan 20 root

                                        Root     Hello Max Fwd
Vlan                   Root ID          Cost     Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0020         24596 aabb.cc00.1f00       200    2   20  15  Et2/2

SW5#show spanning-tree vlan 20 interface e2/1

Vlan                Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- -----
VLAN0020            Altn BLK 100       128.10   P2p
```

As a result, SW5 now uses its E2/2 interface as the root port and the E2/1 interface is now in an alternate blocking state.

3. e3/1 for VLAN 30

The cost to reach the root bridge SW1 from SW2 and SW3 is equal. SW5 uses the tiebreaker of comparing the SBIDs and designates its E2/1 interface as the root port. This task requires that SW5 uses the E3/1 interface connected to SW3 to reach the root. This can be achieved by modifying the BID of SW3 so that it has a lower BID than SW2's BID. However, this would violate Task 3 which requires that SW2 should take over as the root in case the current root fails.

So another way to solve this task would be modifying SW3's cost to the root for VLAN 30 to something lower than SW2's cost to the root. The current cumulative cost to the root from SW5 is 200 via SW2 as shown below.

On SW5:

```
SW5#show spanning-tree vlan 30 root
                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0030         24606 aabb.cc00.1f00       200    2   20  15  Et2/1
```

The **spanning-tree vlan 30 cost 99** on SW3's current root port would result in a lower cumulative cost of 199 to reach the root via SW3 for SW5. After this change, SW5 uses the lower SPID tiebreaker and designates its E3/1 interface as the root port. This is verified in the **show spanning-tree vlan 30** command output below:

On SW3:

```
SW3(config)#interface e1/2
SW3(config-if)#spanning-tree vlan 30 cost 99
```

On SW5:

```
SW5#show spanning-tree vlan 30 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0030         24606 aabb.cc00.1f00       199    2   20  15  Et3/1
```

Note: This task could have been solved by modifying the cost directly on SW5. However, since the task restricts making any changes to SW5, the options are to either set a lower RPC on SW3 or a higher RPC on SW2.

4. e3/2 for VLAN 40

Once again, the current root port for VLAN 40 on SW5 is its E2/1 interface as seen below:

On SW5:

```
SW5#show spanning-tree vlan 40 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0040         24616 aabb.cc00.1f00       200    2   20  15  Et2/1
```

This sub task requires SW5 to designate its E3/2 interface as the root port. To influence SW5's decision, the following two configuration changes are made on SW3 while complying with the task restriction of not making any modification to SW5:

1. SW3's cost to the root is modified to make it lower than SW2's cost to the root, resulting in SW5 choosing SW3 to reach the root. This is achieved with the **spanning-tree vlan 40 cost 99** command on SW3's root port E1/2
2. The port ID of SW3's E5/2 interface is configured to be lower than the default port ID of 128 on its E5/1 interface for VLAN 40 with the **spanning-tree vlan 40 port-priority 64** command. This would result in SW5 choosing its E3/2 interface as the root port as seen below:

On SW3:

```
SW3(config)#interface e1/2
SW3(config-if)#spanning vlan 40 cost 99
```

The above configuration change results in SW5 choosing its E3/1 interface connected to SW3 as the root port:

On SW5:

```
SW5#show spanning-tree vlan 40 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0040         24616 aabb.cc00.1f00       199    2   20  15  Et3/1
```

Finally, the port priority for SW3's E5/2 interface is made lower to make the E3/2 port on SW5 more preferable, hence, the root port:

On SW3:

```
SW3(config)#interface e5/2
SW3(config-if)#spanning-tree vlan 40 port-priority 64
```

To verify the configuration:

On SW5:

```
SW5#show spanning-tree vlan 40 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0040         24616 aabb.cc00.1f00       199    2   20  15  Et3/2
```

Task 6

Ensure that SW4 uses its E3/1 port as the root port for VLAN 20.

The **show spanning-tree vlan 20 root** command output below shows the current root port on SW4 is its E2/1 for VLAN 20. Since the cost to reach the root via SW2 and SW3 tie with the value of 200, SW4 uses the next tie breaker of comparing the SBIDs and chooses to use a port towards SW2 (lower SBID) as the root:

On SW4:

```
SW4#show spanning-tree vlan 20 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0020         24596 aabb.cc00.1f00       200    2   20  15  Et2/1
```

This task requires SW4 to use the interface E3/1 connected to SW3 as the root port. SBIDs cannot be modified to make SW3 more preferable as this would violate Task 3. So, the easiest way to solve this task is to modify the port cost on SW4's E3/1 interface such that it is lower than the cost to reach the root over its E2/1 interface. The cumulative cost to reach the root is 200 as seen above and the default port cost is 100.

To make the E3/1 interface more preferable, the port cost on SW4's E3/1 interface is reduced to 99 with the **spanning-tree vlan 20 cost 99** command:

On SW4:

```
SW4(config)#interface e3/1
SW4(config-if)#spanning-tree vlan 20 cost 99
```

Notice the cost below to reach the root is 199 which is lower than the cost to reach the root over its E2/1 interface

```
SW4#show spanning-tree vlan 20 root

                                        Root    Hello Max Fwd
Vlan                   Root ID          Cost    Time  Age Dly  Root Port
---------------- -------------------- --------- ----- --- ---  ---------
VLAN0020         24596 aabb.cc00.1f00       199    2   20  15  Et3/1
```

As seen above, with the port cost modification on SW4, the total cumulative cost to reach the root over it's E3/1 interface is now 199 resulting in SW4 designating this port as the root port.

Task 7

Ensure that RSTP features are enabled on all ports.

The beginning of this configuration lab explained the process RSTP uses to achieve rapid convergence. This rapid convergence comes at a strict restriction. It only works between links between two switches. If, for example, three switches were connected by a hub, there would be no way for the three switches to signal a proposal or agreement to each other. The resulting spanning-tree topology would be difficult to work out. For this reason, RSTP includes two important link types: **shared** and **point-to-point**.

A switch with a shared link connects to multiple neighboring switches. Due to the nature of such ports, the RSTP convergence enhancements are not used. Instead, the normal timer-based convergence processes are employed. A switch with a point-to-point link connects to a single neighboring switch. These links are candidates for rapid convergence and take full advantage of RSTPs convergence enhancements. The **show spanning-tree output** lists the ports, their role, state, cost, and link type information as in the example output below:

<u>On SW1:</u>

```
SW1#show spanning-tree vlan 1

VLAN0001
  Spanning tree enabled protocol rstp
  Root ID    Priority    24577
             Address     aabb.cc00.1f00
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    24577  (priority 24576 sys-id-ext 1)
             Address     aabb.cc00.1f00
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- ------------------------
Et2/1               Desg FWD 100       128.10   P2p
Et2/2               Desg FWD 100        64.11   P2p
Et3/1               Desg FWD 100       128.14   P2p
Et3/2               Desg FWD 100        64.15   P2p
```

In the above, the “Type” column indicates what link type the interface is operating under. The value is “shared” for shared link type and “P2p” for point-to-point link type.

The switch attempts to predict what kind of link each port has based on the duplex setting. If the port is in half-duplex mode, the switch considers the link to be a shared link. If the link is in full-duplex mode, the switch considers a link to be a point-to-point link. This is a good assumption considering the only way to connect multiple switches to the same port is to use an ethernet hub. Ethernet hubs force half-duplex operation because all ports on a hub share the same broadcast domain and the CSMA-CD algorithm is used to organize transmission on the wire. This leaves ports in full-duplex mode to only have a single switch or other device connected and thus can be assumed to be point-to-point.

There are times where this assumption is not true especially in emulated environments where the auto MDIX process may not work properly. If the auto MDIX process fails, ports fall back to half-duplex and thus have a shared link type which bypasses all RSTP rapid convergence enhancements. In such a case, the

spanning-tree link-type [point-to-point | shared] interface-level command can be used to manually set the link type in the topology.

The task specifies all ports should be candidates to benefit from all of RSTP's rapid convergence properties. As such, the spanning-tree link-type point-to-point interface-level command should be configured on all switch ports. However, looking at the output above from the lab topology, the switch has already appropriately designated the ports as point-to-point and as a result, no further configuration is required.

NOTE: If there are ports that are NOT in designated as point-to-point (P2p) in the **show spanning-tree output**, use the **spanning-tree link-type point-to-point** command in interface configuration mode for those ports.

Task 8

Ensure that interfaces connected to non-switch hosts come online immediately.

a. If SW4 or SW5 detects a switch on these ports when it first comes online, it should process the BPDUs normally. Otherwise, it should not process received BPDUs.
b. If SW6 detects a switch on one of these ports, it should disable the port.

In 802.1D spanning-tree protocol, whenever a port moved from blocking to forwarding state it must first move through the listening and learning states, a process that could take up to 30s with default timers. Cisco implemented the **portfast** feature which allowed specifically configured ports to bypass these states and move straight from blocking to forwarding.

The developers of RSTP borrowed this concept and created what is known as an **edge port**. Edge ports are ports that connect to edge devices (end user stations, servers, APs, etc.). Such ports have a low possibility of causing loops in the network and can move instantly to forwarding state. Edge ports are also very important to RSTP because of the proposal/agreement process that allows fast convergence.

Recall that once a switch determines the location of its root port it first sets all of its non-edge ports to discarding state. After setting non-edge ports to discarding state, the switch signals to its upstream designated switch that it is safe to move the designated switch's port to forwarding. The key part of the process is that non-edge ports are set to discarding and not the edge ports themselves. Because edge