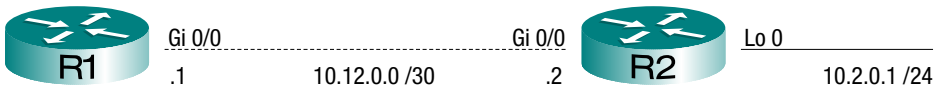
Lab 8a. OSPF

ccnacookbook.com

GOAL

Set up OSPF in a single area so that all subnets are visible from all routers.



Topology—You'll need

- 2 routers of almost any vintage (A 2621xm is cheap. I'm using a 2821 and a 2851. Interface names may vary between FastEthernet and GigabitEthernet, depending on hardware choice)
- 1 Ethernet crossover cable to connect the two routers
- Some way to issue commands over the router console ports

CONFIGURATION STEPS

Wire the topology and give your routers their basic configurations

- Hostname per the diagram
- Tell the router that when it doesn't recognize a command (for example a typo), it shouldn't attempt to contact a DNS server to prepare to telnet to that hostname
- Tell the console port not to log you out after a period of inactivity
- Tell the router that if it sends syslog messages to the console port while you're typing there, it should reprint the prompt and whatever you had already typed so that you know where you were

Configure interface IP addresses per the diagram.

Start an OSPF process on each router and add all subnets on all routers to the same OSPF area.

VERIFICATION

What routes has OSPF added to R1's routing table? _____

_____

From R1, can you ping 10.2.0.1? _____

What's the OSPF router ID of R1? Why? _____

_____

What's the OSPF router ID of R2? Why? _____

_____

Which OSPF process number did you use on R1? On R2? Can they differ? Be the same?

Which OSPF area did you put the interfaces in? Why?

CONFIGURATION WALKTHROUGH

Give the routers their basic configurations, then their interface IP addresses.

R1	R2

`Router(config)# hostname R1`
`R1(config)# no ip domain lookup`
`R1(config)# line console 0`
`R1(config-line)# exec-timeout 0 0`
or "no exec-timeout"
`R1(config-line)# logging synchronous`
	`hostname R6`
	`no ip domain lookup`
	`line con 0`
	` exec-timeout 0 0`
	` logging synchronous`

`R1(config)# interface gi0/0`
`R1(config-if)# ip address 10.12.0.1 255.255.255.252`
`R1(config-if)# no shut`
	`interface Loopback0`
	` ip address 10.2.0.1 255.255.255.0`
	`!`
	`interface GigabitEthernet0/0`
	` ip address 10.12.0.2 255.255.255.252`

Start an OSPF process on each router and add all subnets on all routers to the same OSPF area.

	R1
1	`R1(config)# router ospf 10`
2	`R1(config-router)# network 10.12.0.0 0.0.0.3 area 0`

It's fun when two lines of code do as much and can teach as much as these two. First, line 1

- This line starts the OSPF process running on R1
- 10 is the OSPF process number on R1. It can be any number between 1 and 65,535. It only has meaning on this one router, so it doesn't need to match on R1.

Next, Line 2

- This line adds any interface with a matching network to the OSPF process
- 0.0.0.3 is the wildcard for the network 10.12.0.0. and corresponds to the netmask 255.255.255.252
- In general, a wildcard mask can be obtained from a netmask by subtracting the netmask from 255.255.255.255
- In this case, interface Gigabit 0/0 matches and is added to the OSPF process, meaning that OSPF will discover neighbors out that interface and will advertise that interface's network via OSPF
- Notice that we're involving interfaces in the OSPF process from the "config-router" prompt based on a match of their IP addresses, not fromthe config of the interfaces themselves. That will change when we use OSPFv3 on IPv6

Now some parallel config on R2

R2

```
1 R2(config)# router ospf 20
2 R2(config-router)# network 10.0.0.0 0.255.255.255 area 0
3
4 *Dec 27 23:42:51.867: %OSPF-5-ADJCHG: Process 20, Nbr 10.12.0.1 on
5 GigabitEthernet0/0 from LOADING to FULL, Loading Done
```

Here, we used the power of the wildcard to add all interfaces with an IP address beginning with "10." to the OSPF process. Literally, a wildcard bit of 0 means that the corresponding address bit must match, while a 1 means "we don't care." So, with an address of 10.0.0.0 and wildcard 0.255.255.255, the first 8 bits must match our given "10" and the last 24 bits don't matter. That drags in both interface Gi0/0, with an address of 10.12.0.2, and our loopback interface, with its address of 10.2.0.1.

Lines 4 and 5 show the syslog message as the OSPF neighbor relationship across Gi0/0 to R1 becomes fully adjacent (converged). Process 20 is our own OSPF process number. 10.12.0.1 is the router ID of R1, located out our interface GigabitEthernet0/0.

VERIFICATION WALKTHROUGH

First, let's see what routes are in R1's routing table

R1

```
1  R1# show ip route
2  Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
3         D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
4         N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
5         E1 - OSPF external type 1, E2 - OSPF external type 2
6         i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
7         ia - IS-IS inter area, * - candidate default, U - per-user static route
8         o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
9         + - replicated route, % - next hop override
10
11 Gateway of last resort is not set
12
13       10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
14 O        10.2.0.1/32 [110/2] via 10.12.0.2, 00:25:04, GigabitEthernet0/0
15 C        10.12.0.0/30 is directly connected, GigabitEthernet0/0
16 L        10.12.0.1/32 is directly connected, GigabitEthernet0/0
```

- Lines 2 through 9 are the legend that we can use to tell how individual routes became part of the routing table. Enjoy it; I'll probably omit it in future examples
- Line 16 shows the IP address the we locally ("L" in the left column) added to interface Gi0/0
- Line 15 shows the directly connected route ("C") that was derived from the IP address and netmask on that interface
- R2 also advertised that route to us via OSPF, but the Administrative Distance (AD) of OSPF is 110, while the AD of a directly connected route is 0, so the routing table uses the directly connected route (lower AD wins). The AD difference reflects the reality that a route based on our own interface's IP address is considerably more likely to be correct than a route that some other router told us about
- Within the lowest AD, the routing table will contain the route with the lowest cost.

- Line 14 shows the route to R2's loopback interface, which we learned about through OSPF
- Line 13 is a throwback to the days of classful subnetting which serves as a header, organizing subnets of the same classful network together.

Looking specifically at our OSPF-supplied route,

R1

```
O        10.2.0.1/32 [110/2] via 10.12.0.2, 00:25:04, GigabitEthernet0/0
```

We can tell several things.

- We know that OSPF supplied the route in two ways. First, there's a 'O' in the left colum. Second, the administrative distance of the route is shown as the first number in the square brackets. We know that OSPF has an AD of 110
- We know that the route has an OSPF cost of 2 (the second number in the square brackets). If OSPF had learned more than one route to that subnet, the route(s) with the lowest cost *within* OSPF would have been offered to the routing table to be included or ignored based on whether a source with a lower AD had also offered a route to the same subnet
- We know that OSPF last updated the route information 25 minutes and 4 seconds ago. The periodic reflooding of routes by OSPF (every 30 minutes) won't reset this clock. Repetition is not an update
- As with any route, we can tell that the exit interface is Gi0/0 and the next hop address is 10.12.0.2
- Don't worry about the subnet having a /32 mask, matching only a single address. That's the way loopback interfaces are treated. Being more realistic with a "real" subnet would have required a third router.

If we forget what any of this means, we can pull up a more verbose entry with better labels.

R1

```
1 R1# show ip route 10.2.0.1
2 Routing entry for 10.2.0.1/32
3   Known via "ospf 10", distance 110, metric 2, type intra area
4   Last update from 10.12.0.2 on GigabitEthernet0/0, 00:26:34 ago
5   Routing Descriptor Blocks:
6   * 10.12.0.2, from 10.2.0.1, 00:26:34 ago, via GigabitEthernet0/0
7       Route metric is 2, traffic share count is 1
```

Now, let's look at OSPF, itself.

R1

```
1 R1# show ip ospf
2  Routing Process "ospf 10" with ID 10.12.0.1
3                      39 additional lines of output omitted
```

R2

```
4 R2# show ip ospf
5  Routing Process "ospf 20" with ID 10.2.0.1
```

Looking at just the first line of output from this command, we can learn the Router ID (RID) of each router's OSPF process and the OSPF process number on each router. Notice that the process numbers don't need to match from one router to the next. In fact, it's doubtful that you would ever care what they are at the CCNA level.

R2 chose the RID 10.2.0.1 because

- You hadn't explicitly set the RID to something else
- 10.2.0.1 was the highest IP address of the router's loopback interfaces

R1 chose the RID 10.12.0.1 because

- You hadn't explicitly set the RID to something else
- R1 doesn't have any loopback interfaces
- 10.12.0.1 was the highest IP address of the router's normal (and "up") interfaces

Of course, you can explicitly set the RID on either or both routers to make diagnostic command output easier to read and influence Designated Router (DR) elections on Ethernet subnets. The RID is just a 32-bit number, formatted as if it were an IP address. After you change the number, you'll have to reset the OSPF process to make the change take effect.

R1

```
R1(config)# router ospf 10
R1(config-router)# router-id 0.0.0.1
% OSPF: Reload or use "clear ip ospf process" command, for this to take effect
R1(config-router)# do clear ip ospf process
Reset ALL OSPF processes? [no]: y
R1(config-router)#
Jan  2 19:43:03.446: %OSPF-5-ADJCHG: Process 10, Nbr 10.2.0.1 on
GigabitEthernet0/0 from FULL to DOWN, Neighbor Down: Interface down or detached
Jan  2 19:43:03.450: %OSPF-5-ADJCHG: Process 10, Nbr 10.2.0.1 on
GigabitEthernet0/0 from LOADING to FULL, Loading Done
```

On to figuring out what OSPF area the interfaces are in…

R1

```
R1# show ip ospf database

            OSPF Router with ID (0.0.0.1) (Process ID 10)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
0.0.0.1         0.0.0.1         1478        0x80000002 0x00A15F 1
10.2.0.1        10.2.0.1        1100        0x80000012 0x00AE0B 2

                Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.12.0.2       10.2.0.1        1623        0x80000002 0x0062A6
```

Clearly, everything is in area 0 (the backbone). This command also tells us our router ID (now updated to 0.0.0.1) and the OSPF process number.