Network Planning and Management (PGRE)
Master in Electrical and Computers Engineering
Faculty of Engineering of University of Porto

Routing using OSPF e BGP4 protocols

AFONSO QUEIRÓS UP201808903
HUGO GUIA UP201305075
JOÃO LOUREIRO UP201604453

28-03-2020

Contents

1. Introduction

The aim of this work is to study the concepts of Autonomous System (AS), Routing Interior and Exterior Routing Protocols such as OSPF and BGP4 respectively. In order to do this we are using Quagga, which is a program that allows computers to use Gateway protocols such as OSPF, BGP, RIP and IS-IS even though for this study we are only going to need to use the OSPF in our lab PC's. Much like the previous lab works this one is also going to be done remotely, due to present restrictions.

1.1 Autonomous System

An Autonomous System is a set of networks subordinated to a single Technical Management and that shares the same Routing policy (Unique Administrative Management). This means that the network routing is done without direct human interaction while different providers are able to operate the network at the same time. This system uses both Interior Gateway Protocol (IGP) (to operate within an AS to ensure IP connectivity within it) and Exterior Gateway Protocol (EGP) (to operate between AS's to allow routing and policies between them). It was then firstly defined in the RFC 1930 that AS started to run BGP thus being able to scale much better. This made the AS change from 16 bits [1, 65535] to 32 bits [1, 4,294,967,295].

1.2 OSPF

"Open Shortest Path First" is a routing IGP (Interior Gateway Protocol). This protocol works inside an autonomous system and is responsible for connecting every other network in the same AS. It was first declared in RFC 1131, in 1989 and the most recent version was declared in RFC 5340, in 2008, where support for the IPv6 was added.

The protocol manages its resources by adding costs to different networks thus having a way of reduce and optimize traffic. Much like the Dijkstra algorithm it uses the link cost and overall network capacity (which takes into account different parameters like bandwidth and link speed). In order to do this, HELLO messages are traded between every other router, through multicast, until every router has every link cost in the network and all are properly synchronized with each other. There are also maintenance packets being traded, such as LSP (link state packets) which keep information of the network topology and the state of certain links in case the network suffers any transformation.

It's important to mention that OSPF also divides into different routing areas, which are attributed in 32 bit numbers. The 0.0.0.0 is defined as the first area and all of the subsequent areas should connect to this backbone.

1.3 BGP

"Border Gateway Protocol" is the protocol responsible for EGP (Exterior Gateway Protocol). It is in charge of routing different AS (Autonomous Systems) such as the ones presented previously thus being one level above it. Its main job is to define the optimal routes between different AS and connect different ISP (Internet Service Providers). There is also iBGP which is used as an Internal Gateway Protocol but for this study we will only be testing the EGP version of it, eBGP4.

The way BGP works is similar the other routing protocols, but also different in other aspects once it communicates using TCP which means it works in a different layer (Application Layer). This means that all TCP restrictions have to be met, like timers such as "keepalive", "hold" and others in order to mantain the session and knowing when there is an update in the network. eBGP is designed to run only between directly connected neighbors which is the main reason it scales better than OSPF.

2. Interior Routing

The first thing to do is to install Quagga tool that makes it possible, through certain protocols such as OSPF and BGP, to communicate between machines, transforming them into routers (using deamons), where the possibility of exchanging information between them through routing tables is created. The command "apt-get install quagga" was used to install this service.

After installation, two files were created corresponding to the configuration of the zebra and ospf daemon, in /etc/quagga directory, whose contents are shown in the following figure.

```
root@tux23:~# cat /etc/quagga/zebra.conf
! -*- zebra -*-
!
! zebra sample configuration file
!
! $Id: zebra.conf.sample,v 1.1 2002/12/13 20:15:30 paul Exp $
!
hostname tux23
password botagel
enable password botagel
!
! Interface's description.
!
!interface lo
! description test of desc.
!
!interface sit0
! multicast

!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!

!log file zebra.log

interface eth1
        ip address 172.16.21.1/24
```

Figure 2.1: tux23 zebra config file

```
root@tux24:~# cat /etc/quagga/zebra.conf
! -*- zebra -*-
!
! zebra sample configuration file
!
! $Id: zebra.conf.sample,v 1.1 2002/12/13 20:15:30 paul Exp $
!
hostname tux24
password botagel
enable password botagel
!
! Interface's description.
!
!interface lo
! description test of desc.
!
!interface sit0
! multicast

!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!

!log file zebra.log

interface eth1
        ip address 172.16.22.3/24
```

Figure 2.2: tux24 zebra config file

```
root@tux22:~# cat /etc/quagga/zebra.conf
! -*- zebra -*-
!
! zebra sample configuration file
!
! $Id: zebra.conf.sample,v 1.1 2002/12/13 20:15:30 paul Exp $
!
hostname tux22
password botagel
enable password botagel
log file /var/log/quagga/zebra_22.log
!
! Interface's description.
!
!interface lo
! description test of desc.
!
!interface sit0
! multicast

!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!

interface eth2
        ip address 172.16.22.2/24

interface eth1
        ip address 172.16.21.2/24
```

Figure 2.3: tux22 zebra config file

```
root@tux23:~# cat /etc/quagga/ospfd.conf
! -*- ospf -*-
!
! OSPFd sample configuration file
!
!
hostname tux23
password botagel
!enable password botagel
!
!router ospf
!  network 192.168.1.0/24 area 0
!
log stdout
router ospf
 network 172.16.21.0/24 area 0
interface eth1
```

Figure 2.4: tux23 ospf config file

```
root@tux24:~# cat /etc/quagga/ospfd.conf
! -*- ospf -*-
!
! OSPFd sample configuration file
!
!
hostname tux24
password botagel
!enable password botagel
!
!router ospf
!  network 192.168.1.0/24 area 0
!
log stdout
router ospf
 network 172.16.22.0/24 area 0
interface eth1
```

Figure 2.5: tux24 ospf config file

```
root@tux22:~# cat /etc/quagga/ospfd.conf
! -*- ospf -*-
!
! OSPFd sample configuration file
!
!
hostname tux22
password botagel
!enable password botagel
!
!router ospf
!  network 192.168.1.0/24 area 0
!
log stdout

router ospf
        network 172.16.21.0/24 area 0
        network 172.16.22.0/24 area 0

interface eth1
interface eth2
```

Figure 2.6: tux22 ospf config file

Following that, we configured the system architecture, more precisely machine IPs, router and switch commands to perform trunk between different VLANs. The table below shows briefly the addresses reserved for each entity.

tux	if		MAC	switch	VLAN	router
tux21	eth0	172.16.1.21	00:c0:df:08:d5:b2	Fa0/1		
	eth1	172.16.1.121	00:0f:fe:8c:af:9d	Fa0/2		
tux22	eth0	172.16.1.22	00:21:5a:61:2b:72	Fa0/3		
	eth1	172.16.21.2	00:c0:df:13:4c:f3	Fa0/4	21	
	eth2	172.16.22.2	00:01:02:9f:80:17	Fa0/5	22	
tux23	eth0	172.16.1.23	00:21:5a:5a:7d:12	Fa0/6		
	eth1	172.16.21.1	00:50:fc:ee:0e:93	Fa0/7	21	
	eth2		00:01:02:9f:81:49			
tux24	eth0	172.16.1.24	00:08:54:50:3f:2c	Fa0/8		
	eth1	172.16.22.3	00:22:64:a6:a4:f1	Fa0/9	22	
	eth2		00:01:02:a0:fe:ae			
netlab				Gi0/1		
		172.16.21.4 /// 172.16.22.4		Gi0/2		GE0/0
						GE0/1

Figure 2.7: Address of entire architecture

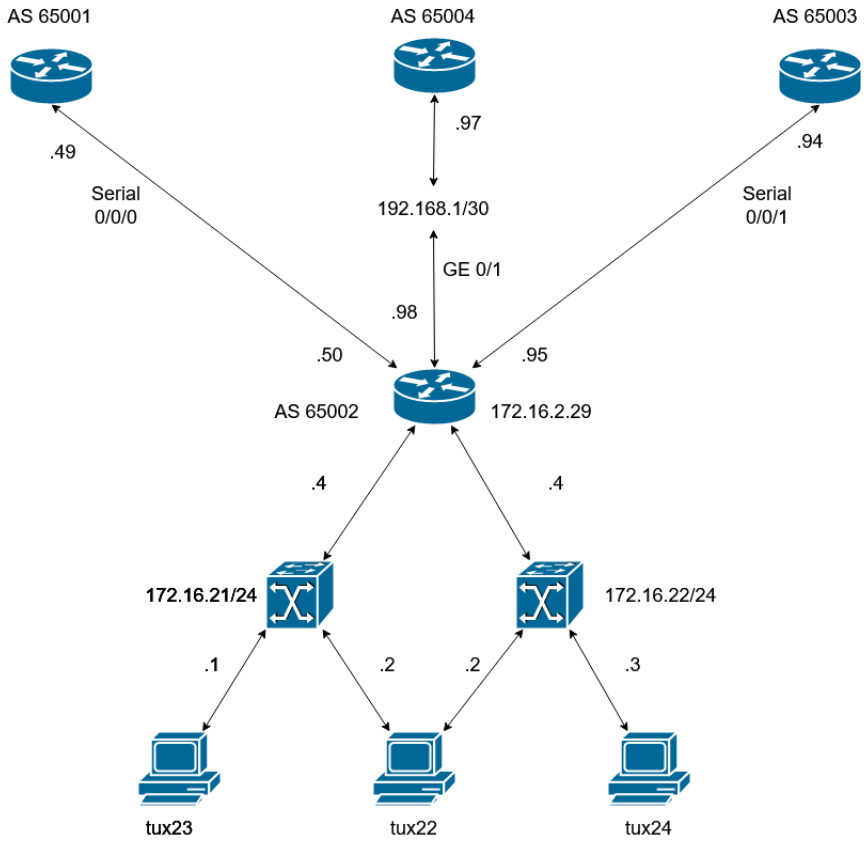

Figure 2.8: Network setup

As for the configuration of the switch, two VLANs were created. On VLAN 21 tux23 eth1 and tux22 eth1 are connected. On VLAN 22, tux24 eth1 and tux22 eth2 are connected. Finally, a trunk was created on the GigabitEthernet 0/2 port and the VLANS previously created were assigned to it. The settings can be

checked below:

```
configure terminal

vlan 21-22
exit

interface fastEthernet 0/4
switchport mode access
switch port access vlan 21
exit

interface fastEthernet 0/7
switchport mode access
switch port access vlan 21
exit

interface fastEthernet 0/5
switchport mode access
switch port access vlan 22
exit

interface fastEthernet 0/9
switchport mode access
switch port access vlan 22
exit

interface gigabitEthernet 0/2
switchport mode trunk
switchport trunk allowed vlan add 21,22
exit
```

Figure 2.9: Switch configuration

```
configure terminal
interface GigabitEthernet 0/0
no ip address
exit

interface GigabitEthernet 0/0.1
encapsulation dot1Q 1 native
ip address 172.16.1.29 255.255.255.0
no shutdown
exit

interface GigabitEthernet 0/0.21
encapsulation dot1Q 21
ip address 172.16.21.4 255.255.255.0
no shutdown
exit

interface GigabitEthernet 0/0.22
encapsulation dot1Q 22
ip address 172.16.22.4 255.255.255.0
no shutdown
exit

router ospf 1
router-id 172.16.21.4
auto-cost reference -bandwidth 1000
log-adjacency-changes
network 172.16.21.1 0.0.0.255 area 0.0.0.0
network 172.16.22.1 0.0.0.255 area 0.0.0.0
default-information originate always
end
```

Figure 2.10: Router Configuration

Explaining certain commands:

- "log-adjacency-changes" - router to send a syslog message when the state of an Open Shortest Path First (OSPF) neighbor changes.
- "default-information originate always" - Specifies to always advertise the default route regardless of whether the route table has a default route.

In tux23 and tux24 it was needed to add the default gateway for these new networks that have been implemented. After modifying them, the routing table in each tux reports as follows:

```
root@tux23:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.21.4     0.0.0.0         UG    0      0        0 eth1
0.0.0.0         172.16.1.254    0.0.0.0         UG    0      0        0 eth0
172.16.1.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.21.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
172.16.22.0     172.16.21.4     255.255.255.0   UG    20     0        0 eth1
```

Figure 2.11: tux23 routing table

```
root@tux24:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.22.4     0.0.0.0         UG    0      0        0 eth1
0.0.0.0         172.16.1.254    0.0.0.0         UG    0      0        0 eth0
172.16.1.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.21.0     172.16.22.4     255.255.255.0   UG    20     0        0 eth1
172.16.22.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
```

Figure 2.12: tux24 routing table

3. Exterior Routing

In order to allow for exterior routing, BGP peering between Autonomous Systems (AS) needed to be activated on the router. Therefore, new commands were issued, adding to those already previously mentioned in this report. According to the supplied information, our bench communicates with bench number 1 via serial 0/0/0 interface, with bench number 3 via serial 0/0/1 and with bench number 4 via Gigabit Ethernet 0/1. So, these three neighbour AS were added to the router configuration as seen below.

```
configure terminal

interface serial 0/0/0
ip address 192.168.1.50 255.255.255.252
no shutdown
exit

interface serial 0/0/1
ip address 192.168.1.93 255.255.255.252
no shutdown
exit

interface GigabitEthernet 0/1
ip address 192.168.1.98 255.255.255.252
no shutdown
exit

router bgp 65002

neighbor 192.168.1.49 remote-as 65001
neighbor 192.168.1.94 remote-as 65003
neighbor 192.168.1.97 remote-as 65004
```

Figure 3.1: BGP enabling commands

After configuring the router with the previous commands, the final running config of the router at this point was as shown at the next picture. At this point, we could connect with the benches we were supposed to.

```
interface GigabitEthernet0/0.1
 encapsulation dot1Q 1 native
 ip address 172.16.1.29 255.255.255.0
!
interface GigabitEthernet0/0.21
 encapsulation dot1Q 21
 ip address 172.16.21.4 255.255.255.0
!
interface GigabitEthernet0/0.22
 encapsulation dot1Q 22
 ip address 172.16.22.4 255.255.255.0
!
interface GigabitEthernet0/1
 ip address 192.168.1.98 255.255.255.252
 duplex auto
 speed auto
!
interface Serial0/0/0
 ip address 192.168.1.50 255.255.255.252
 clock rate 2000000
!
interface Serial0/0/1
 ip address 192.168.1.93 255.255.255.252
 no clock rate 2000000
!
router ospf 1
 router-id 172.16.21.4
 log-adjacency-changes
 auto-cost reference-bandwidth 1000
 network 172.16.21.0 0.0.0.255 area 0.0.0.0
 network 172.16.22.0 0.0.0.255 area 0.0.0.0
 default-information originate always
!
router bgp 65002
 no synchronization
 bgp log-neighbor-changes
 network 172.16.21.0 mask 255.255.255.0
 network 172.16.22.0 mask 255.255.255.0
 redistribute connected
 neighbor 192.168.1.49 remote-as 65001
 neighbor 192.168.1.94 remote-as 65003
 neighbor 192.168.1.97 remote-as 65004
 no auto-summary
!
```

Figure 3.2: Router running-config

4. Results

4.1 Interior Routing

In order to test the efficiency of the built architecture, certain tests were performed. One of them consists of cutting the physical connection between the TUX22 and VLAN21, so that the traffic, to travel from TUX23 to TUX24, has to be routed through the router in order to arrive at the intended machine. The test has concluded with success. The traceroute below shows the path that traffic travels through the system.

```
root@tux23:~# traceroute -i eth1 172.16.22.2
traceroute to 172.16.22.2 (172.16.22.2), 30 hops max, 60 byte packets
 1  172.16.21.4 (172.16.21.4)  0.427 ms  0.506 ms  0.530 ms
 2  172.16.22.2 (172.16.22.2)  0.351 ms  0.327 ms  0.317 ms
```

Figure 4.1: Trace Route of the System.

The routing tables of the machines that has ospf enable are presented in the next two figures, so that it can demonstrate the correct configuration of the entire process. To access this information, the command "telnet localhost 2601" was issued in order to access the quagga terminal. In it, we can type "show ip route ospf" to obtain relevant information about the routing tables. The first picture presents information about the system with all the connections physically established. The other picture has information about the experiment where we

```
root@tux22:~# telnet localhost 2601
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
tux22> enable
Password:
tux22# show ip route ospf
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel, N - NHRP,
       > - selected route, * - FIB route

O   0.0.0.0/0 [110/1] via 172.16.21.4, eth1, 00:01:12
                      via 172.16.22.4, eth2, 00:01:12
O   172.16.21.0/24 [110/10] is directly connected, eth1, 00:01:13
O   172.16.22.0/24 [110/10] is directly connected, eth2, 00:15:46
tux22#
```

Figure 4.2: Routing table while there is a physical connection established

```
root@tux22:~# telnet localhost 2601
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
tux22> enable
Password:
tux22# show ip route ospf
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel, N - NHRP,
       > - selected route, * - FIB route

O   0.0.0.0/0 [110/1] via 172.16.22.4, eth2, 00:01:03
O>* 172.16.21.0/24 [110/11] via 172.16.22.4, eth2, 00:01:04
O   172.16.22.0/24 [110/10] is directly connected, eth2, 00:17:14
tux22#
```

Figure 4.3: Routing table with the removal of the physical connection

4.2 Exterior Routing

After configuring the BGP as previously mentioned in this report, issuing the command "show bgp" in the router console provided us the full routing table of the area border router.

```
tux-rtr2#show bgp
BGP table version is 75, local router ID is 172.16.22.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.1.0/24    0.0.0.0                  0         32768 ?
*> 172.16.11.0/24   192.168.1.49             0             0 65001 i
*> 172.16.12.0/24   192.168.1.49             0             0 65001 i
*> 172.16.21.0/24   0.0.0.0                  0         32768 i
*> 172.16.22.0/24   0.0.0.0                  0         32768 i
*> 172.16.31.0/24   192.168.1.94             0             0 65003 i
*> 172.16.32.0/24   192.168.1.94             0             0 65003 i
*> 172.16.41.0/24   192.168.1.97             0             0 65004 i
*> 172.16.42.0/24   192.168.1.97             0             0 65004 i
*> 172.16.51.0/24   192.168.1.97                           0 65004 65005 i
*> 172.16.52.0/24   192.168.1.97                           0 65004 65005 i
*> 192.168.1.48/30  0.0.0.0                  0         32768 ?
*> 192.168.1.92/30  0.0.0.0                  0         32768 ?
*> 192.168.1.96/30  0.0.0.0                  0         32768 ?
```

Figure 4.4: BGP routing table

In addition, we can also use the command "show ip route" to get more information about the current state of the routing table.

```
tux-rtr2#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, + - replicated route

Gateway of last resort is not set

      172.16.0.0/16 is variably subnetted, 15 subnets, 2 masks
C        172.16.1.0/24 is directly connected, GigabitEthernet0/0.1
L        172.16.1.29/32 is directly connected, GigabitEthernet0/0.1
B        172.16.11.0/24 [20/0] via 192.168.1.49, 23:16:45
S        172.16.11.0/32 [1/0] via 192.168.1.49
B        172.16.12.0/24 [20/0] via 192.168.1.49, 23:16:45
C        172.16.21.0/24 is directly connected, GigabitEthernet0/0.21
L        172.16.21.4/32 is directly connected, GigabitEthernet0/0.21
C        172.16.22.0/24 is directly connected, GigabitEthernet0/0.22
L        172.16.22.4/32 is directly connected, GigabitEthernet0/0.22
B        172.16.31.0/24 [20/0] via 192.168.1.94, 2d01h
B        172.16.32.0/24 [20/0] via 192.168.1.94, 2d01h
B        172.16.41.0/24 [20/0] via 192.168.1.97, 2d00h
B        172.16.42.0/24 [20/0] via 192.168.1.97, 2d00h
B        172.16.51.0/24 [20/0] via 192.168.1.97, 21:54:43
B        172.16.52.0/24 [20/0] via 192.168.1.97, 21:54:43
     192.168.1.0/24 is variably subnetted, 6 subnets, 2 masks
C        192.168.1.48/30 is directly connected, Serial0/0/0
L        192.168.1.50/32 is directly connected, Serial0/0/0
C        192.168.1.92/30 is directly connected, Serial0/0/1
L        192.168.1.93/32 is directly connected, Serial0/0/1
C        192.168.1.96/30 is directly connected, GigabitEthernet0/1
L        192.168.1.98/32 is directly connected, GigabitEthernet0/1
```

Figure 4.5: IP route table

In tux24, if we issue the traceroute command to the equivalent in other networks, we can see the way the connection is established as well as the network hops.

```
root@tux24:~# traceroute -i eth1 172.16.12.3
traceroute to 172.16.12.3 (172.16.12.3), 30 hops max, 60 byte packets
 1  172.16.22.4 (172.16.22.4)  0.563 ms  0.584 ms  0.618 ms
 2  192.168.1.49 (192.168.1.49)  1.021 ms  1.250 ms  1.518 ms
 3  172.16.12.3 (172.16.12.3)  1.889 ms * *
root@tux24:~# traceroute -i eth1 172.16.32.3
traceroute to 172.16.32.3 (172.16.32.3), 30 hops max, 60 byte packets
 1  172.16.22.4 (172.16.22.4)  0.583 ms  0.606 ms  0.640 ms
 2  192.168.1.94 (192.168.1.94)  0.955 ms  1.161 ms  1.426 ms
 3  172.16.32.3 (172.16.32.3)  1.915 ms * *
root@tux24:~# traceroute -i eth1 172.16.42.3
traceroute to 172.16.42.3 (172.16.42.3), 30 hops max, 60 byte packets
 1  172.16.22.4 (172.16.22.4)  0.440 ms  0.507 ms  0.534 ms
 2  192.168.1.97 (192.168.1.97)  0.690 ms  0.677 ms  0.790 ms
 3  172.16.42.3 (172.16.42.3)  0.628 ms *  0.575 ms
```

Figure 4.6: traceroute results for the different network devices

In order to ensure that the BGP routing process forwards our AS traffic with the neighbors with the highest AS number, which in our case is AS65004, a slight alteration had to be done to the router configuration. The following commands were issued:

```
configure terminal

route-map List 1 permit 10
set local-preference 101
exit

router bgp 65002
neighbor 192.168.1.97 route-map List 1 in
exit
```

Figure 4.7: Commands to make the traffic go through the neighbor with the highest numbered autonomous station

5. Conclusion

This laboratory study (remote access lab-study) allowed us to simulate and understand better the concepts of Interior and Exterior routing. For the OSPF protocol we were able to configure the machines using quagga and thus simulate IGP (Internal Gateway Protocol) using only our net laboratory machines. This gave us a better understanding of how OSPF behaves and how well it works autonomously. This part of the study also forced us to train our networking routing configuration skills, once the switches and router also needed tweaking.

As for the EGP (External Gateway Protocol), we managed to properly set up BGP and so connect the different AS of the netlab completing the full mesh of this topology. We just needed to to add to the routing list the correspondent AS that we would be connecting to and by doing so we're complementing the OSPF protocol previously configured.

By doing this, BGP allows us to connect to other autonomous systems that are physically connected to our AS, maintaining individual backbones.

Summing up, even with some trouble along the way regarding some configuration issues, we believe that we completed the job and our learning goals were achieved.