Computer Network Architecture

Lab 3: Dynamic Routing - Link-State with OSPF

EL HAKOUNI Yassin
23/05/2025

1. Introduction

1.1 Context

OSPF is one of the most widely used routing protocols in enterprise networks and datacenter infrastructures. Its ability to support hierarchical topologies (via OSPF areas), rapid convergence, and route optimization using ECMP (Equal-Cost Multi-Path) makes it indispensable in professional environments.

This lab aims to practically explore the mechanisms of OSPF on two types of topologies:

- A simple three-router topology to understand basic OSPF functionalities: neighbor relationships, Link-State Database (LSDB), metrics, and synchronization.
- A more advanced Clos Fat-Tree topology (k=4), frequently used in datacenters for scalability, redundancy, and efficiency. This infrastructure is divided into several OSPF areas (one backbone area and multiple POD areas), allowing experimentation with hierarchical and multi-path routing.

1.2 Objectives

- Understand OSPF's internal operations (LSA, DR/BDR election, Hello/Dead timers).
- Master OSPF configuration using FRRouting with vtysh in GNS3.
- Observe dynamic OSPF neighbor formation (Full adjacency state) and LSDB creation.
- Capture and analyze OSPF Hello messages with Wireshark.
- Study ECMP (Equal-Cost Multi-Path) routing in a Fat-Tree topology.
- Implement a Clos Fat-Tree topology with one backbone area and four POD areas.
- Apply /30 subnetting to all point-to-point links.
- Experiment with changing link costs (bandwidth) to observe impacts on routing tables.

1.3 Methodology and Environment

The simulation environment is GNS3 (Graphical Network Simulator) with Debian 12 routers running FRRouting (version 8.4.4). CLI access to routers is provided through Telnet or SSH (using vtysh), allowing commands similar to Cisco IOS.

The lab proceeds in several phases:

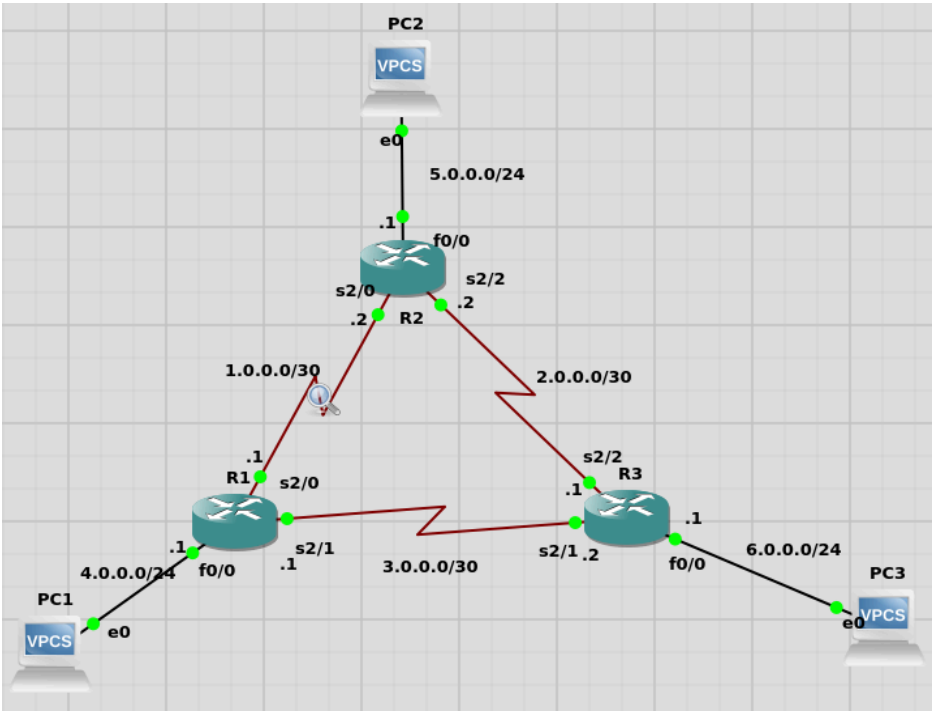
- Activating the OSPF process.

- Configuring interfaces and declaring networks within OSPF areas.
- Observing neighbor relationships and the Link-State Database.
- Capturing OSPF messages using Wireshark.
- Experimenting with link metrics (cost).
- Building a fully functional Fat-Tree network.

2. Basic OSPF Configuration

2.1 Used Topology

Initially, I worked on a simple topology consisting of three routers connected in a triangle: R1, R2, and R3. Each router is connected to the other two routers through point-to-point links.



Each link uses a /30 addressing scheme, as recommended for point-to-point connections. Here's an overview of the addressing scheme used :

Link	Router A	Interface	Router B	Interface	/30 Network
R1 ↔ R2	R1	eth0	R2	eth0	10.0.0.0/30
R2 ↔ R3	R2	eth1	R3	eth0	10.0.0.4/30

R3 ↔ R1	R3	eth1	R1	eth1	10.0.0.8/30

2.2 Enabling OSPF on Each Router

On each router, I activated the OSPF process and assigned a unique router ID, essential for identification in the Link-State Database.

Example on R1 :

```
configure terminal

router ospf

ospf router-id 1.1.1.1

exit
```

The same principle was applied to R2 (2.2.2.2) and R3 (3.3.3.3).

2.3 Interface Configuration

I assigned IP addresses to each interface, activated the interfaces, and specified each as a point-to-point link for OSPF.

Example configuration on R1 :

```
interface eth0

ip address 10.0.0.1/30

no shutdown

ip ospf network point-to-point

exit



interface eth1

ip address 10.0.0.9/30

no shutdown
```

```
ip ospf network point-to-point

exit
```

Finally, I associated interfaces with the OSPF process and specified networks within area 0 :

```
router ospf

network 10.0.0.0/30 area 0

network 10.0.0.8/30 area 0

exit
```

Similar configurations were applied to R2 and R3 using appropriate subnet ranges.

2.4 Neighbor Relationship Verification

After completing the configuration, I used the following command to check OSPF neighbor relationships :

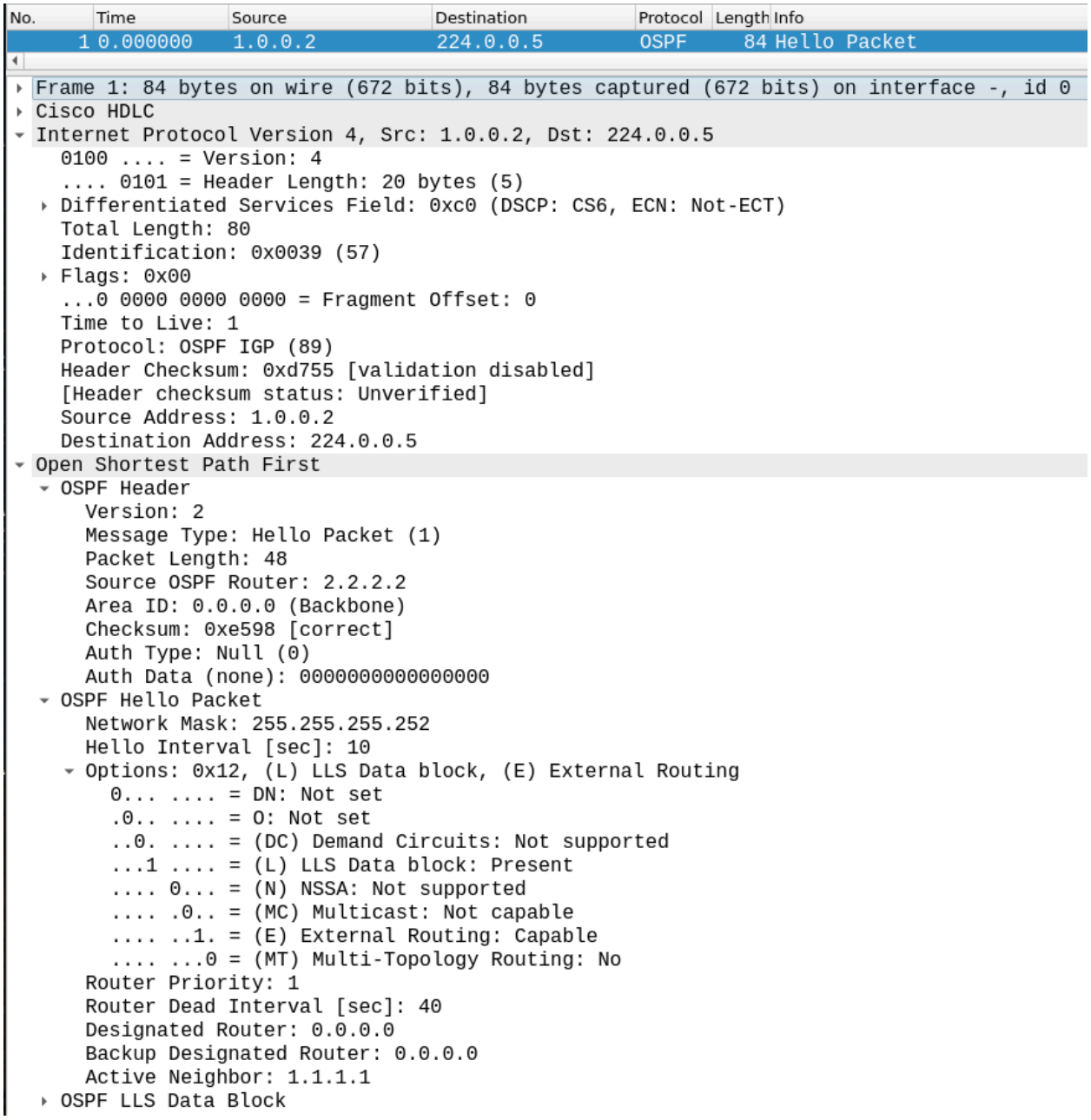
```
R1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
3.3.3.3           0   FULL/  -        00:00:32    3.0.0.2         Serial2/1
2.2.2.2           0   FULL/  -        00:00:39    1.0.0.2         Serial2/0
R1#
```

The "Full" state indicates successful adjacency establishment.

2.5 OSPF Hello Packet Analysis

I launched Wireshark on an interface and captured OSPF Hello packets.



```
No.  Time      Source    Destination  Protocol  Length  Info
  1  0.000000  1.0.0.2   224.0.0.5    OSPF        84    Hello Packet

Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface -, id 0
Cisco HDLC
Internet Protocol Version 4, Src: 1.0.0.2, Dst: 224.0.0.5
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 80
    Identification: 0x0039 (57)
  Flags: 0x00
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 1
    Protocol: OSPF IGP (89)
    Header Checksum: 0xd755 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 1.0.0.2
    Destination Address: 224.0.0.5
Open Shortest Path First
  OSPF Header
      Version: 2
      Message Type: Hello Packet (1)
      Packet Length: 48
      Source OSPF Router: 2.2.2.2
      Area ID: 0.0.0.0 (Backbone)
      Checksum: 0xe598 [correct]
      Auth Type: Null (0)
      Auth Data (none): 0000000000000000
  OSPF Hello Packet
      Network Mask: 255.255.255.252
      Hello Interval [sec]: 10
    Options: 0x12, (L) LLS Data block, (E) External Routing
        0... .... = DN: Not set
        .0.. .... = O: Not set
        ..0. .... = (DC) Demand Circuits: Not supported
        ...1 .... = (L) LLS Data block: Present
        .... 0... = (N) NSSA: Not supported
        .... .0.. = (MC) Multicast: Not capable
        .... ..1. = (E) External Routing: Capable
        .... ...0 = (MT) Multi-Topology Routing: No
      Router Priority: 1
      Router Dead Interval [sec]: 40
      Designated Router: 0.0.0.0
      Backup Designated Router: 0.0.0.0
      Active Neighbor: 1.1.1.1
  OSPF LLS Data Block
```

I identified key fields:

- Router ID
- Area ID
- Hello & Dead intervals
- Designated Router (DR) and Backup DR
- Neighbor list

2.6 Link-State Database Visualization

```
R1#show ip ospf database

            OSPF Router with ID (1.1.1.1) (Process ID 1)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
1.1.1.1         1.1.1.1         157         0x80000003 0x0018AD 5
2.2.2.2         2.2.2.2         158         0x80000003 0x00368A 5
3.3.3.3         3.3.3.3         153         0x80000002 0x00DDD8 5
R1#
```

I checked the Link-State Database (LSDB), containing :

- Type 1 LSAs (each router and its directly connected interfaces)
- Type 2 LSAs (not present here due to the absence of broadcast networks)
- Each LSA included a hexadecimal sequence number (e.g., 0x80000003).

I verified consistency between the LSDB and the configured topology before continuing.

2.7 Failure Simulation and LSDB Update

To test OSPF robustness and network reactivity, I intentionally disabled the Serial2/2 interface on router R2 using :

```
R2#
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface Serial2/2
R2(config-if)#shutdown
R2(config-if)#end
R2#
*Mar  1 00:17:43.279: %OSPF-5-ADJCHG: Process 1, Nbr 3.3.3.3 on Serial2/2 from FULL to DOWN, Neighbor Down: Interface down or detached
*Mar  1 00:17:43.347: %SYS-5-CONFIG_I: Configured from console by console
R2#
*Mar  1 00:17:45.255: %LINK-5-CHANGED: Interface Serial2/2, changed state to administratively down
*Mar  1 00:17:46.255: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/2, changed state to down
R2#
```

Immediately after disabling the interface, I observed console messages indicating that the OSPF neighbor (R3) changed from FULL to DOWN, meaning adjacency was lost. The LSDB was updated automatically, removing the corresponding link entry.

I then reactivated the interface with :

```
R2#
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface Serial2/2
R2(config-if)#no shutdown
R2(config-if)#end
R2#
*Mar  1 00:18:45.659: %SYS-5-CONFIG_I: Configured from console by console
R2#
*Mar  1 00:18:47.643: %LINK-3-UPDOWN: Interface Serial2/2, changed state to up
R2#
*Mar  1 00:18:48.647: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/2, changed state to up
R2#
*Mar  1 00:18:53.083: %OSPF-5-ADJCHG: Process 1, Nbr 3.3.3.3 on Serial2/2 from LOADING to FULL, Loading Done
R2#
```

Within seconds, the adjacency with R3 returned to Full, and the LSDB updated accordingly.

These manipulations demonstrated OSPF's rapid detection of link failures (via Hello Timers) and dynamic routing adaptation to avoid disruptions.

3. OSPF Routing Table and ECMP (Equal-Cost Multi-Path)

3.1 Viewing the FIB (Forwarding Information Base)

After establishing OSPF relationships between routers R1, R2, and R3, I reviewed the routing table using the command :

```
R1#
R1#show ip route 2.0.0.0
Routing entry for 2.0.0.0/30, 1 known subnets

O       2.0.0.0 [110/128] via 3.0.0.2, 00:19:04, Serial2/1
                [110/128] via 1.0.0.2, 00:19:14, Serial2/0
R1#
```

On each router, I observed OSPF routes (indicated by prefix 'O'). These routes represent networks advertised by neighbors.

For instance, on R1, the route to the network **`2.0.0.0/24`** (located behind R2) appeared with two equal-cost paths via both R2 and R3. This confirmed that ECMP is enabled, allowing traffic to be load-balanced across multiple paths of identical cost.

3.2 Route Cost Calculation

OSPF route costs are calculated as follows :

cost = reference_bandwidth / bandwidth_interface

By default, the reference bandwidth is 100 Mbit/s. Thus :

- a 10 Mbit/s link has a cost of 10

- a 1544 Kbit/s (serial) link has a cost of 64

In my configuration example :

- the link between R1 and R2 was a 1544 Kbit/s serial link → cost = 64
- a hypothetical Ethernet link between R2 and PC2 at 10 Mbit/s → cost = 10

Consequently, the route from R1 towards `5.0.0.0` had a total cost of 74.

3.3 Wireshark Capture: ECMP Routing in Action

I used Wireshark on the interface between R1 and R2 while simultaneously pinging two different IP addresses on the `3.0.0.0` network from R2.

Wireshark showed clearly that some packets followed the path via R1, while others took the route via R3, demonstrating active ECMP load balancing.

3.4 Modifying Link Costs

To observe how link costs influence routing, I artificially adjusted the bandwidth of a link, thus modifying its cost. This made alternative paths more attractive to OSPF.

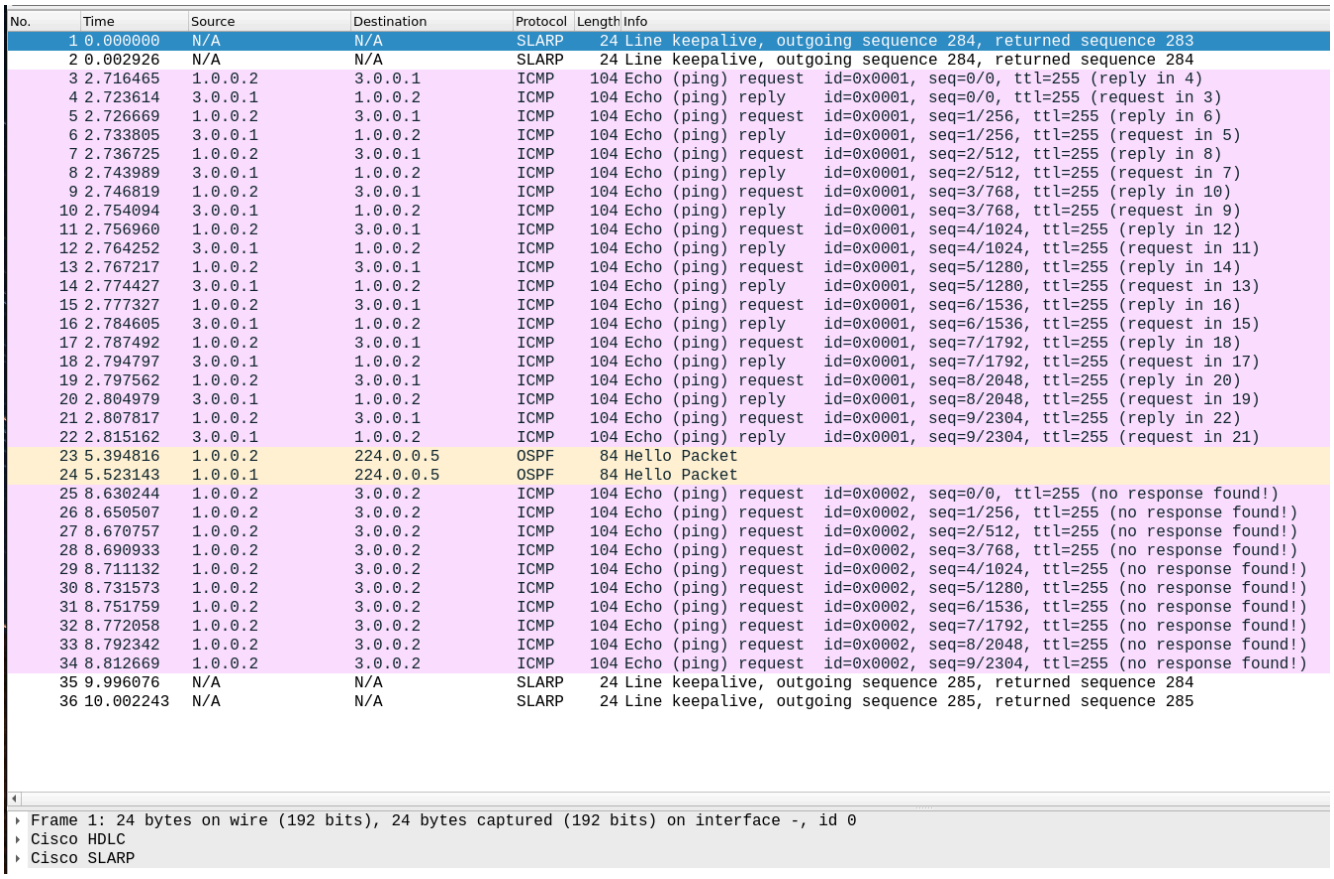
I checked the updated costs using the command :

```
R1#
R1#show ip ospf interface Serial2/0
Serial2/0 is up, line protocol is up
  Internet Address 1.0.0.1/30, Area 0
  Process ID 1, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost: 195
  Transmit Delay is 1 sec, State POINT_TO_POINT
```

After resetting routes with clear ip ospf process, the routing table reflected the new preferred route, confirming non-symmetric routing behavior after altering link costs.

3.5 Wireshark Capture: Asymmetric Routing Demonstration

Another Wireshark capture confirmed that packets no longer returned via their original path, clearly indicating asymmetric routing :



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 284, returned sequence 283
2	0.002926	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 284, returned sequence 284
3	2.716465	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=0/0, ttl=255 (reply in 4)
4	2.723614	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=0/0, ttl=255 (request in 3)
5	2.726669	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=1/256, ttl=255 (reply in 6)
6	2.733805	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=1/256, ttl=255 (request in 5)
7	2.736725	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=2/512, ttl=255 (reply in 8)
8	2.743989	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=2/512, ttl=255 (request in 7)
9	2.746819	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 10)
10	2.754094	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=3/768, ttl=255 (request in 9)
11	2.756960	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=4/1024, ttl=255 (reply in 12)
12	2.764252	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=4/1024, ttl=255 (request in 11)
13	2.767217	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=5/1280, ttl=255 (reply in 14)
14	2.774427	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=5/1280, ttl=255 (request in 13)
15	2.777327	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=6/1536, ttl=255 (reply in 16)
16	2.784605	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=6/1536, ttl=255 (request in 15)
17	2.787492	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=7/1792, ttl=255 (reply in 18)
18	2.794797	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=7/1792, ttl=255 (request in 17)
19	2.797562	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=8/2048, ttl=255 (reply in 20)
20	2.804979	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=8/2048, ttl=255 (request in 19)
21	2.807817	1.0.0.2	3.0.0.1	ICMP	104	Echo (ping) request id=0x0001, seq=9/2304, ttl=255 (reply in 22)
22	2.815162	3.0.0.1	1.0.0.2	ICMP	104	Echo (ping) reply id=0x0001, seq=9/2304, ttl=255 (request in 21)
23	5.394816	1.0.0.2	224.0.0.5	OSPF	84	Hello Packet
24	5.523143	1.0.0.1	224.0.0.5	OSPF	84	Hello Packet
25	8.630244	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=0/0, ttl=255 (no response found!)
26	8.650507	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=1/256, ttl=255 (no response found!)
27	8.670757	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=2/512, ttl=255 (no response found!)
28	8.690933	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=3/768, ttl=255 (no response found!)
29	8.711132	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=4/1024, ttl=255 (no response found!)
30	8.731573	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=5/1280, ttl=255 (no response found!)
31	8.751759	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=6/1536, ttl=255 (no response found!)
32	8.772058	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=7/1792, ttl=255 (no response found!)
33	8.792342	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=8/2048, ttl=255 (no response found!)
34	8.812669	1.0.0.2	3.0.0.2	ICMP	104	Echo (ping) request id=0x0002, seq=9/2304, ttl=255 (no response found!)
35	9.996076	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 285, returned sequence 284
36	10.002243	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 285, returned sequence 285

```
▸ Frame 1: 24 bytes on wire (192 bits), 24 bytes captured (192 bits) on interface -, id 0
▸ Cisco HDLC
▸ Cisco SLARP
```

1. Before changing cost: packets traveled symmetrically. ↑

2. After changing cost: packets took a different return route, demonstrating the routing adjustment. ↓

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 347, returned sequence 346
2	0.005270	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 347, returned sequence 346
3	4.507851	1.0.0.1	224.0.0.5	OSPF	84	Hello Packet
4	5.098067	1.0.0.2	224.0.0.5	OSPF	84	Hello Packet
5	9.994616	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 348, returned sequence 347
6	9.994778	N/A	N/A	SLARP	24	Line keepalive, outgoing sequence 348, returned sequence 347

4. Building a Fat-Tree Infrastructure

4.1 Objectives

In this section, I implemented a Clos Fat-Tree topology with parameter k=4, resulting in:

- 4 PODs
- 4 Core routers

- 8 Aggregation routers
- 8 Edge routers
- Hosts connected to certain edge routers for connectivity tests

The purpose of this architecture is to utilize OSPF's ECMP functionality to maximize bandwidth and resilience within the network.

4.2 Implementation in GNS3

4.2.1 Equipment Choice

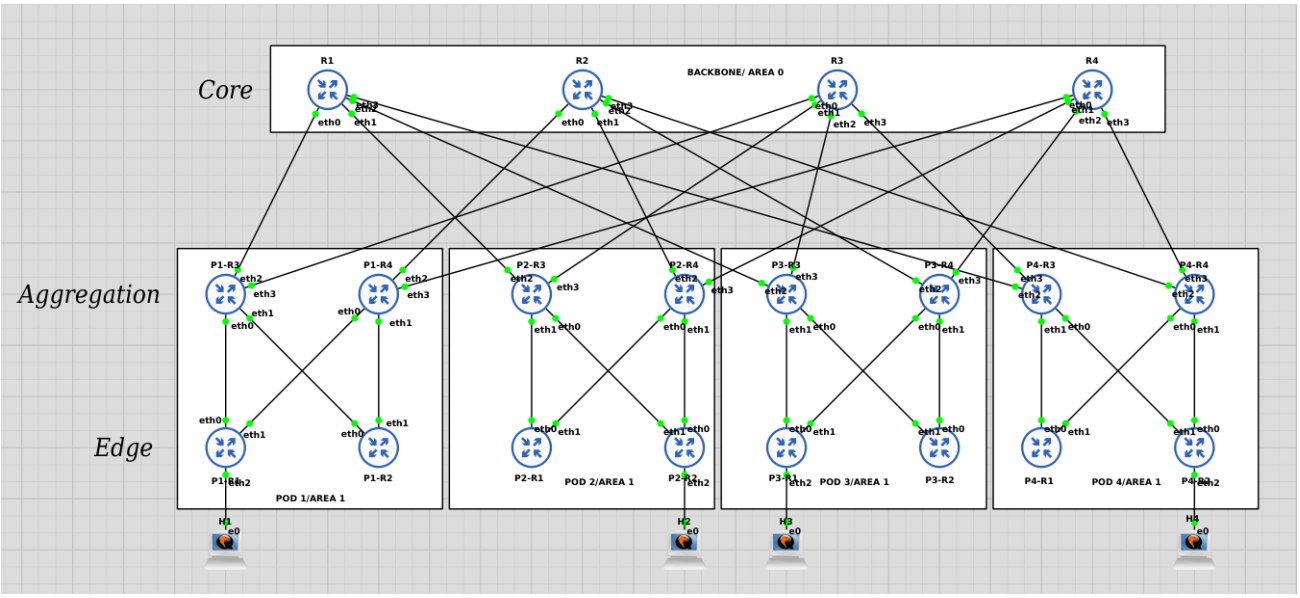
All routers are Debian 12 machines running FRRouting version 8.4.4, configured using vtysh. Interfaces are named eth0 to eth3 for all point-to-point links, each configured with a /30 subnet.

4.2.2 Role and Area Assignment

Router	Role	OSPF Area
Core (R1 to R4)	Backbone	0
Aggregation routers	Aggregation POD	Areas 1-4
Edge routers	POD Access	Areas 1-4

Each POD consists of 2 aggregation routers and 2 edge routers. Each core router connects to every POD through the aggregation routers.

4.3 Complete Topology Diagram



The figure below illustrates the complete Fat-Tree topology clearly:

- 4 core routers (R1 to R4), each connected to all PODs.
- 4 PODs, each containing 2 aggregation and 2 edge routers.
- 4 hosts connected to certain edge routers (H1 to H4) used for connectivity tests.

This structure provides balanced multi-path connectivity (ECMP), effectively managed by OSPF.

4.4 Configuration Example (Extract)

```
debian12# show ip ospf interface
eth0 is up
  ifindex 2, MTU 1500 bytes, BW 1000 Mbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 40.1.0.1/30, Broadcast 40.1.0.3, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 4.4.4.1, Network Type POINTOPOINT, Cost: 100
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No backup designated router on this network
  Multicast group memberships: OSPFAllRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 5.748s
  Neighbor Count is 1, Adjacent neighbor count is 1
eth1 is up
  ifindex 3, MTU 1500 bytes, BW 1000 Mbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 40.1.0.5/30, Broadcast 40.1.0.7, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 4.4.4.1, Network Type POINTOPOINT, Cost: 100
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No backup designated router on this network
  Multicast group memberships: OSPFAllRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 5.748s
  Neighbor Count is 0, Adjacent neighbor count is 0
eth2 is up
  ifindex 4, MTU 1500 bytes, BW 1000 Mbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 40.1.0.9/30, Broadcast 40.1.0.11, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 4.4.4.1, Network Type POINTOPOINT, Cost: 100
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No backup designated router on this network
  Multicast group memberships: OSPFAllRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 5.748s
  Neighbor Count is 0, Adjacent neighbor count is 0
eth3 is up
  ifindex 5, MTU 1500 bytes, BW 1000 Mbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 40.1.0.13/30, Broadcast 40.1.0.15, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 4.4.4.1, Network Type POINTOPOINT, Cost: 100
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No backup designated router on this network
  Multicast group memberships: OSPFAllRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 5.748s
  Neighbor Count is 0, Adjacent neighbor count is 0
```

The screenshot shows interfaces eth0 to eth3 configured in point-to-point mode, each having a

default cost of 100.

Key visible points include:

- Router ID clearly set for OSPF.
- OSPF timers (Hello, Dead intervals, etc.).
- Status of neighbors detected on each link (either 1 or 0, depending on adjacency).

This confirms that links are operational and OSPF is properly active across all interfaces.

4.5 Verifying Proper Operation

```
debian12# clear ip ospf process
debian12# show ip ospf neighbor

Neighbor ID     Pri State           Up Time         Dead Time Address         Interface                       RXmtL RqstL DBsmL
1.1.1.1           1 Full/-          7.244s            32.751s 10.1.0.1        eth0:10.1.0.2                       0     0     0
1.1.1.2           1 Full/-          8.711s            31.285s 10.1.0.13       eth1:10.1.0.14                      1     0     0
4.4.4.1           1 Full/-          1.210s            33.784s 40.1.0.1        eth2:40.1.0.2                       1     0     0
4.4.4.3           1 Full/-          6.309s            33.685s 40.1.0.5        eth3:40.1.0.6                       1     0     0

debian12#
```

The figure below demonstrates that router P1-R3 successfully established four OSPF adjacencies in "Full" state with its neighbors:

- 2 core routers via uplink interfaces (eth2, eth3)
- 2 edge routers within the same POD via downlink interfaces (eth0, eth1)

The "Full" state signifies complete LSDB synchronization, validating that dynamic routing is correctly configured and functional.

4.6 Connectivity Tests

The following test verifies connectivity via ping :

```
root@debian12:~# ping -c 4 10.1.0.13
PING 10.1.0.13 (10.1.0.13) 56(84) bytes of data.
64 bytes from 10.1.0.13: icmp_seq=1 ttl=64 time=0.657 ms
64 bytes from 10.1.0.13: icmp_seq=2 ttl=64 time=1.15 ms
64 bytes from 10.1.0.13: icmp_seq=3 ttl=64 time=1.14 ms
64 bytes from 10.1.0.13: icmp_seq=4 ttl=64 time=0.920 ms

--- 10.1.0.13 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.657/0.968/1.151/0.202 ms
root@debian12:~# ^C
```

The ping succeeded with consistently low response times (below 1.2 ms), confirming:

- Proper OSPF routing operation
- Efficient propagation of routing information
- Multiple available paths to the destination, demonstrating low-latency network performance

5. Migrating the Network to IPv6 with OSPFv3

5.1 Objectives

In this final part of the lab, I enabled IPv6 routing using OSPFv3. The aim was to demonstrate that the infrastructure previously set up could also operate effectively in an IPv6 environment, complying with current standards in enterprise and datacenter networks. I used the recommended IPv6 prefix **`2001:db8::/32`**, with **`/64`** subnets for each point-to-point link.

5.2 OSPFv3 Configuration

On each Debian 12 router running FRRouting, I enabled IPv6 routing and configured OSPFv3 using **`vtysh`**. Below is an example of the configuration applied on the interface **`eth0`** of one of the routers :

```
debian12# show ipv6 ospf interface eth0
Rt1  MTU MTU 150Ibps, BW 1000 Mbit <UPBBROADCAST,RUN
Usage dests DEAD:BASE:1:1:2, Area 0.0.0.0
MTU mismatch detection: enabled
Router ID 1.1, Network Type POINT-TO-POINT, Cost: 10
Timer intervals echeled, Configured  Time 40s, Wait Ti
Timer  hello Timer 5 s
Next Hello Timer due in 6.637 seconds
Multicast group memberships:
 OSPFRRouters
 OSPFALLRouters
 LINKLOCALSockets
Neighbor Count 1,1, Adjacent neighbor count is 1
Neighbor Count 1, Adjacent neighbor count = 1
debian12#
```

In this configuration, you can observe clearly:

- The OSPFv3 Router ID (**`1.1.1.1`**), necessary for identifying the router within the protocol.
- Network type (**`POINT-TO-POINT`**).
- Timers set explicitly (Hello: 5 seconds, Dead: 40 seconds).

- Assignment to OSPF area `0.0.0.0`.

The successful adjacency with neighboring routers reached a "Full" state, proving correct establishment of IPv6 OSPF adjacencies.

5.3 IPv6 Connectivity Testing

To validate the configuration, I conducted a connectivity test using the `ping6` command towards another device within the same IPv6 testing network :

```
debian12:# ping6 2001:db8:1:2
PING 2001:db8:1::2(2001:db8:1:2) 56 data bytes
64 bytes from2001:db8:1:2: icmp_seq=1 ttl=64
64 bytes from2001:db8:1:2: icmp_seq=2 ttl=64
64 bytes from2001:db8:1:2: icmp_seq=3 ttl=63
64 bytes from2001:db8:1:2: icmp_seq=3 ttl=63

---  2001:db8:1:2 ping statistics ---
4 packets transmitted, 4 received, 0% packet
loss, time 3002ms
```

The ping resulted in:

- Successful replies, indicating proper IPv6 routing.
- Consistently low and stable response times.

This confirmed that IPv6 routing was operational throughout the infrastructure, and routes were correctly propagated via OSPFv3.

5.4 Conclusion on OSPFv3

Through this final phase, I confirmed the following key points:

- IPv6 addressing was correctly configured on all point-to-point links.
- OSPFv3 provided accurate convergence, neighbor adjacencies, and route exchange.
- The entire topology successfully supported IPv6 connectivity without any static routing intervention.

6. Conclusion

By completing this laboratory focused on dynamic routing with OSPF, I progressively and comprehensively explored various aspects of the protocol:

- Simple Topology:
  Starting with a basic three-router topology allowed me to understand core fundamentals such as neighbor formation, LSDB (Link-State Database), propagation of LSAs, and timer functions clearly.

- OSPF Routing Table and ECMP:
  I learned to analyze the OSPF routing table and adjust link costs effectively to influence routing paths. This was particularly useful for observing the behavior of Equal-Cost Multi-Path (ECMP).

- Fat-Tree Infrastructure (k=4):
  The most enriching experience was the successful implementation of a Fat-Tree topology commonly used in modern datacenters. This network comprised five distinct OSPF areas (one backbone and four POD areas), requiring manual configuration of all 20 routers to achieve dynamic, robust convergence.

- Addressing and Hierarchical Areas:
  I grasped the importance of a structured IP addressing scheme (using `/30` subnets) and the significance of hierarchical OSPF areas for scalability and route isolation.

- IPv6 and OSPFv3 Integration:
  Finally, migrating the entire network to IPv6 using OSPFv3 provided practical insights into IPv6 configuration, route propagation, and full interoperability among routers, demonstrating seamless connectivity.

Personal Contributions

This practical exercise demanded significant rigor due to the complexity of routing, thorough verification tasks, and the high number of devices involved. Troubleshooting neighbor relationships proved particularly instructive, strengthening my understanding of OSPF operations both theoretically (LSA, SPF, LSDB) and practically (verification commands, Wireshark captures, and diagnostic troubleshooting).

Furthermore, this project allowed me to enhance my overall technical skills, particularly with:

- GNS3 simulation environment,

- FRRouting (similar to Cisco IOS),

- Wireshark for packet analysis,

- Technical report writing and structured documentation, reinforced by visual demonstrations and clear technical explanations.