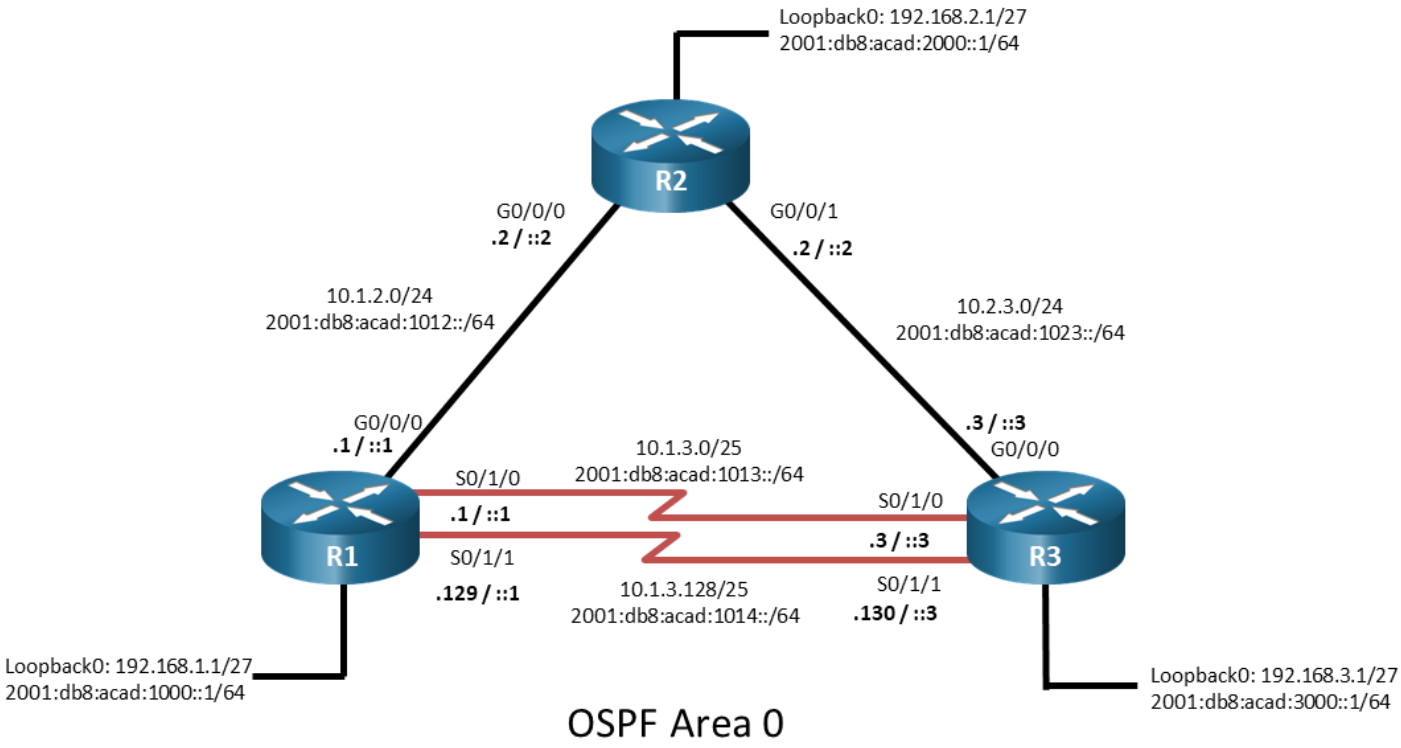
Lab - Use Connectivity Tests and Debug for Network Assurance (Instructor Version)

Instructor Note: Red font color or gray highlights indicate text that appears in the instructor copy only.

Answers: [24.1.2 Lab - Use Connectivity Tests and Debug for Network Assurance]

Topology



Addressing Table

Device	Interface	IPv4 Address	IPv6 Address	IPv6 Link-Local
R1	G0/0/0	10.1.2.1/24	2001:db8:acad:1012::1/64	fe80::1:1
	S0/1/0	10.1.3.1/25	2001:db8:acad:1013::1/64	fe80::1:2
	S0/1/1	10.1.3.129/25	2001:db8:acad:1014::1/64	fe80::1:3
	Loopback0	192.168.1.1/24	2001:db8:acad:1000::1/64	fe80::1:4
	Loopback1	172.16.1.1/24	2001:db8:acad:1721::1/64	fe80::1:5
R2	G0/0/0	10.1.2.2/24	2001:db8:acad:1012::2/64	fe80::2:1
	G0/0/1	10.2.3.2/24	2001:db8:acad:1023::2/64	fe80::2:2

Device	Interface	IPv4 Address	IPv6 Address	IPv6 Link-Local
	Loopback0	192.168.2.1/24	2001:db8:acad:2000::1/64	fe80::2:3
R3	G0/0/0	10.2.3.3/24	2001:db8:acad:1023::3/64	fe80::3:1
	S0/1/0	10.1.3.3/25	2001:db8:acad:1013::3/64	fe80::3:2
	S0/1/1	10.1.3.130/25	2001:db8:acad:1014::3/64	fe80::3:3
	Loopback0	192.168.3.1/24	2001:db8:acad:3000::1/64	fe80::3:4

Objectives

Part 1: Build the Network and Configure Basic Device Settings and Interface Addressing

Part 2: Explore Ping Options and Extended Ping Commands

Part 3: Explore Traceroute Options and Extended Traceroute Commands

Part 4: Explore Common Debug Commands and Conditional Debugging

Part 5: Troubleshoot OSPF with Debugging

Background / Scenario

You should be familiar with the ping and traceroute utilities, and with some debug commands. These are essential tools in your troubleshooting toolkit. In this lab, we will explore beyond the basic commands to examine options available with each of the utilities. Mastering these utilities will help to improve your troubleshooting skills.

Note: This lab is an exercise in using permutations of ping, traceroute, and debug, and does not necessarily reflect network troubleshooting best practices.

Note: The routers used with CCNP hands-on labs are Cisco 4221 with Cisco IOS XE Release 16.9.4 (universalk9 image). Other routers, and Cisco IOS versions can be used. Depending on the model and Cisco IOS version, the commands available and the output produced might vary from what is shown in the labs.

Note: Make sure that the routers have been erased and have no startup configurations. If you are unsure, contact your instructor.

Instructor Note: Refer to the Instructor Lab Manual for the procedures to initialize and reload devices.

Required Resources

- 3 Routers (Cisco 4221 with Cisco IOS XE Release 16.9.4 universal image or comparable)
- 1 PC (Choice of operating system with a terminal emulation program installed)
- Console cables to configure the Cisco IOS devices via the console ports
- Ethernet and serial cables as shown in the topology

Instructions

Part 1: Build the Network and Configure Basic Device Settings and Interface Addressing

In Part 1, you will set up the network topology and configure basic settings and interface addressing on routers.

Step 1: Cable the network as shown in the topology.

Attach the devices as shown in the topology diagram, and cable as necessary.

Step 2: Configure basic settings for each router.

a. Console into each router, enter global configuration mode, and apply the basic settings. The startup configurations for each device are provided below.

Router R1

```
hostname R1
no ip domain lookup
ipv6 unicast-routing
banner motd # R1, Using Ping, Traceroute, and Debug #
line con 0
 exec-timeout 0 0
 logging synchronous
 exit
line vty 0 4
 privilege level 15
 password cisco123
 exec-timeout 0 0
 logging synchronous
 login
 exit
interface g0/0/0
 ip address 10.1.2.1 255.255.255.0
 ipv6 address fe80::1:1 link-local
 ipv6 address 2001:db8:acad:1012::1/64
 no shutdown
 exit
interface s0/1/0
 ip address 10.1.3.1 255.255.255.128
 ipv6 address fe80::1:2 link-local
 ipv6 address 2001:db8:acad:1013::1/64
 no shutdown
 exit
interface s0/1/1
 ip address 10.1.3.129 255.255.255.128
 ipv6 address fe80::1:3 link-local
 ipv6 address 2001:db8:acad:1014::1/64
 no shutdown
 exit
interface loopback 0
 ip address 192.168.1.1 255.255.255.0
 ipv6 address fe80::1:4 link-local
 ipv6 address 2001:db8:acad:1000::1/64
 no shutdown
```

```
 exit
interface loopback 1
 ip address 172.16.1.1 255.255.255.0
 ipv6 address fe80::1:5 link-local
 ipv6 address 2001:db8:acad:1721::1/64
 no shutdown
 exit
router ospf 4
 router-id 1.1.1.4
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.1.0 0.0.0.255 area 0
 exit
ipv6 router ospf 6
 router-id 1.1.1.6
 interface g0/0/0
  ipv6 ospf 6 area 0
  exit
 interface s0/1/0
  ipv6 ospf 6 area 0
  exit
 interface s0/1/1
  ipv6 ospf 6 area 0
  exit
 interface loopback 0
  ipv6 ospf 6 area 0
  ipv6 ospf network point-to-point
  exit
ip route 172.16.3.0 255.255.255.0 g0/0/0 10.1.2.2
end
```

Router R2

```
hostname R2
no ip domain lookup
ipv6 unicast-routing
banner motd # R2, Using Ping, Traceroute, and Debug #
line con 0
 exec-timeout 0 0
 logging synchronous
 exit
line vty 0 4
 privilege level 15
 password cisco123
 exec-timeout 0 0
 logging synchronous
 login
 exit
```

```
interface g0/0/0
 ip address 10.1.2.2 255.255.255.0
 ipv6 address fe80::2:1 link-local
 ipv6 address 2001:db8:acad:1012::2/64
 no shutdown
 exit
interface g0/0/1
 ip address 10.2.3.2 255.255.255.0
 ipv6 address fe80::2:2 link-local
 ipv6 address 2001:db8:acad:1023::2/64
 no shutdown
 exit
interface loopback 0
 ip address 192.168.2.1 255.255.255.0
 ipv6 address fe80::2:3 link-local
 ipv6 address 2001:db8:acad:2000::1/64
 no shutdown
 exit
router ospf 4
 router-id 2.2.2.4
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.2.0 0.0.0.255 area 0
 exit
ipv6 router ospf 6
 router-id 2.2.2.6
 interface g0/0/0
 ipv6 ospf 6 area 0
 exit
interface g0/0/1
 ipv6 ospf 6 area 0
 exit
interface loopback 0
 ipv6 ospf 6 area 0
 ipv6 ospf network point-to-point
 exit
end
```

Router R3

```
hostname R3
no ip domain lookup
ipv6 unicast-routing
banner motd # R3, Using Ping, Traceroute, and Debug #
line con 0
 exec-timeout 0 0
 logging synchronous
 exit
```

```
line vty 0 4
 privilege level 15
 password cisco123
 exec-timeout 0 0
 logging synchronous
 login
 exit
interface g0/0/0
 ip address 10.2.3.3 255.255.255.0
 ipv6 address fe80::3:1 link-local
 ipv6 address 2001:db8:acad:1023::3/64
 no shutdown
 exit
interface s0/1/0
 ip address 10.1.3.3 255.255.255.128
 ipv6 address fe80::3:2 link-local
 ipv6 address 2001:db8:acad:1013::3/64
 no shutdown
 exit
interface s0/1/1
 ip address 10.1.3.130 255.255.255.128
 ipv6 address fe80::3:3 link-local
 ipv6 address 2001:db8:acad:1014::3/64
 no shutdown
 exit
interface loopback 0
 ip address 192.168.3.1 255.255.255.0
 ipv6 address fe80::3:4 link-local
 ipv6 address 2001:db8:acad:3000::1/64
 no shutdown
 exit
router ospf 4
 router-id 3.3.3.4
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.3.0 0.0.0.255 area 0
 exit
ipv6 router ospf 6
 router-id 3.3.3.6
interface g0/0/0
 ipv6 ospf 6 area 0
 exit
interface s0/1/0
 ipv6 ospf 6 area 0
 exit
interface s0/1/1
```

```
 ipv6 ospf 6 area 0
 exit
interface loopback 0
 ipv6 ospf 6 area 0
 ipv6 ospf network point-to-point
 exit
end
```

b. Set the clock on each router to UTC time.

c. Save the running configuration to startup-config.

Part 2: Explore Ping Options and Extended Ping Commands

a. The basic ping command supports several different Layer 3 protocols, as shown below. If none is specified, IOS XE defaults to IP (IPv4 or IPv6). Please note that the ping command has additional options that appear based on the current device configuration, such as **ping atm** and **ping vrf**.

```
R1# ping ?
  WORD      Ping destination address or hostname
  ancp      ANCP echo
  clns      CLNS echo
  ethernet  Ethernet echo
  ip        IP echo
  ipv6      IPv6 echo
  srb       srb echo
  tag       Tag encapsulated IP echo
  <cr>      <cr>
```

b. In our current network, the options available to R1 for a ping to 192.168.3.1 are listed below.

```
R1# ping 192.168.3.1 ?
  Extended-data  specify extended data pattern
  data           specify data pattern
  df-bit         enable do not fragment bit in IP header
  dscp           Specify DSCP value in ASCII/Numeric
  ingress        LAN source interface for Ingress
  repeat         specify repeat count
  size           specify datagram size
  source         specify source address or name
  timeout        specify timeout interval
  tos            specify type of service value
  validate       validate reply data
  <cr>           <cr>
```

The options listed allow you to craft the ping packet to meet test criteria for your situation:

- **data and extended-data** - By default, the payload of a ping is simply 0xABCD. The data (16 bits) and extended-data (32 bits) options allow you to specify something else. A potential use for this could be looking for payload corruption occurring between two endpoints.
- **df-bit** - Tells intermediate routers not to fragment the ping. This is commonly used with the size option to test a path for a specific MTU value.

- **dscp** - Sets a particular *Differentiated Services Code Point* value. The dscp value is used in Quality of Service, so this option allows you to probe basic QoS compliance through the network.
- **ingress** - Allows you to specify an ingress interface.
- **repeat** - By default, ping will send 5 packets. This option allows you to specify how many packets to send. This can be useful to test load balancing and other packet-stream situations.
- **size** - Allows you specify something other than the default 100-byte packet size. This is commonly used to path MTU validation. It could also be used in combination with the repeat option to keep the transmitting interface busy for a longer period of time.
- **source** - Allows you to specify a source address or interface. By default, the address of the outbound interface used to route to the destination will be used as the source address. Setting the source manually helps to verify full reachability.
- **timeout** - Sets the timeout value for each packet to something other than the default of 2 seconds. This option would most likely be used in very slow or very large networks.
- **tos** - Sets the *Type of Service* value to something other than the default of 0. This is similar to DSCP but for older methods of service assurance.
- **validate** - Enables the option to validate each reply.

c. There are several different response codes to a ping. Here are the most common:

- **!** - The exclamation point indicates receipt of a single reply. The ping successfully reached the destination, the destination replied, and the reply made it back to the originator. This is an ICMP Type 0 message.
- **.** - The period indicates that the ping timed out. Normally this indicates that the distant end received the ping, but could not reply for some reason.
- **U** - The destination of the ping was unreachable. Although not visible in the router output, there are 16 sub-codes to the unreachable message that describe exactly what was unreachable.
- **M** - An intermediate device could not fragment the packet, and so could not forward it.
- **?** - Unknown packet received.
- **&** - Packet lifetime exceeded.

d. To experiment with these options and see what codes are returned, issue the command **ping 192.168.3.1 source Loopback0**. The ping should be successful, indicated by 5 exclamation points.

```
R1# ping 192.168.3.1 source Loopback0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/3 ms
```

e. Issue the command **ping 172.16.3.1 source Loopback0**. The ping should fail with R2 sending back Destination Unreachable messages. If you examine routing table on R2, you can see why the ping failed.

```
R1# ping 172.16.3.1 source Loopback0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.3.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
U.U.U
Success rate is 0 percent (0/5)
```

f. The common MTU size for ethernet links is 1500 bytes. Issue the command **ping 192.168.3.1 source Loopback0 size 1750 df-bit** to see that the ping to 192.168.3.1 no longer works because the intermediate device could not fragment it.

```
R1# ping 192.168.3.1 source Loopback0 size 1750 df-bit
Type escape sequence to abort.
Sending 5, 1750-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
Packet sent with the DF bit set
.....
Success rate is 0 percent (0/5)
```

g. The command line version of ping and its options is very useful. Another version of ping you can use is the extended ping. The extended ping starts with you typing ping at a privileged exec prompt and pressing enter. From there, you will be taken through a menu to allow you to configure your ping. In addition to the options we have already discussed, there are IP header options and the capability to sweep a range of packet sizes.

The IP header options are Loose, Strict, Record, Timestamp and Verbose. Verbose is automatically included with any of the other options:

- **Loose** - Short for Loose Source Route, this option allows you (the source) to influence the path the ping takes by specifying the address(es) of the hop(s) you want the packet to go through. These are suggestions to the router, so if there is a better path, the router may use it.
- **Strict** - Short for Strict Source Route, this option allows you to mandate the exact path you want the ping to take. Intermediate devices must follow the next hop(s) specified until the destination is reached or there are no more next-hop addresses specified.
- **Record** - This option causes the address of up to nine intermediary hops to be recorded and returned to the originating host. This option records not only the hops along the way to the destination, but the hops in the return path as well.
- **Timestamp** - Used to measure roundtrip time to particular hosts.

As an example of using the IP options, issue a single ping to 192.168.3.1, and include the Record option to see what the results are.

```
R1# ping
Protocol [ip]:
Target IP address: 192.168.3.1
Repeat count [5]: 1
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Ingress ping [n]:
Source address or interface: Loopback0
DSCP Value [0]:
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0x0000ABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: Record
Number of hops [ 9 ]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
```

```
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
Packet has IP options:  Total option bytes= 39, padded length=40
 Record route: <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)

Reply to request 0 (1 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
   (10.1.2.1)
   (10.2.3.2)
   (192.168.3.1)
   (10.1.2.2) <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
 End of list

Success rate is 100 percent (1/1), round-trip min/avg/max = 1/1/1 ms
```

h. Use the extended ping to do a ping size sweep to determine the MTU of the path between R1 and R3.

```
R1# ping
Protocol [ip]:
Target IP address: 192.168.3.1
Repeat count [5]: 1
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Ingress ping [n]:
Source address or interface: Loopback0
DSCP Value [0]:
Type of service [0]:
Set DF bit in IP header? [no]: y
Validate reply data? [no]:
Data pattern [0x0000ABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
```

```
Sweep range of sizes [n]: y
Sweep min size [36]: 1475
Sweep max size [18024]: 1515
Sweep interval [1]:
Type escape sequence to abort.
Sending 41, [1475..1515]-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
Packet sent with the DF bit set
!!!!!!!!!!!!!!!!!!!!!!!!!.............
Success rate is 63 percent (26/41), round-trip min/avg/max = 1/1/2 ms
```

As you can see in the output, the pings were working until the 26th ping was sent with a payload of 1501 bytes.

Part 3: Explore Traceroute Options and Extended Traceroute Commands

Step 1: Explore traceroute options.

The traceroute command is used to discover the routes that packets actually take when traveling to their destination. The device (for example, a router or a PC) sends out a sequence of User Datagram Protocol (UDP) datagrams to an invalid port address at the remote host.

Three datagrams are sent, each with a Time-To-Live (TTL) field value set to one. The TTL value of 1 causes the datagram to "timeout" as soon as it hits the first router in the path; this router then responds with an ICMP Time Exceeded Message (TEM) indicating that the datagram has expired.

Another three UDP messages are now sent, each with the TTL value set to 2, which causes the second router to return ICMP TEMs. This process continues until the packets actually reach the other destination. Because these datagrams are trying to access an invalid port at the destination host, ICMP Port Unreachable Messages are returned, indicating an unreachable port. This event signals the traceroute program that it is finished.

The purpose behind this is to record the source of each ICMP Time Exceeded Message to provide a trace of the path the packet took to reach the destination.

Because a person could glean much valuable information from traceroute, it is often blocked at network boundaries.

Traceroute returns different information based on the status of the probes that have been sent. The table below shows the different codes.

Character	Description
nn msec	For each node, the round-trip time in milliseconds for the specified number of probes
*	The probe timed out
A	Administratively prohibited (example, access-list)
Q	Source quench (destination too busy)
I	User interrupted test
U	Port unreachable
H	Host unreachable
N	Network unreachable

Character	Description
P	Protocol Unreachable
T	Timeout
?	Unknown packet type

a. Like ping, traceroute supports several different Layer 3 protocols, but defaults to supporting IPv4.

```
R1# traceroute ?
  WORD       Trace route to destination address or hostname
  aaa        Define trace options for AAA events/actions/errors
  appletalk  AppleTalk Trace
  clns       ISO CLNS Trace
  ethernet   Ethernet Traceroute
  ip         IP Trace
  ipv6       IPv6 Trace
  ipx        IPX Trace
  <cr>       <cr>
```

b. After you have specified a target, other options are available.

```
R1# traceroute 192.168.3.1 ?
  dscp     Specify DSCP value in ASCII/Numeric
  ingress  LAN source interface for Ingress
  numeric  display numeric address
  port     specify port number
  probe    specify number of probes per hop
  source   specify source address or name
  timeout  specify time out
  ttl      specify minimum and maximum ttl
  <cr>     <cr>
```

The options for the command line version of traceroute include the following:

- **Numeric** - Displays output only in numeric form instead of trying to resolve by DNS as well.
- **Port** - Allows you to set a different starting port number, which defaults to 33433.
- **Probe** - Allows you to change the default 3 probes per hop.
- **Source** - Allows you to specify a different source address for the probe.
- **Timeout** - Allows you to specify a timeout other than the default of 3 seconds.
- **ttl** - Allows you to specify a minimum and maximum ttl other than the default 1 and 30.

c. On R1, issue the command **traceroute 192.168.3.1** to see what the output is when all the options are left to their default settings.

```
R1# traceroute 192.168.3.1
Type escape sequence to abort.
Tracing the route to 192.168.3.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.1.2.2 2 msec 1 msec 1 msec
  2 10.2.3.3 1 msec *  1 msec
```

The output tells you that 192.168.3.1 is reachable in two hops, with the next hop being 10.1.2.2. Also notice that one of the three probes sent to 10.2.3.3 timed out.

d. To experiment with these options and see what codes are returned, on R1 issue the command **traceroute 192.168.3.1 source Loopback0 ttl 2 10**.

```
R1# traceroute 192.168.3.1 source Loopback0 ttl 2 10
Type escape sequence to abort.
Tracing the route to 192.168.3.1
VRF info: (vrf in name/id, vrf out name/id)
  2 10.2.3.3 1 msec *  1 msec
```

Notice in the output that the only hop that replied was 10.2.3.3. In the previous traceroute with the default settings, the first hop that replied was 10.1.2.2. In this case, that hop is there, but the probe started out with a TTL of two, so it was not reported.

e. Now try a traceroute to a non-existent destination. R1 has a static route to 172.16.3.1 via R2, but that network doesn't exist. A traceroute to 172.16.3.1 on R1 will yield something similar to the following:

```
R1# traceroute 172.16.3.1
Type escape sequence to abort.
Tracing the route to 172.16.3.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.1.2.2 1 msec 1 msec 1 msec
  2 10.1.2.2 !H  *  !H
```

The H in the output indicates that the host is unreachable from IP address 10.1.2.2.

f. Traceroute also has an extended option. Much like the extended ping, it starts out with just the command **traceroute** and guides the user through all the different options using a menu.

```
R1# traceroute
Protocol [ip]:
Target IP address: 192.168.3.1
Ingress traceroute [n]:
Source address or interface: Loopback0
DSCP Value [0]:
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 192.168.3.1
VRF info: (vrf in name/id, vrf out name/id)
  1 10.1.2.2 1 msec 1 msec 1 msec
  2 10.2.3.3 1 msec *  1 msec
```

Part 4: Explore Common Debug Commands and Conditional Debugging

The debug utility is a powerful tool that must be used carefully. Debug receives top priority from the CPU and as such, the output and operation of a debug command could be so busy that it keeps the CPU from

responding to any other commands, including the command to stop the debug. Debug can make your system unresponsive.

There are many debug options available, and you should be as deliberate and as specific as possible when giving a debug command, especially on a production system. It is possible to further limit debug output using access lists, as well as conditional debugging.

Step 1: Use common debug commands.

a. Debug can be used to get a deeper look at how things are working on your router. For example, when you issue the command **debug ipv6 nd** on R1, you get information about neighbor discovery events related to R1. In the output below, R1 received an RA on g0/0/0 from R2.

```
R1# debug ipv6 nd
  ICMP Neighbor Discovery events debugging is on
  ICMP ND HA events debugging is ON
R1#
*Jan 27 10:45:54.944: ICMPv6-ND: (GigabitEthernet0/0/0,FE80::2:1) Received RA
*Jan 27 10:45:54.944: ICMPv6-ND: Validating ND packet options: valid
*Jan 27 10:45:54.944: ICMPv6-ND: Prefix : 2001:DB8:ACAD:1012::, Length: 64,
VldLifetime: 2592000, Prf Lifetime: 604800, PI Flags: C0
```

b. To see what debugs are running on a device, issue the command **show debug**. To disable a debug, use the **no** form of the command that started the debug, or issue the command **undebug all**.

```
R1# show debug
IOSXE Conditional Debug Configs:

Conditional Debug Global State: Stop

IOSXE Packet Tracing Configs:

Packet Infra debugs:

Ip Address                                              Port
------------------------------------------------------|----------

  ICMP Neighbor Discovery events debugging is on
  ICMP ND HA events debugging is ON

R1# undebug all
All possible debugging has been turned off
```

c. Generally, you use a debug command to try and discover why something is not working correctly. For example, the pings we did earlier from R1 to 172.16.3.1 were timing out. There was no indication of what was happening, and you could not actually see the ICMP return messages. But you can with debug. On R1, issue the command **debug ip icmp** and then **ping 172.16.3.1 source loopback0 repeat 1**. The output shows you that the ping failed (destination unreachable) at R2.

```
R1# debug ip icmp
ICMP packet debugging is on
R1# ping 172.16.3.1 source loopback0 repeat 1
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 172.16.3.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
```

```
U
Success rate is 0 percent (0/1)
R1#
*Jan 27 10:50:25.308: ICMP: dst (192.168.1.1) host unreachable rcv from 10.1.2.2
R1# undebug all
```

Step 2: Use conditional debug commands.

a. As mentioned previously, you can use an access list to limit the output of a debug command. Issue the command **debug ip packet** to see what kind of information is displayed. **Note**: This is being done on a small topology with no users. Extreme care should be taken with this command on a production network.

b. After you have collected some output, issue the **undebug all** command.

```
R1# debug ip packet
IP packet debugging is on
R1#
*Jan 27 11:23:59.200: IP: s=10.1.3.1 (local), d=224.0.0.5 (Serial0/1/0), len
100, sending broad/multicast
R1#
*Jan 27 11:24:04.110: IP: s=10.1.3.129 (local), d=224.0.0.5 (Serial0/1/1),
len 100, sending broad/multicast
R1#
*Jan 27 11:24:06.185: IP: s=10.1.2.1 (local), d=224.0.0.5
(GigabitEthernet0/0/0), len 100, sending broad/multicast
R1# unde
*Jan 27 11:24:09.137: IP: s=10.1.3.1 (local), d=224.0.0.5 (Serial0/1/0), len
100, sending broad/multicast
R1# undebug all
All possible debugging has been turned off
```

c. Create standard access-list number 1 on R1 that specifies the source address 10.1.3.1. Then, issue the command **debug ip packet 1**.

```
R1# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)# access-list 1 permit host 10.1.3.1
R1(config)# exit
R1#
*Jan 27 11:25:27.974: %SYS-5-CONFIG_I: Configured from console by console
R1#
R1# debug ip packet 1
IP packet debugging is on for access list 1
R1#
*Jan 27 11:25:45.077: IP: s=10.1.3.1 (local), d=224.0.0.5 (Serial0/1/0), len
100, sending broad/multicast
R1#
*Jan 27 11:25:55.049: IP: s=10.1.3.1 (local), d=224.0.0.5 (Serial0/1/0), len
100, sending broad/multicast
R1# undebug all
All possible debugging has been turned off
```

```
R1#
```

Notice that the debug output now only shows packets sourced from 10.1.3.1.

d. Another method of performing conditional debugging is to debug based on a specific interface. The idea is the same; we want to limit the output of a debug command. For this example, we will try to the get the same output as we received debugging with an ACL in step b. First, issue the command **debug interface Serial0/1/0**, then issue the command **debug ip packet**.

```
R1# debug interface Serial0/1/0
Condition 1 set
R1# debug ip packet
IP packet debugging is on

*Jan 27 12:25:30.396: IP: s=10.1.3.1 (local), d=224.0.0.5 (Serial0/1/0), len 100,
sending broad/multicastOutput intf Se0/1/0 matches debug filer

*Jan 27 12:25:39.553: IP: s=10.1.3.1 (local), d=224.0.0.5 (Serial0/1/0), len 100,
sending broad/multicastOutput intf Se0/1/0 matches debug filer

*Jan 27 12:25:49.446: IP: s=10.1.3.1 (local), d=224.0.0.5 (Serial0/1/0), len 100,
sending broad/multicastOutput intf Se0/1/0 matches debug filer

R1# undebug all
1 conditions have been removed
All possible debugging has been turned off
```

Note that the **undebug all** command also removed the debug condition. If the condition was not removed by this command, you would need to issue the command **no debug condition 1** or **no debug condition all**.

Part 5: Troubleshoot OSPF with Debugging

The basic process for establishing an OSPF adjacency and exchanging routing information is well known. A router sends OSPF hello packets out of an interface. The OSPF hello packets contain certain parameters that a potential neighbor must match. When the exchange of hello packets is complete an adjacency is formed, and routing information is then shared. But what if the adjacency is not being formed? The normal syslog output does not provide the details you may need to successfully fix the issue. But with the correct debug command, you can find the misconfigurations and correct them.

Instructor Note: For this part of the lab, you need to have a way to inject errors into the OSPF configuration of R2 or R3 so that adjacency does not occur. If your students are working in pairs, you might have one student purposely change the configuration so OSPF does not work, and have their partner troubleshoot the error. Or you yourself can manually add errors. The errors that will be used for demonstration purposes here are mismatched timer values, mismatched network type, and mismatched MTU.

Router R2

```
en
conf t
interface g0/0/0
 ip ospf network point-to-multipoint
 exit
end
```

Router R3

```
en
conf t
interface s0/1/0
 ip mtu 768
 exit
interface s0/1/1
 ip ospf hello-interval 120
 exit
end
```

a. Errors have been injected into the network effecting OSPF adjacencies at R1. Issue the command **show ip ospf neighbor** on R1 to see the status of neighbors.

```
R1# sho ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
3.3.3.4           0   INIT/  -        00:00:36    10.1.3.3        Serial0/1/0
```

As you can see, we are only receiving hellos from one neighbor, when there should be three based on the topology diagram. The one neighbor we have received hellos from is in the INIT state.

Note: You may need to clear the ospf process on all devices.

b. On R1, issue the command **debug ip ospf hello** and examine the output.

```
R1# debug ip ospf hello
OSPF hello debugging is on
R1#
R1#
*Jan 27 13:54:22.107: OSPF-4 HELLO Se0/1/0: Rcv hello from 3.3.3.4 area 0 10.1.3.3
*Jan 27 13:54:22.421: OSPF-4 HELLO Se0/1/1: Rcv hello from 3.3.3.4 area 0 10.1.3.130
*Jan 27 13:54:22.422: OSPF-4 HELLO Se0/1/1: Mismatched hello parameters from
10.1.3.130
*Jan 27 13:54:22.422: OSPF-4 HELLO Se0/1/1: Dead R 48 C 40, Hello R 12 C 10
R1#
*Jan 27 13:54:31.770: OSPF-4 HELLO Se0/1/0: Rcv hello from 3.3.3.4 area 0 10.1.3.3
*Jan 27 13:54:38.072: OSPF-4 HELLO Gi0/0/0: Rcv hello from 2.2.2.4 area 0 10.1.2.2
*Jan 27 13:54:38.072: OSPF-4 HELLO Gi0/0/0: Mismatched hello parameters from 10.1.2.2
*Jan 27 13:54:38.072: OSPF-4 HELLO Gi0/0/0: Dead R 120 C 40, Hello R 30 C 10 Mask R
255.255.255.0 C 255.255.255.0
```

The debug output shows that there are mismatched timers on the two interfaces where neighbors should be. Taking a closer look at the timers, we can see that the neighbor on S0/1/1 has the hello interval set to 12 seconds while the locally configured timer is 10 seconds. Verify this in the output of **show ip ospf interface s0/1/1 | include Timer** on R1. Notify the administrator for the neighbor router to determine why the value was changed and what the correct value should be.

```
R1# show ip ospf interface s0/1/1 | include Timer
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
```

Continuing our examination, we can see that the neighbor on g0/0/0 is also sending different timers in their hello. What reason might there be for this? The output says that the hello interval is set to 40 seconds and the dead interval set to 120 seconds. The output also includes two different subnet mask values. The timer values and mask information indicate that the remote router interface is configured as something other than a point-to-point interface. A check with the administrator at R2 finds that the interface is configured for point-to-multipoint.

The **debug ip ospf hello** output showed hellos being received on both serial interfaces, but the connection via S0/1/0 is in INIT state. Examine the output of **debug ip ospf adj** to see if there are clues there.

```
R1# debug ip ospf adj
OSPF adjacency debugging is on
R1#
*Jan 27 14:18:19.192: OSPF-4 ADJ   Se0/1/0: Rcv DBD from 3.3.3.4 seq 0x153A opt0x52
flag 0x7 len 32  mtu 768 state INIT
*Jan 27 14:18:19.192: OSPF-4 ADJ   Se0/1/0: Nbr 3.3.3.4 has smaller interface MTU
*Jan 27 14:18:19.193: OSPF-4 ADJ   Se0/1/0: 2 Way Communication to 3.3.3.4, state 2WAY
*Jan 27 14:18:19.193: OSPF-4 ADJ   Se0/1/0: Nbr 3.3.3.4: Prepare dbase exchange
*Jan 27 14:18:19.193: OSPF-4 ADJ   Se0/1/0: Send DBD to 3.3.3.4 seq 0x22DB opt 0x52
flag 0x7 len 32
*Jan 27 14:18:19.193: OSPF-4 ADJ   Se0/1/0: NBR Negotiation Done. We are the SLAVE
*Jan 27 14:18:19.193: OSPF-4 ADJ   Se0/1/0: Nbr 3.3.3.4: Summary list built, size 3
R1#
*Jan 27 14:18:19.193: OSPF-4 ADJ   Se0/1/0: Send DBD to 3.3.3.4 seq 0x153A opt 0x52
flag 0x2 len 92
R1#
*Jan 27 14:18:23.860: OSPF-4 ADJ   Se0/1/0: Rcv DBD from 3.3.3.4 seq 0x153A opt0x52
flag 0x7 len 32  mtu 768 state EXCHANGE
*Jan 27 14:18:23.860: OSPF-4 ADJ   Se0/1/0: Nbr 3.3.3.4 has smaller interface MTU
```

The debug output indicates that R3 neighbor 3.3.3.4 has a smaller interface MTU than R1 has on interface S0/1/0. Notify the administrator for the neighbor router to determine why the value was changed and what the correct value should be.

c. With all of the errors fixed, the resulting neighbor table on R1 should look like this:

```
R1# show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
3.3.3.4           0   FULL/  -        00:00:35    10.1.3.130      Serial0/1/1
3.3.3.4           0   FULL/  -        00:00:35    10.1.3.3        Serial0/1/0
2.2.2.4           1   FULL/BDR        00:00:35    10.1.2.2        GigabitEthernet0/0/0
```

Router Interface Summary Table

Router Model	Ethernet Interface #1	Ethernet Interface #2	Serial Interface #1	Serial Interface #2
1800	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
1900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2801	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
2811	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)

Router Model	Ethernet Interface #1	Ethernet Interface #2	Serial Interface #1	Serial Interface #2
4221	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
4300	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)

Note: To find out how the router is configured, look at the interfaces to identify the type of router and how many interfaces the router has. There is no way to effectively list all the combinations of configurations for each router class. This table includes identifiers for the possible combinations of Ethernet and Serial interfaces in the device. The table does not include any other type of interface, even though a specific router may contain one. An example of this might be an ISDN BRI interface. The string in parenthesis is the legal abbreviation that can be used in Cisco IOS commands to represent the interface.

Device Configs – Final

Router R1

```
R1# show run
Building configuration...


Current configuration : 2052 bytes
!
version 16.9
service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
!
hostname R1
!
boot-start-marker
boot-end-marker
!
no aaa new-model
!
no ip domain lookup
!
login on-success log
!
subscriber templating
!
ipv6 unicast-routing
multilink bundle-name authenticated
!
spanning-tree extend system-id
!
redundancy
 mode none
```

```
!
interface Loopback0
 ip address 192.168.1.1 255.255.255.224
 ipv6 address FE80::1:4 link-local
 ipv6 address 2001:DB8:ACAD:1000::1/64
 ipv6 ospf 6 area 0
 ipv6 ospf network point-to-point
!
interface GigabitEthernet0/0/0
 ip address 10.1.2.1 255.255.255.0
 negotiation auto
 ipv6 address FE80::1:1 link-local
 ipv6 address 2001:DB8:ACAD:1012::1/64
 ipv6 ospf 6 area 0
!
interface GigabitEthernet0/0/1
 no ip address
 negotiation auto
!
interface Serial0/1/0
 ip address 10.1.3.1 255.255.255.128
 ipv6 address FE80::1:2 link-local
 ipv6 address 2001:DB8:ACAD:1013::1/64
 ipv6 ospf 6 area 0
!
interface Serial0/1/1
 ip address 10.1.3.129 255.255.255.128
 ipv6 address FE80::1:3 link-local
 ipv6 address 2001:DB8:ACAD:1014::1/64
 ipv6 ospf 6 area 0
!
router ospf 4
 router-id 1.1.1.4
 network 10.0.0.0 0.255.255.255 area 0
 network 192.0.0.0 0.255.255.255 area 0
!
ip forward-protocol nd
no ip http server
ip http secure-server
ip route 172.16.3.0 255.255.255.0 GigabitEthernet0/0/0 10.1.2.2
!
access-list 1 permit 10.1.3.1
ipv6 router ospf 6
 router-id 1.1.1.6
!
control-plane
!
banner motd ^C R1, Using Ping, Traceroute, and Debug  ^C
!
line con 0
```

```
 exec-timeout 0 0
 logging synchronous
 transport input none
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 exec-timeout 0 0
 privilege level 15
 password cisco123
 logging synchronous
 login
!
end
```

Router R2

```
R2# show run
Building configuration...


Current configuration : 1760 bytes
!
version 16.9
service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
!
hostname R2
!
boot-start-marker
boot-end-marker
!
no aaa new-model
!
no ip domain lookup
!
login on-success log
!
subscriber templating
!
ipv6 unicast-routing
multilink bundle-name authenticated
!
spanning-tree extend system-id
!
redundancy
 mode none
!
```

```
interface Loopback0
 ip address 192.168.2.1 255.255.255.224
 ipv6 address FE80::2:3 link-local
 ipv6 address 2001:DB8:ACAD:2000::1/64
 ipv6 ospf 6 area 0
 ipv6 ospf network point-to-point
!
interface GigabitEthernet0/0/0
 ip address 10.1.2.2 255.255.255.0
 negotiation auto
 ipv6 address FE80::2:1 link-local
 ipv6 address 2001:DB8:ACAD:1012::2/64
 ipv6 ospf 6 area 0
!
interface GigabitEthernet0/0/1
 ip address 10.2.3.2 255.255.255.0
 negotiation auto
 ipv6 address FE80::2:2 link-local
 ipv6 address 2001:DB8:ACAD:1023::2/64
 ipv6 ospf 6 area 0
!
router ospf 4
 router-id 2.2.2.4
 network 10.0.0.0 0.255.255.255 area 0
 network 192.0.0.0 0.255.255.255 area 0
!
ip forward-protocol nd
no ip http server
ip http secure-server
!
ipv6 router ospf 6
 router-id 2.2.2.6
!
control-plane
!
banner motd ^C R2, Using Ping, Traceroute, and Debug  ^C
!
line con 0
 exec-timeout 0 0
 logging synchronous
 transport input none
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 exec-timeout 0 0
 privilege level 15
 password cisco123
 logging synchronous
 login
```

```
!
end
```

Router R3

```
R3# show run
Building configuration...


Current configuration : 1956 bytes
!
version 16.9
service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
!
hostname R3
!
boot-start-marker
boot-end-marker
!
no aaa new-model
!
no ip domain lookup
!
login on-success log
!
subscriber templating
!
ipv6 unicast-routing
multilink bundle-name authenticated
!
spanning-tree extend system-id
!
redundancy
 mode none
!
interface Loopback0
 ip address 192.168.3.1 255.255.255.224
 ipv6 address FE80::3:4 link-local
 ipv6 address 2001:DB8:ACAD:3000::1/64
 ipv6 ospf 6 area 0
 ipv6 ospf network point-to-point
!
interface GigabitEthernet0/0/0
 ip address 10.2.3.3 255.255.255.0
 negotiation auto
 ipv6 address FE80::3:1 link-local
 ipv6 address 2001:DB8:ACAD:1023::3/64
```

```
 ipv6 ospf 6 area 0
!
interface GigabitEthernet0/0/1
 no ip address
 negotiation auto
!
interface Serial0/1/0
 ip address 10.1.3.3 255.255.255.128
 ipv6 address FE80::3:2 link-local
 ipv6 address 2001:DB8:ACAD:1013::3/64
 ipv6 ospf 6 area 0
!
interface Serial0/1/1
 ip address 10.1.3.130 255.255.255.128
 ipv6 address FE80::3:3 link-local
 ipv6 address 2001:DB8:ACAD:1014::3/64
 ipv6 ospf 6 area 0
!
router ospf 4
 router-id 3.3.3.4
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.3.0 0.0.0.255 area 0
!
ip forward-protocol nd
no ip http server
ip http secure-server
!
ipv6 router ospf 6
 router-id 3.3.3.6
!
control-plane
!
banner motd ^C R3, Using Ping, Traceroute, and Debug  ^C
!
line con 0
 exec-timeout 0 0
 logging synchronous
 transport input none
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 exec-timeout 0 0
 privilege level 15
 password cisco123
 logging synchronous
 login
!
end
```