Arquiteturas de Comunicação

Project report

Bruno Gomes, Diogo Silva

Universidade de Aveiro

Arquiteturas de Comunicação

DEPARTAMENTO DE ELETRÓNICA
TELECOMUNICAÇÕES E INFORMÁTICA

Project report

Bruno Gomes, Diogo Silva
(103320) brunofgomes@ua.pt, (104341) diogobranco.as@ua.pt

30/12/2024

Abstract

This project focuses on the technical design, configuration, and testing of a Content Delivery Network (CDN) infrastructure to support a multi-client enterprise environment. The project involves the company CDN4ALL LLC, a provider of Infrastructure-as-a-Service (IaaS) solutions, which operates across geographically distributed Points of Presence (PoPs) in Porto, Lisbon, Barcelona, and Chicago.

Key objectives:

1. **Private Network Design:** Implementing private networks for both large clients and small-to-medium enterprises (SMEs) with guaranteed bandwidth allocation and advanced routing services.

2. **Traffic Differentiation:** Configuring traffic prioritization policies to ensure service quality and efficiency.

3. **Integration of Advanced Networking Technologies:** Utilizing MPLS VPNs, VXLAN, and BGP EVPN mechanisms for robust connectivity and client-specific requirements.

The project leverages VyOS and Cisco C7200 devices for network infrastructure, with specific emphasis on efficient interconnection, advanced service routing, and assured forwarding policies.

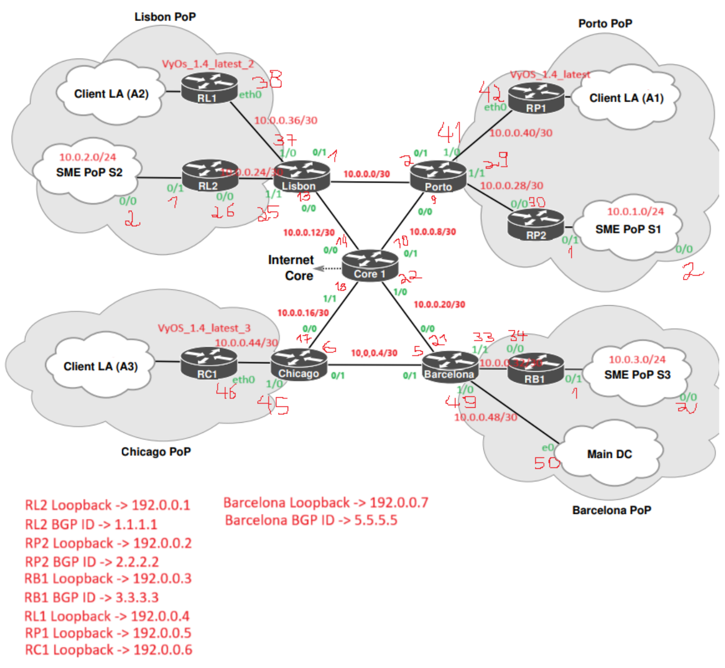
Contents

List of Figures

Chapter 1

Basic assembly and core connectivity

1.1 Network diagram



RL2 Loopback -> 192.0.0.1
RL2 BGP ID -> 1.1.1.1
RP2 Loopback -> 192.0.0.2
RP2 BGP ID -> 2.2.2.2
RB1 Loopback -> 192.0.0.3
RB1 BGP ID -> 3.3.3.3
RL1 Loopback -> 192.0.0.4
RP1 Loopback -> 192.0.0.5
RC1 Loopback -> 192.0.0.6

Barcelona Loopback -> 192.0.0.7
Barcelona BGP ID -> 5.5.5.5

Figure 1.1: Planned network topology

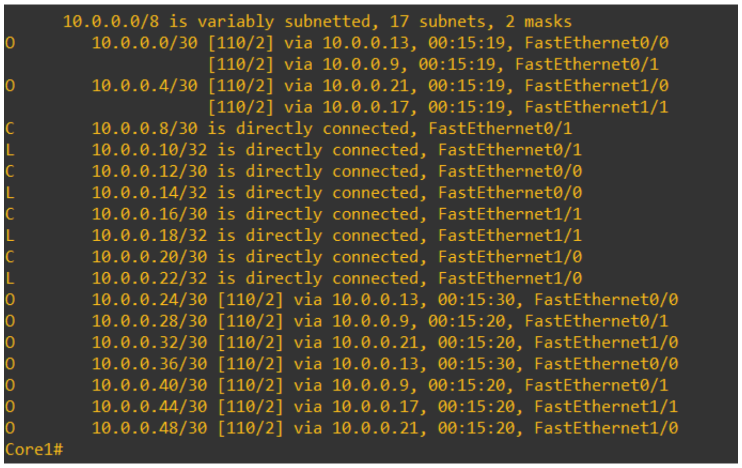
1.2 Implementation

As seen on the network diagram above, we used several IPv4 networks with a mask of /30, we started in the address 10.0.0.0 and then used as many networks necessary to obtain full connectivity, we decided to use a mask of /30 because in this way we do not waste as many addresses if we used a /24 mask for instance.

For the core implementation of this network, we decided to run OSPF and MPLS protocols on routers, Lisbon, Porto, Core1, Chicago and Barcelona. RL2, RP2 and RB1 are also running these protocols but only on the MPLS core side, on the client side there is no OSPF nor MPLS. OSPF was used in order to provide the best path for routing data to its intended destination, and MPLS LDP was used to build label-switch path (LSP) databases that are used to forward traffic through MPLS networks.

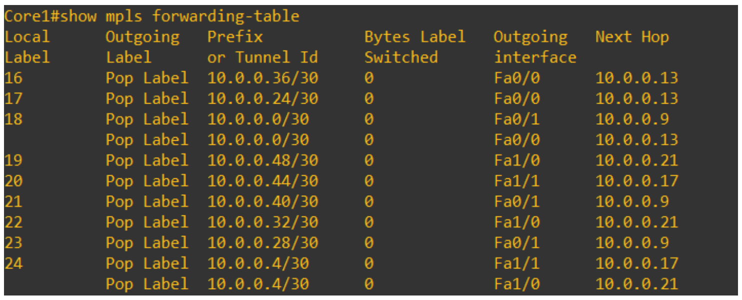
MPLS is important for our project, because it allow us to optimize traffic forwarding based on the different labels attributed to the routing paths, this aspect works well with traffic engineering methods which should have been explored in this project.

Instead of performing pings from every single point in the network to verify full connectivity, we prefer to demonstrate the routes learned by OSPF on each main router, along with the MPLS forwarding table for each router.



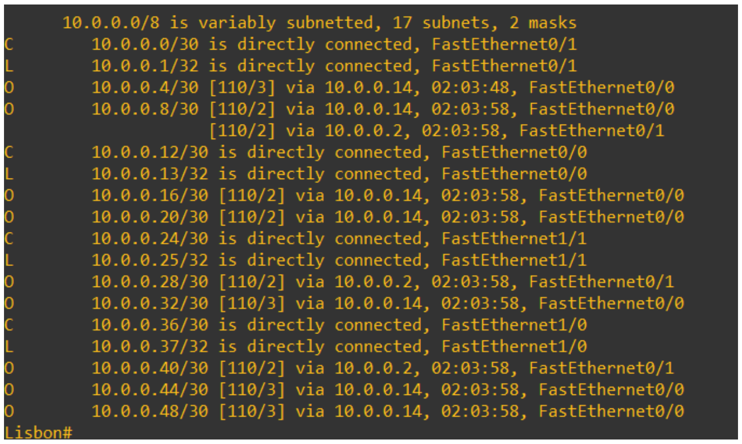
```
      10.0.0.0/8 is variably subnetted, 17 subnets, 2 masks
O        10.0.0.0/30 [110/2] via 10.0.0.13, 00:15:19, FastEthernet0/0
                     [110/2] via 10.0.0.9, 00:15:19, FastEthernet0/1
O        10.0.0.4/30 [110/2] via 10.0.0.21, 00:15:19, FastEthernet1/0
                     [110/2] via 10.0.0.17, 00:15:19, FastEthernet1/1
C        10.0.0.8/30 is directly connected, FastEthernet0/1
L        10.0.0.10/32 is directly connected, FastEthernet0/1
C        10.0.0.12/30 is directly connected, FastEthernet0/0
L        10.0.0.14/32 is directly connected, FastEthernet0/0
C        10.0.0.16/30 is directly connected, FastEthernet1/1
L        10.0.0.18/32 is directly connected, FastEthernet1/1
C        10.0.0.20/30 is directly connected, FastEthernet1/0
L        10.0.0.22/32 is directly connected, FastEthernet1/0
O        10.0.0.24/30 [110/2] via 10.0.0.13, 00:15:30, FastEthernet0/0
O        10.0.0.28/30 [110/2] via 10.0.0.9, 00:15:20, FastEthernet0/1
O        10.0.0.32/30 [110/2] via 10.0.0.21, 00:15:20, FastEthernet1/0
O        10.0.0.36/30 [110/2] via 10.0.0.13, 00:15:30, FastEthernet0/0
O        10.0.0.40/30 [110/2] via 10.0.0.9, 00:15:20, FastEthernet0/1
O        10.0.0.44/30 [110/2] via 10.0.0.17, 00:15:20, FastEthernet1/1
O        10.0.0.48/30 [110/2] via 10.0.0.21, 00:15:20, FastEthernet1/0
Core1#
```

Figure 1.2: Core1 router learned routes



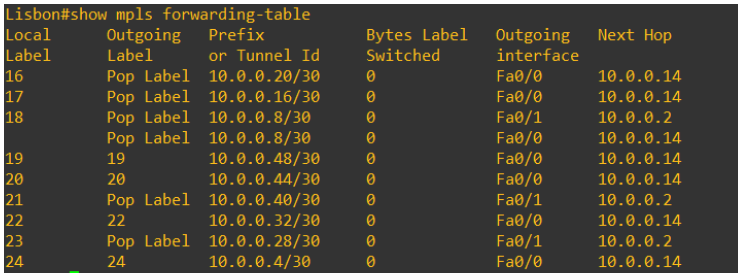
```
Core1#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.0.0.36/30     0             Fa0/0      10.0.0.13
17         Pop Label  10.0.0.24/30     0             Fa0/0      10.0.0.13
18         Pop Label  10.0.0.0/30      0             Fa0/1      10.0.0.9
           Pop Label  10.0.0.0/30      0             Fa0/0      10.0.0.13
19         Pop Label  10.0.0.48/30     0             Fa1/0      10.0.0.21
20         Pop Label  10.0.0.44/30     0             Fa1/1      10.0.0.17
21         Pop Label  10.0.0.40/30     0             Fa0/1      10.0.0.9
22         Pop Label  10.0.0.32/30     0             Fa1/0      10.0.0.21
23         Pop Label  10.0.0.28/30     0             Fa0/1      10.0.0.9
24         Pop Label  10.0.0.4/30      0             Fa1/1      10.0.0.17
           Pop Label  10.0.0.4/30      0             Fa1/0      10.0.0.21
```

Figure 1.3: Core1 router MPLS forwarding-table



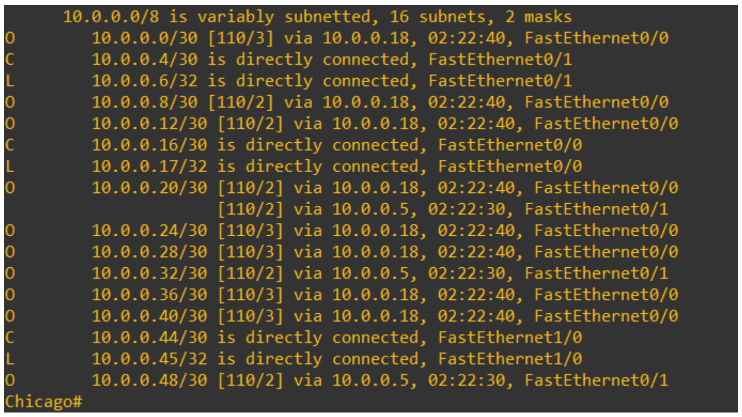
```
      10.0.0.0/8 is variably subnetted, 17 subnets, 2 masks
C        10.0.0.0/30 is directly connected, FastEthernet0/1
L        10.0.0.1/32 is directly connected, FastEthernet0/1
O        10.0.0.4/30 [110/3] via 10.0.0.14, 02:03:48, FastEthernet0/0
O        10.0.0.8/30 [110/2] via 10.0.0.14, 02:03:58, FastEthernet0/0
                     [110/2] via 10.0.0.2, 02:03:58, FastEthernet0/1
C        10.0.0.12/30 is directly connected, FastEthernet0/0
L        10.0.0.13/32 is directly connected, FastEthernet0/0
O        10.0.0.16/30 [110/2] via 10.0.0.14, 02:03:58, FastEthernet0/0
O        10.0.0.20/30 [110/2] via 10.0.0.14, 02:03:58, FastEthernet0/0
C        10.0.0.24/30 is directly connected, FastEthernet1/1
L        10.0.0.25/32 is directly connected, FastEthernet1/1
O        10.0.0.28/30 [110/2] via 10.0.0.2, 02:03:58, FastEthernet0/1
O        10.0.0.32/30 [110/3] via 10.0.0.14, 02:03:58, FastEthernet0/0
C        10.0.0.36/30 is directly connected, FastEthernet1/0
L        10.0.0.37/32 is directly connected, FastEthernet1/0
O        10.0.0.40/30 [110/2] via 10.0.0.2, 02:03:58, FastEthernet0/1
O        10.0.0.44/30 [110/3] via 10.0.0.14, 02:03:58, FastEthernet0/0
O        10.0.0.48/30 [110/3] via 10.0.0.14, 02:03:58, FastEthernet0/0
Lisbon#
```

Figure 1.4: Lisbon router learned routes



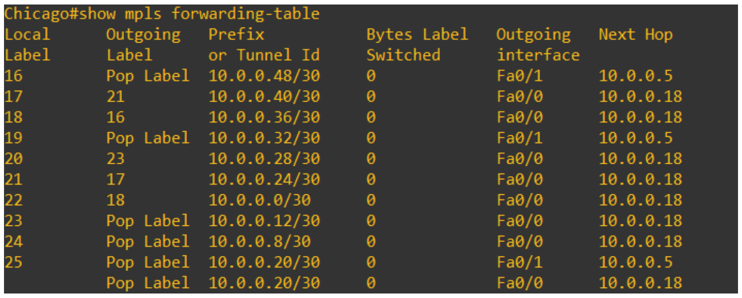
```
Lisbon#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.0.0.20/30     0             Fa0/0      10.0.0.14
17         Pop Label  10.0.0.16/30     0             Fa0/0      10.0.0.14
18         Pop Label  10.0.0.8/30      0             Fa0/1      10.0.0.2
           Pop Label  10.0.0.8/30      0             Fa0/0      10.0.0.14
19         19         10.0.0.48/30     0             Fa0/0      10.0.0.14
20         20         10.0.0.44/30     0             Fa0/0      10.0.0.14
21         Pop Label  10.0.0.40/30     0             Fa0/1      10.0.0.2
22         22         10.0.0.32/30     0             Fa0/0      10.0.0.14
23         Pop Label  10.0.0.28/30     0             Fa0/1      10.0.0.2
24         24         10.0.0.4/30      0             Fa0/0      10.0.0.14
```

Figure 1.5: Lisbon router MPLS forwarding-table



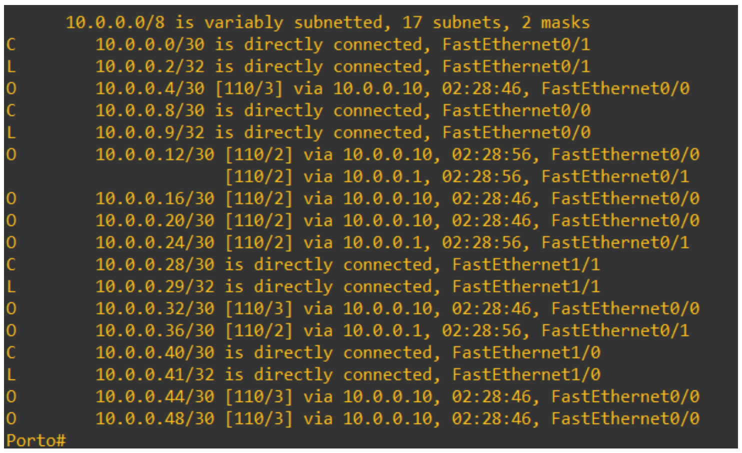
```
      10.0.0.0/8 is variably subnetted, 16 subnets, 2 masks
O        10.0.0.0/30 [110/3] via 10.0.0.18, 02:22:40, FastEthernet0/0
C        10.0.0.4/30 is directly connected, FastEthernet0/1
L        10.0.0.6/32 is directly connected, FastEthernet0/1
O        10.0.0.8/30 [110/2] via 10.0.0.18, 02:22:40, FastEthernet0/0
O        10.0.0.12/30 [110/2] via 10.0.0.18, 02:22:40, FastEthernet0/0
C        10.0.0.16/30 is directly connected, FastEthernet0/0
L        10.0.0.17/32 is directly connected, FastEthernet0/0
O        10.0.0.20/30 [110/2] via 10.0.0.18, 02:22:40, FastEthernet0/0
                      [110/2] via 10.0.0.5, 02:22:30, FastEthernet0/1
O        10.0.0.24/30 [110/3] via 10.0.0.18, 02:22:40, FastEthernet0/0
O        10.0.0.28/30 [110/3] via 10.0.0.18, 02:22:40, FastEthernet0/0
O        10.0.0.32/30 [110/2] via 10.0.0.5, 02:22:30, FastEthernet0/1
O        10.0.0.36/30 [110/3] via 10.0.0.18, 02:22:40, FastEthernet0/0
O        10.0.0.40/30 [110/3] via 10.0.0.18, 02:22:40, FastEthernet0/0
C        10.0.0.44/30 is directly connected, FastEthernet1/0
L        10.0.0.45/32 is directly connected, FastEthernet1/0
O        10.0.0.48/30 [110/2] via 10.0.0.5, 02:22:30, FastEthernet0/1
Chicago#
```

Figure 1.6: Chicago router learned routes



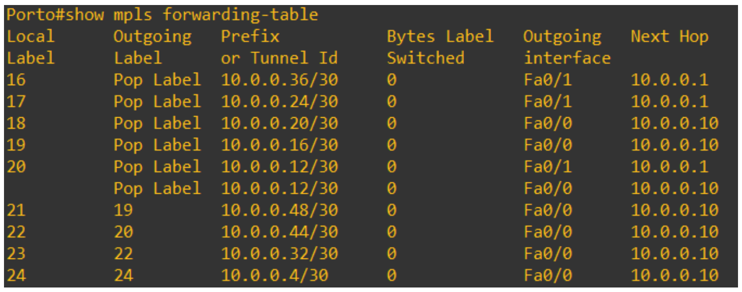
```
Chicago#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.0.0.48/30     0             Fa0/1      10.0.0.5
17         21         10.0.0.40/30     0             Fa0/0      10.0.0.18
18         16         10.0.0.36/30     0             Fa0/0      10.0.0.18
19         Pop Label  10.0.0.32/30     0             Fa0/1      10.0.0.5
20         23         10.0.0.28/30     0             Fa0/0      10.0.0.18
21         17         10.0.0.24/30     0             Fa0/0      10.0.0.18
22         18         10.0.0.0/30      0             Fa0/0      10.0.0.18
23         Pop Label  10.0.0.12/30     0             Fa0/0      10.0.0.18
24         Pop Label  10.0.0.8/30      0             Fa0/0      10.0.0.18
25         Pop Label  10.0.0.20/30     0             Fa0/1      10.0.0.5
           Pop Label  10.0.0.20/30     0             Fa0/0      10.0.0.18
```

Figure 1.7: Chicago router MPLS forwarding-table



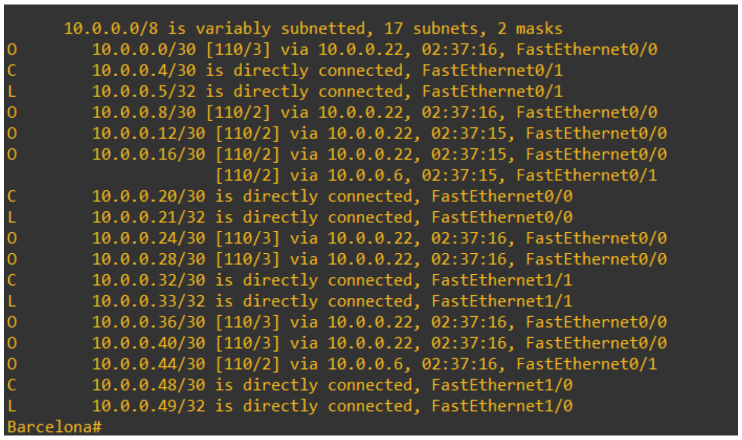
```
      10.0.0.0/8 is variably subnetted, 17 subnets, 2 masks
C        10.0.0.0/30 is directly connected, FastEthernet0/1
L        10.0.0.2/32 is directly connected, FastEthernet0/1
O        10.0.0.4/30 [110/3] via 10.0.0.10, 02:28:46, FastEthernet0/0
C        10.0.0.8/30 is directly connected, FastEthernet0/0
L        10.0.0.9/32 is directly connected, FastEthernet0/0
O        10.0.0.12/30 [110/2] via 10.0.0.10, 02:28:56, FastEthernet0/0
                      [110/2] via 10.0.0.1, 02:28:56, FastEthernet0/1
O        10.0.0.16/30 [110/2] via 10.0.0.10, 02:28:46, FastEthernet0/0
O        10.0.0.20/30 [110/2] via 10.0.0.10, 02:28:46, FastEthernet0/0
O        10.0.0.24/30 [110/2] via 10.0.0.1, 02:28:56, FastEthernet0/1
C        10.0.0.28/30 is directly connected, FastEthernet1/1
L        10.0.0.29/32 is directly connected, FastEthernet1/1
O        10.0.0.32/30 [110/3] via 10.0.0.10, 02:28:46, FastEthernet0/0
O        10.0.0.36/30 [110/2] via 10.0.0.1, 02:28:56, FastEthernet0/1
C        10.0.0.40/30 is directly connected, FastEthernet1/0
L        10.0.0.41/32 is directly connected, FastEthernet1/0
O        10.0.0.44/30 [110/3] via 10.0.0.10, 02:28:46, FastEthernet0/0
O        10.0.0.48/30 [110/3] via 10.0.0.10, 02:28:46, FastEthernet0/0
Porto#
```

Figure 1.8: Porto router learned routes



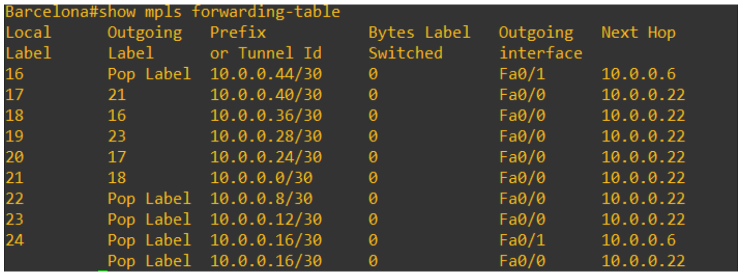
```
Porto#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.0.0.36/30     0             Fa0/1      10.0.0.1
17         Pop Label  10.0.0.24/30     0             Fa0/1      10.0.0.1
18         Pop Label  10.0.0.20/30     0             Fa0/0      10.0.0.10
19         Pop Label  10.0.0.16/30     0             Fa0/0      10.0.0.10
20         Pop Label  10.0.0.12/30     0             Fa0/1      10.0.0.1
           Pop Label  10.0.0.12/30     0             Fa0/0      10.0.0.10
21         19         10.0.0.48/30     0             Fa0/0      10.0.0.10
22         20         10.0.0.44/30     0             Fa0/0      10.0.0.10
23         22         10.0.0.32/30     0             Fa0/0      10.0.0.10
24         24         10.0.0.4/30      0             Fa0/0      10.0.0.10
```

Figure 1.9: Porto router MPLS forwarding-table



```
      10.0.0.0/8 is variably subnetted, 17 subnets, 2 masks
O        10.0.0.0/30 [110/3] via 10.0.0.22, 02:37:16, FastEthernet0/0
C        10.0.0.4/30 is directly connected, FastEthernet0/1
L        10.0.0.5/32 is directly connected, FastEthernet0/1
O        10.0.0.8/30 [110/2] via 10.0.0.22, 02:37:16, FastEthernet0/0
O        10.0.0.12/30 [110/2] via 10.0.0.22, 02:37:15, FastEthernet0/0
O        10.0.0.16/30 [110/2] via 10.0.0.22, 02:37:15, FastEthernet0/0
                      [110/2] via 10.0.0.6, 02:37:15, FastEthernet0/1
C        10.0.0.20/30 is directly connected, FastEthernet0/0
L        10.0.0.21/32 is directly connected, FastEthernet0/0
O        10.0.0.24/30 [110/3] via 10.0.0.22, 02:37:16, FastEthernet0/0
O        10.0.0.28/30 [110/3] via 10.0.0.22, 02:37:16, FastEthernet0/0
C        10.0.0.32/30 is directly connected, FastEthernet1/1
L        10.0.0.33/32 is directly connected, FastEthernet1/1
O        10.0.0.36/30 [110/3] via 10.0.0.22, 02:37:16, FastEthernet0/0
O        10.0.0.40/30 [110/3] via 10.0.0.22, 02:37:16, FastEthernet0/0
O        10.0.0.44/30 [110/2] via 10.0.0.6, 02:37:16, FastEthernet0/1
C        10.0.0.48/30 is directly connected, FastEthernet1/0
L        10.0.0.49/32 is directly connected, FastEthernet1/0
Barcelona#
```

Figure 1.10: Barcelona router learned routes



```
Barcelona#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.0.0.44/30     0             Fa0/1      10.0.0.6
17         21         10.0.0.40/30     0             Fa0/0      10.0.0.22
18         16         10.0.0.36/30     0             Fa0/0      10.0.0.22
19         23         10.0.0.28/30     0             Fa0/0      10.0.0.22
20         17         10.0.0.24/30     0             Fa0/0      10.0.0.22
21         18         10.0.0.0/30      0             Fa0/0      10.0.0.22
22         Pop Label  10.0.0.8/30      0             Fa0/0      10.0.0.22
23         Pop Label  10.0.0.12/30     0             Fa0/0      10.0.0.22
24         Pop Label  10.0.0.16/30     0             Fa0/1      10.0.0.6
           Pop Label  10.0.0.16/30     0             Fa0/0      10.0.0.22
```

Figure 1.11: Barcelona router MPLS forwarding-table

Note: To obtain these tables we only turned on routers from Lisbon,Porto,Chicago, Barcelona and Core1, the rest of the network is off.

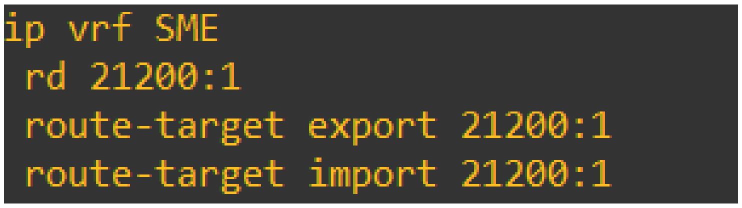
Chapter 2

SME1, SME2, and SME3 association private network

2.1 Implementation

For this part of the project we implemented a VRF which is a virtual routing and forwarding table for each router that connects to the different clients of the SME industrial association. In this way we can segment all the traffic that is generated by these clients.

All routers responsible for forwarding SME traffic have very similar configurations, instead of going through each one of them we will only show RL2 configuration.

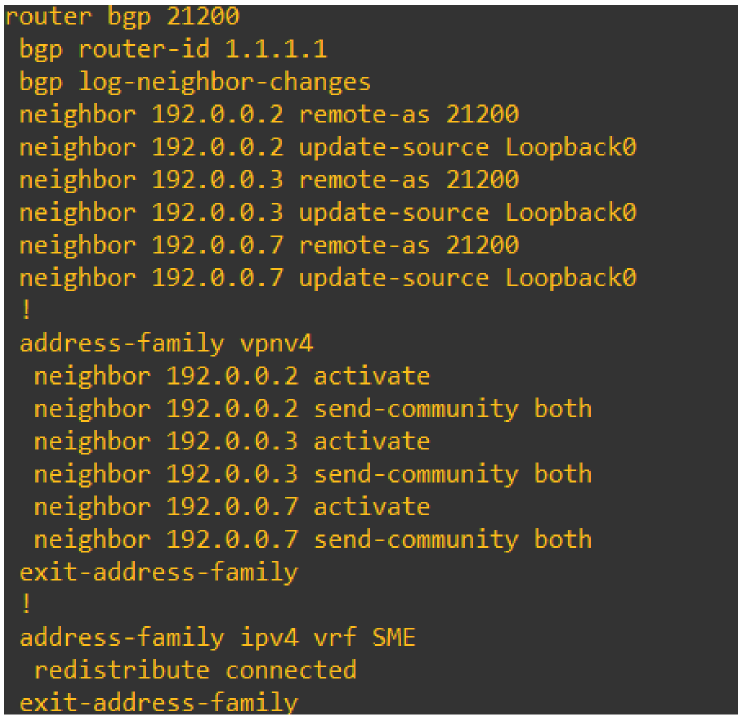


```
ip vrf SME
 rd 21200:1
 route-target export 21200:1
 route-target import 21200:1
```

Figure 2.1: SME VRF instance from RL2

This configuration sets up a VRF named SME with a unique identifier (RD 21200:1) and defines its routing policies:

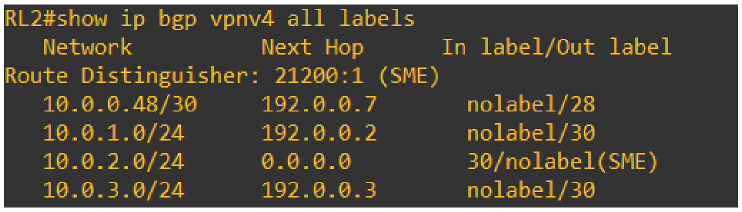
- Routes originating from SME are tagged with RT 21200:1 when advertised (exported).
- The VRF will also accept and use routes tagged with RT 21200:1 (imported).



```
router bgp 21200
 bgp router-id 1.1.1.1
 bgp log-neighbor-changes
 neighbor 192.0.0.2 remote-as 21200
 neighbor 192.0.0.2 update-source Loopback0
 neighbor 192.0.0.3 remote-as 21200
 neighbor 192.0.0.3 update-source Loopback0
 neighbor 192.0.0.7 remote-as 21200
 neighbor 192.0.0.7 update-source Loopback0
 !
 address-family vpnv4
  neighbor 192.0.0.2 activate
  neighbor 192.0.0.2 send-community both
  neighbor 192.0.0.3 activate
  neighbor 192.0.0.3 send-community both
  neighbor 192.0.0.7 activate
  neighbor 192.0.0.7 send-community both
 exit-address-family
 !
 address-family ipv4 vrf SME
  redistribute connected
 exit-address-family
```

Figure 2.2: RL2 configuration

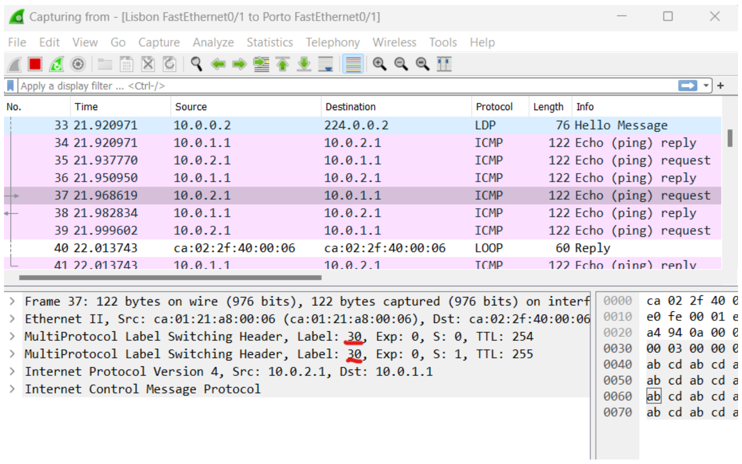
The SME VRF instance is exactly the same for all routers responsible to manage traffic generated from SME clients, what differs from router to router is mainly the BGP neighbors, since RL2 has the address 192.0.0.1 on its loopback interface, this router will create a BGP neighbor relation with all other routers that forward SME traffic, in this specific case RP2 router has the address 192.0.0.2, RB1 has the address 192.0.0.3 and the Barcelona router has the address 192.0.0.7. The routes are learned through BGP and OSPF protocols, MPLS LDP also gives an unique label in order to differentiate traffic generated by SME clients, as seen in the image below.



```
RL2#show ip bgp vpnv4 all labels
   Network          Next Hop      In label/Out label
Route Distinguisher: 21200:1 (SME)
   10.0.0.48/30     192.0.0.7       nolabel/28
   10.0.1.0/24      192.0.0.2       nolabel/30
   10.0.2.0/24      0.0.0.0         30/nolabel(SME)
   10.0.3.0/24      192.0.0.3       nolabel/30
```

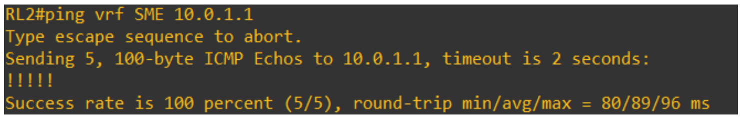
Figure 2.3: Attributed label to SME clients

Performing a ping from client SME2 to SME1 we can see that the packet will have 2 MPLS labels, this is because packets originated from the SME network will be given one label and when that packet enters the MPLS core it will be give another label depending on the link where the packet will go through.


Capturing from - [Lisbon FastEthernet0/1 to Porto FastEthernet0/1]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

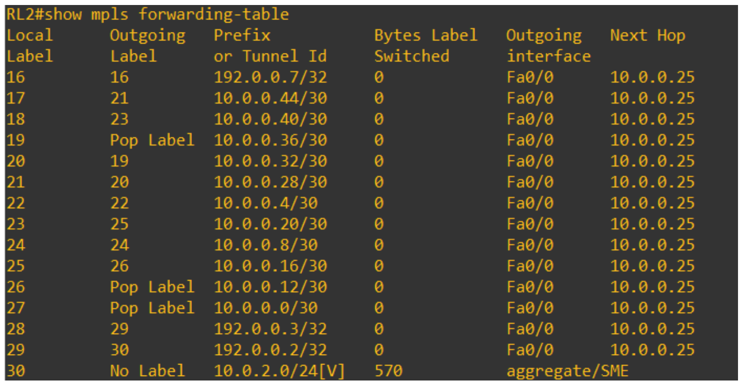
```
No.   Time       Source             Destination        Protocol  Length Info
   33 21.920971  10.0.0.2           224.0.0.2          LDP         76 Hello Message
   34 21.920971  10.0.1.1           10.0.2.1           ICMP       122 Echo (ping) reply
   35 21.937770  10.0.2.1           10.0.1.1           ICMP       122 Echo (ping) request
   36 21.950950  10.0.1.1           10.0.2.1           ICMP       122 Echo (ping) reply
   37 21.968619  10.0.2.1           10.0.1.1           ICMP       122 Echo (ping) request
   38 21.982834  10.0.1.1           10.0.2.1           ICMP       122 Echo (ping) reply
   39 21.999602  10.0.2.1           10.0.1.1           ICMP       122 Echo (ping) request
   40 22.013743  ca:02:2f:40:00:06  ca:02:2f:40:00:06  LOOP        60 Reply
   41 22.013743  10.0.1.1           10.0.2.1           ICMP       122 Echo (ping) reply

> Frame 37: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interf
> Ethernet II, Src: ca:01:21:a8:00:06 (ca:01:21:a8:00:06), Dst: ca:02:2f:40:00:06
> MultiProtocol Label Switching Header, Label: 30, Exp: 0, S: 0, TTL: 254
> MultiProtocol Label Switching Header, Label: 30, Exp: 0, S: 1, TTL: 255
> Internet Protocol Version 4, Src: 10.0.2.1, Dst: 10.0.1.1
> Internet Control Message Protocol

0000  ca 02 2f 40 0
0010  e0 fe 00 01 e
0020  a4 94 0a 00 0
0030  00 03 00 00 0
0040  ab cd ab cd a
0050  ab cd ab cd a
0060  ab cd ab cd a
0070  ab cd ab cd a
```

Figure 2.4: Packet capture that shows ping performed from SME2 to SME1



```
RL2#ping vrf SME 10.0.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 80/89/96 ms
```

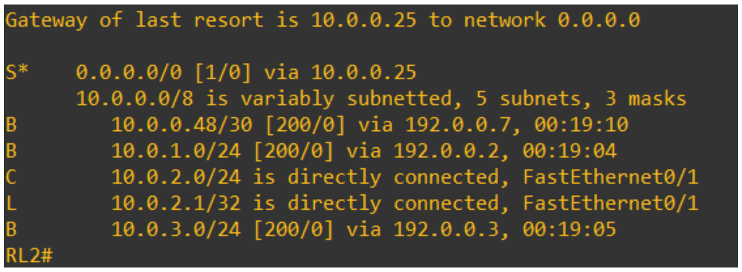
Figure 2.5: SME2 to SME1 connectivity



```
RL2#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         16         192.0.0.7/32     0             Fa0/0      10.0.0.25
17         21         10.0.0.44/30     0             Fa0/0      10.0.0.25
18         23         10.0.0.40/30     0             Fa0/0      10.0.0.25
19         Pop Label  10.0.0.36/30     0             Fa0/0      10.0.0.25
20         19         10.0.0.32/30     0             Fa0/0      10.0.0.25
21         20         10.0.0.28/30     0             Fa0/0      10.0.0.25
22         22         10.0.0.4/30      0             Fa0/0      10.0.0.25
23         25         10.0.0.20/30     0             Fa0/0      10.0.0.25
24         24         10.0.0.8/30      0             Fa0/0      10.0.0.25
25         26         10.0.0.16/30     0             Fa0/0      10.0.0.25
26         Pop Label  10.0.0.12/30     0             Fa0/0      10.0.0.25
27         Pop Label  10.0.0.0/30      0             Fa0/0      10.0.0.25
28         29         192.0.0.3/32     0             Fa0/0      10.0.0.25
29         30         192.0.0.2/32     0             Fa0/0      10.0.0.25
30         No Label   10.0.2.0/24[V]   570           aggregate/SME
```

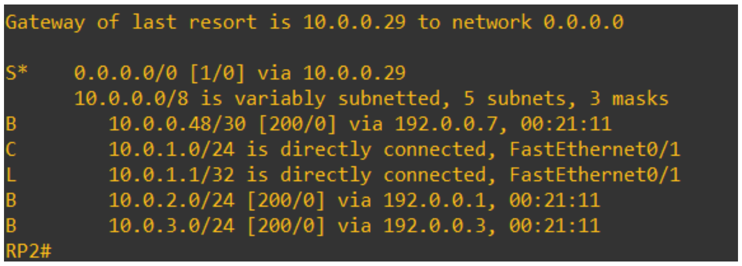
Figure 2.6: RL2 MPLS forwarding-table

Here are the VRF tables, which demonstrate that we can successfully ping between the different PoPs of the SME clients.



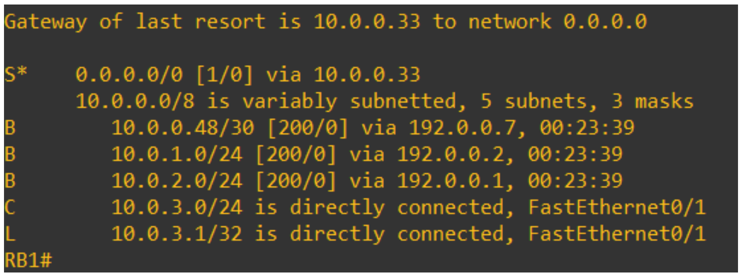
```
Gateway of last resort is 10.0.0.25 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 10.0.0.25
      10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
B        10.0.0.48/30 [200/0] via 192.0.0.7, 00:19:10
B        10.0.1.0/24 [200/0] via 192.0.0.2, 00:19:04
C        10.0.2.0/24 is directly connected, FastEthernet0/1
L        10.0.2.1/32 is directly connected, FastEthernet0/1
B        10.0.3.0/24 [200/0] via 192.0.0.3, 00:19:05
RL2#
```

Figure 2.7: RL2 VRF table



```
Gateway of last resort is 10.0.0.29 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 10.0.0.29
      10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
B        10.0.0.48/30 [200/0] via 192.0.0.7, 00:21:11
C        10.0.1.0/24 is directly connected, FastEthernet0/1
L        10.0.1.1/32 is directly connected, FastEthernet0/1
B        10.0.2.0/24 [200/0] via 192.0.0.1, 00:21:11
B        10.0.3.0/24 [200/0] via 192.0.0.3, 00:21:11
RP2#
```

Figure 2.8: RP2 VRF table



```
Gateway of last resort is 10.0.0.33 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 10.0.0.33
      10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
B        10.0.0.48/30 [200/0] via 192.0.0.7, 00:23:39
B        10.0.1.0/24 [200/0] via 192.0.0.2, 00:23:39
B        10.0.2.0/24 [200/0] via 192.0.0.1, 00:23:39
C        10.0.3.0/24 is directly connected, FastEthernet0/1
L        10.0.3.1/32 is directly connected, FastEthernet0/1
RB1#
```

Figure 2.9: RB1 VRF table

Chapter 3

SME1, SME2, and SME3 CDN advanced service routing

3.1 Implementation

For this part of the project we implemented a conditional CDN service, that will answer to DNS queries based on the IP address of the client, for instance, clients SME1, SME2 and SME3 will have different service providers because these clients are on different networks, a server that is suitable for a client might no be suitable for the other, that is why it is important to define different "views" in the master server DNS service (which is located in the Barcelona PoP) in order to make the transmission of the service more efficient.

To start with, we defined a view in the file "/etc/bind/named.conf.local" of the master server for each SME client, in this way the bind server is able to handle the queries of each client in a specific way based on their IP address. If the client's network matches any view of the file, the server accesses that view where there is a database that says the IP address of the master server which answers to the queries made by clients, and there is also the IP address of the best service provider for the specific client that performs the query.

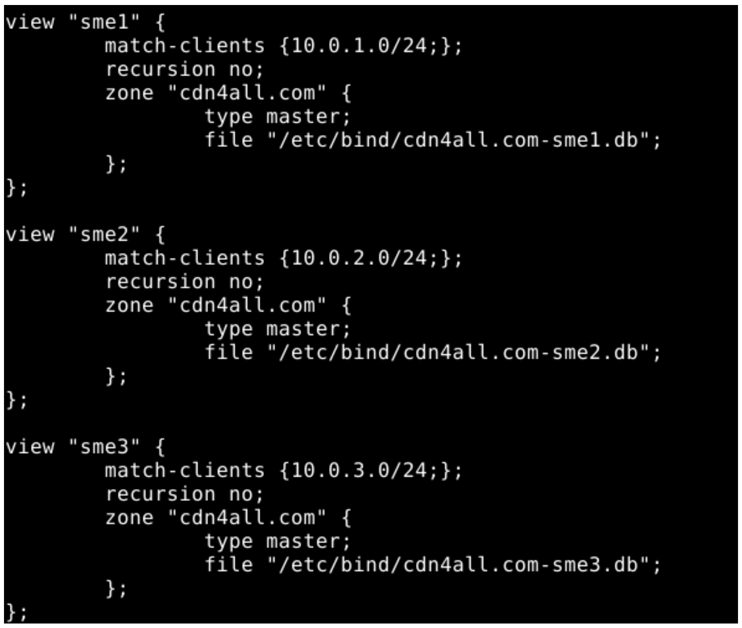

```
view "sme1" {
        match-clients {10.0.1.0/24;};
        recursion no;
        zone "cdn4all.com" {
                type master;
                file "/etc/bind/cdn4all.com-sme1.db";
        };
};

view "sme2" {
        match-clients {10.0.2.0/24;};
        recursion no;
        zone "cdn4all.com" {
                type master;
                file "/etc/bind/cdn4all.com-sme2.db";
        };
};

view "sme3" {
        match-clients {10.0.3.0/24;};
        recursion no;
        zone "cdn4all.com" {
                type master;
                file "/etc/bind/cdn4all.com-sme3.db";
        };
};
```

Figure 3.1: Created views for each SME client

For example, the "sme1" view targets all clients within the 10.0.1.0/24 network. This view then accesses the database "cdn4all.com-sme1.db", which provides both the IP address of the master server and the most suitable service provider for the client.

The databases for each one of the SME clients are the following:

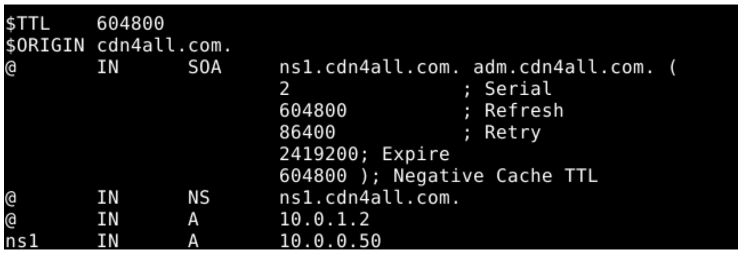
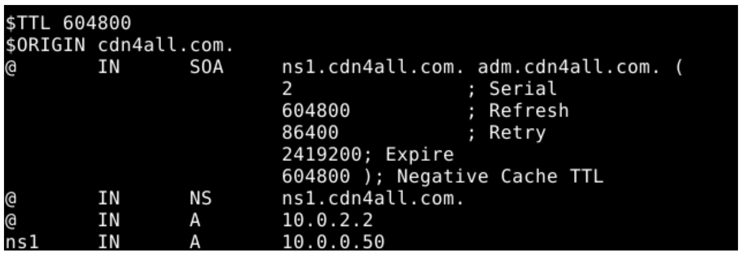

```
$TTL    604800
$ORIGIN cdn4all.com.
@       IN      SOA     ns1.cdn4all.com. adm.cdn4all.com. (
                        2               ; Serial
                        604800          ; Refresh
                        86400           ; Retry
                        2419200; Expire
                        604800 ); Negative Cache TTL
@       IN      NS      ns1.cdn4all.com.
@       IN      A       10.0.1.2
ns1     IN      A       10.0.0.50
```

Figure 3.2: SME1 client database

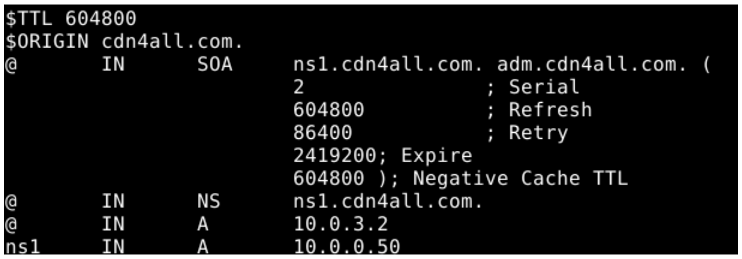
- **Service Provider:** 10.0.1.2
- **Master Server:** 10.0.0.50



```
$TTL 604800
$ORIGIN cdn4all.com.
@       IN      SOA     ns1.cdn4all.com. adm.cdn4all.com. (
                        2               ; Serial
                        604800          ; Refresh
                        86400           ; Retry
                        2419200; Expire
                        604800 ); Negative Cache TTL
@       IN      NS      ns1.cdn4all.com.
@       IN      A       10.0.2.2
ns1     IN      A       10.0.0.50
```

Figure 3.3: SME2 client database

- **Service Provider:** 10.0.2.2
- **Master Server:** 10.0.0.50

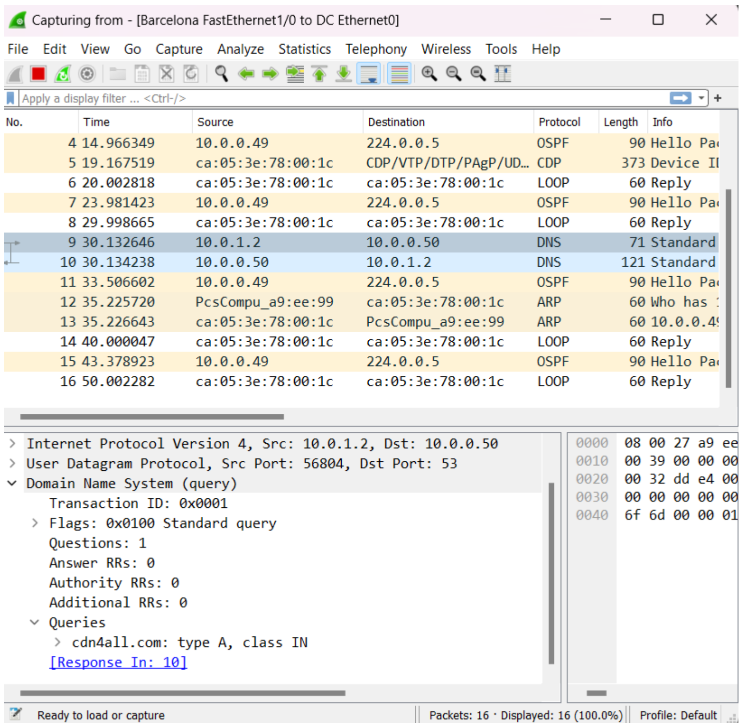


```
$TTL 604800
$ORIGIN cdn4all.com.
@       IN      SOA     ns1.cdn4all.com. adm.cdn4all.com. (
                        2               ; Serial
                        604800          ; Refresh
                        86400           ; Retry
                        2419200; Expire
                        604800 ); Negative Cache TTL
@       IN      NS      ns1.cdn4all.com.
@       IN      A       10.0.3.2
ns1     IN      A       10.0.0.50
```

Figure 3.4: SME3 client database

- **Service Provider:** 10.0.3.2
- **Master Server:** 10.0.0.50

By performing a ping from the SME1, SME2 and SME3 networks to the "cdn4all.com" domain, we can confirm that the address is correctly translated, as indicated by the Wireshark captures.



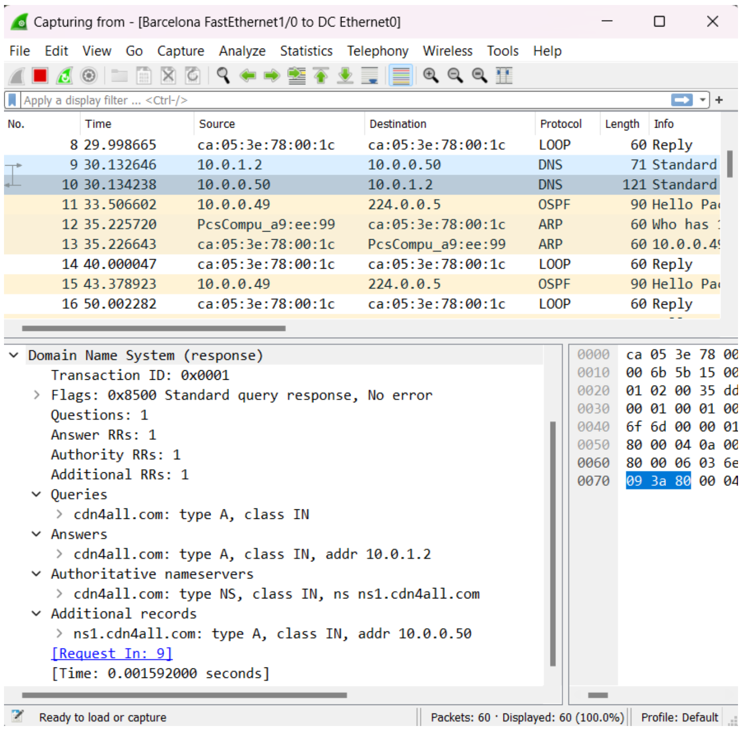
Capturing from - [Barcelona FastEthernet1/0 to DC Ethernet0]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time        Source              Destination              Protocol  Length Info
   4 14.966349   10.0.0.49           224.0.0.5                OSPF         90 Hello Pa
   5 19.167519   ca:05:3e:78:00:1c   CDP/VTP/DTP/PAgP/UD...   CDP         373 Device I
   6 20.002818   ca:05:3e:78:00:1c   ca:05:3e:78:00:1c        LOOP         60 Reply
   7 23.981423   10.0.0.49           224.0.0.5                OSPF         90 Hello Pa
   8 29.998665   ca:05:3e:78:00:1c   ca:05:3e:78:00:1c        LOOP         60 Reply
   9 30.132646   10.0.1.2            10.0.0.50                DNS          71 Standard
  10 30.134238   10.0.0.50           10.0.1.2                 DNS         121 Standard
  11 33.506602   10.0.0.49           224.0.0.5                OSPF         90 Hello Pa
  12 35.225720   PcsCompu_a9:ee:99   ca:05:3e:78:00:1c        ARP          60 Who has
  13 35.226643   ca:05:3e:78:00:1c   PcsCompu_a9:ee:99        ARP          60 10.0.0.4
  14 40.000047   ca:05:3e:78:00:1c   ca:05:3e:78:00:1c        LOOP         60 Reply
  15 43.378923   10.0.0.49           224.0.0.5                OSPF         90 Hello Pa
  16 50.002282   ca:05:3e:78:00:1c   ca:05:3e:78:00:1c        LOOP         60 Reply
```

```
> Internet Protocol Version 4, Src: 10.0.1.2, Dst: 10.0.0.50
> User Datagram Protocol, Src Port: 56804, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x0001
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > cdn4all.com: type A, class IN
    [Response In: 10]
```

```
0000  08 00 27 a9 ee
0010  00 39 00 00 00
0020  00 32 dd e4 00
0030  00 00 00 00 00
0040  6f 6d 00 00 01
```

Figure 3.5: SME1 client query


Capturing from - [Barcelona FastEthernet1/0 to DC Ethernet0]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Ready to load or capture
Packets: 60 · Displayed: 60 (100.0%)
Profile: Default

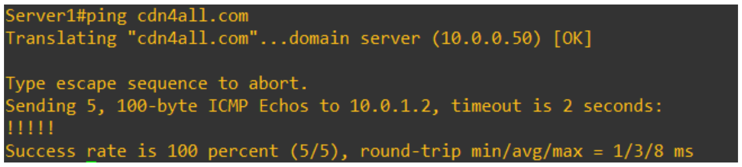
```
No.  Time       Source             Destination        Protocol Length Info
   8 29.998665  ca:05:3e:78:00:1c  ca:05:3e:78:00:1c  LOOP       60 Reply
   9 30.132646  10.0.1.2           10.0.0.50          DNS        71 Standard
  10 30.134238  10.0.0.50          10.0.1.2           DNS       121 Standard
  11 33.506602  10.0.0.49          224.0.0.5          OSPF       90 Hello Pa
  12 35.225720  PcsCompu_a9:ee:99  ca:05:3e:78:00:1c  ARP        60 Who has
  13 35.226643  ca:05:3e:78:00:1c  PcsCompu_a9:ee:99  ARP        60 10.0.0.4
  14 40.000047  ca:05:3e:78:00:1c  ca:05:3e:78:00:1c  LOOP       60 Reply
  15 43.378923  10.0.0.49          224.0.0.5          OSPF       90 Hello Pa
  16 50.002282  ca:05:3e:78:00:1c  ca:05:3e:78:00:1c  LOOP       60 Reply
```

```
Domain Name System (response)
    Transaction ID: 0x0001
  > Flags: 0x8500 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 1
    Queries
      > cdn4all.com: type A, class IN
    Answers
      > cdn4all.com: type A, class IN, addr 10.0.1.2
    Authoritative nameservers
      > cdn4all.com: type NS, class IN, ns ns1.cdn4all.com
    Additional records
      > ns1.cdn4all.com: type A, class IN, addr 10.0.0.50
    [Request In: 9]
    [Time: 0.001592000 seconds]
```

```
0000  ca 05 3e 78 00
0010  00 6b 5b 15 00
0020  01 02 00 35 dd
0030  00 01 00 01 00
0040  6f 6d 00 00 01
0050  80 00 04 0a 00
0060  80 00 06 03 6e
0070  09 3a 80 00 04
```

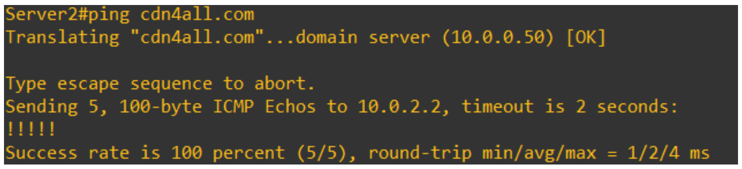
Figure 3.6: Master Server response

Note: As observed, the client requesting the service is the service provider itself. This approach was chosen because we decided to use the service provider as a client within the network instead of adding additional routers to represent clients. This decision was made to prevent significant performance degradation that could result from an excessive number of routers, which would greatly slow down our system.



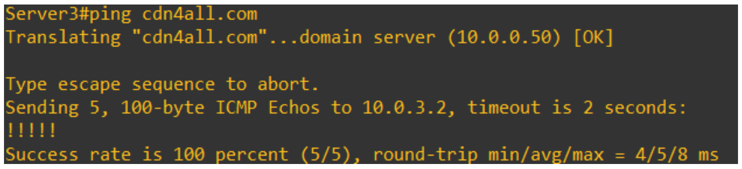
```
Server1#ping cdn4all.com
Translating "cdn4all.com"...domain server (10.0.0.50) [OK]

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/8 ms
```

Figure 3.7: Successful ping from SME1 client to "cdn4all.com" domain



```
Server2#ping cdn4all.com
Translating "cdn4all.com"...domain server (10.0.0.50) [OK]

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Figure 3.8: Successful ping from SME2 client to "cdn4all.com" domain



```
Server3#ping cdn4all.com
Translating "cdn4all.com"...domain server (10.0.0.50) [OK]

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.3.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/8 ms
```

Figure 3.9: Successful ping from SME3 client to "cdn4all.com" domain

Chapter 4

Client LA Layer 2 private network

4.1 Assigned IPs to VPCs

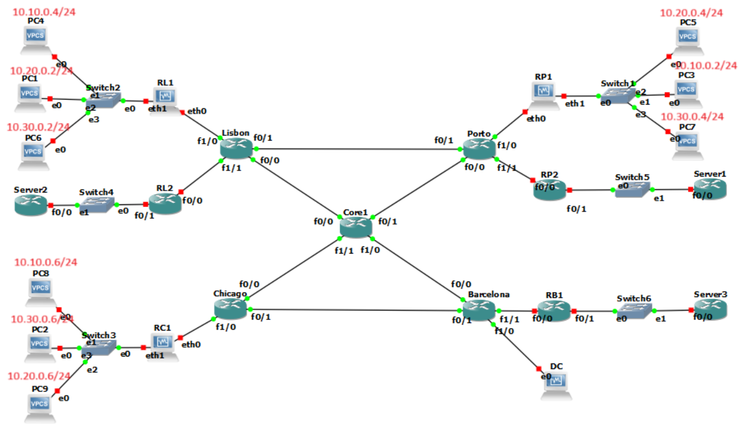
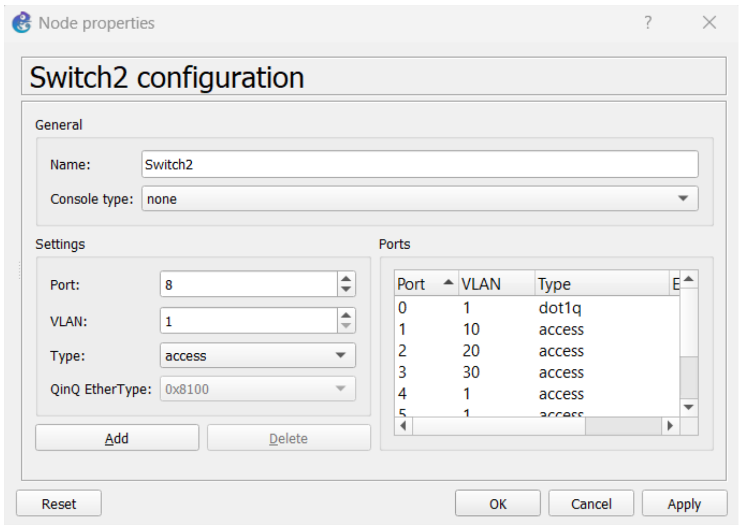


Figure 4.1: Network diagram with VPCs

4.2 Implementation

For this part of the project we utilized the protocol L2VPN/EVPN with VXLAN transport to obtain inter-connectivity across VLANs in different points of presence of the network. To segment the different VLANs we utilized cisco switches and VyOS routers.

The switch configuration is basically the same across all different PoPs:


Switch2 configuration

General

Name:	Switch2
Console type:	none

Settings

Port:	8
VLAN:	1
Type:	access
QinQ EtherType:	0x8100

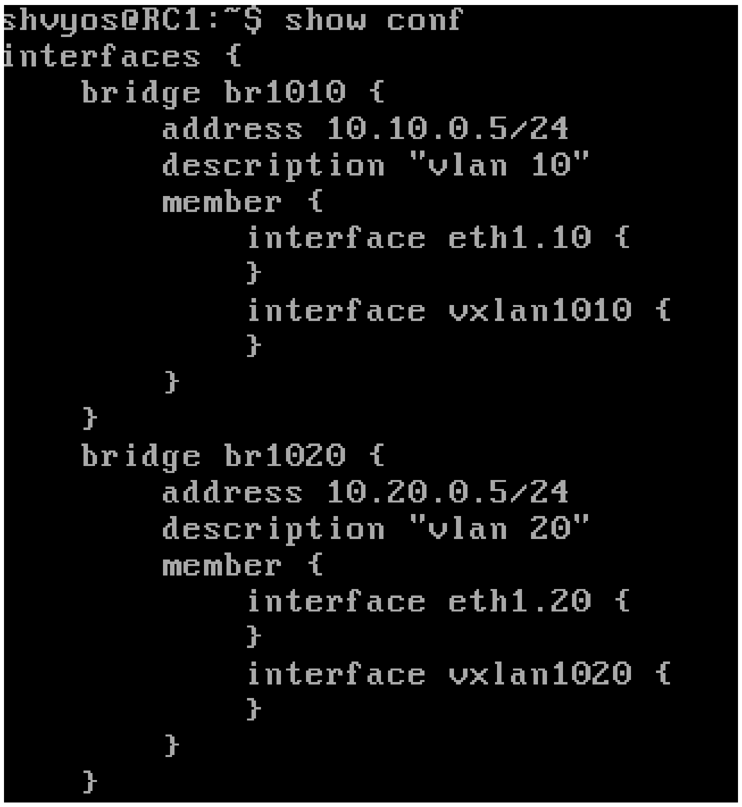
Ports

Port	VLAN	Type
0	1	dot1q
1	10	access
2	20	access
3	30	access
4	1	access
5	1	access

Figure 4.2: Switch2 configuration

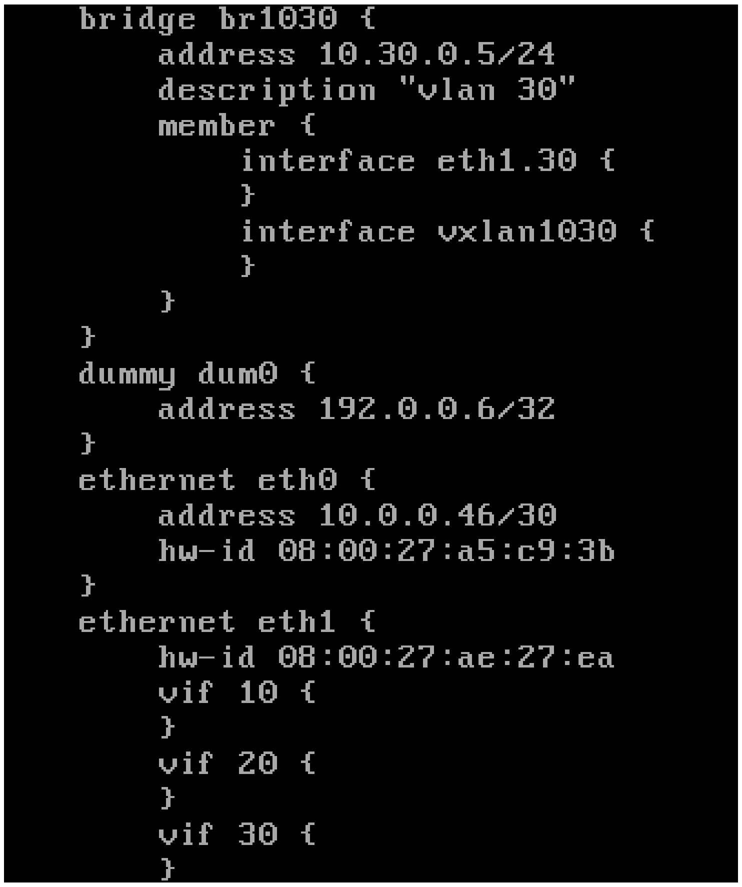
We configured an access port for each VLAN (10, 20, and 30) to properly segment traffic originating from different VLANs. Additionally, we set up a trunk port connecting to the VyOS router, which allows traffic from all VLANs to pass through. This configuration enables routing of VLAN traffic to other Points of Presence (PoPs).

Moreover, we configured a bridge interface on the VyOS router for each VLAN, enabling logical network segmentation with its own MAC address space to handle Layer 2 forwarding decisions. To match traffic from each VLAN, we created three virtual interfaces and assigned a unique VXLAN Network Identifier (VNI) to each VXLAN. This setup allows us to distinguish between virtualized Layer 2 networks while utilizing a shared Layer 3 infrastructure.



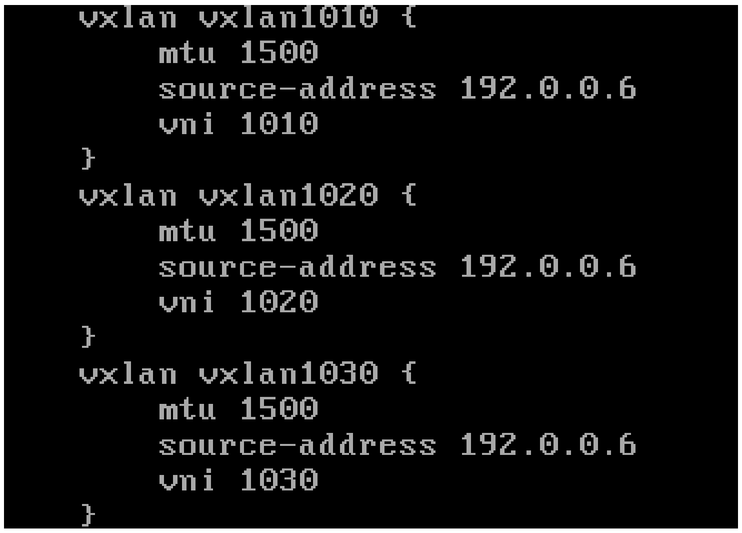
```
shvyos@RC1:~$ show conf
interfaces {
    bridge br1010 {
        address 10.10.0.5/24
        description "vlan 10"
        member {
            interface eth1.10 {
            }
            interface vxlan1010 {
            }
        }
    }
    bridge br1020 {
        address 10.20.0.5/24
        description "vlan 20"
        member {
            interface eth1.20 {
            }
            interface vxlan1020 {
            }
        }
    }
```

Figure 4.3: RC1 Bridge 1010 and Bridge 1020



```
    bridge br1030 {
        address 10.30.0.5/24
        description "vlan 30"
        member {
            interface eth1.30 {
            }
            interface vxlan1030 {
            }
        }
    }
    dummy dum0 {
        address 192.0.0.6/32
    }
    ethernet eth0 {
        address 10.0.0.46/30
        hw-id 08:00:27:a5:c9:3b
    }
    ethernet eth1 {
        hw-id 08:00:27:ae:27:ea
        vif 10 {
        }
        vif 20 {
        }
        vif 30 {
        }
```

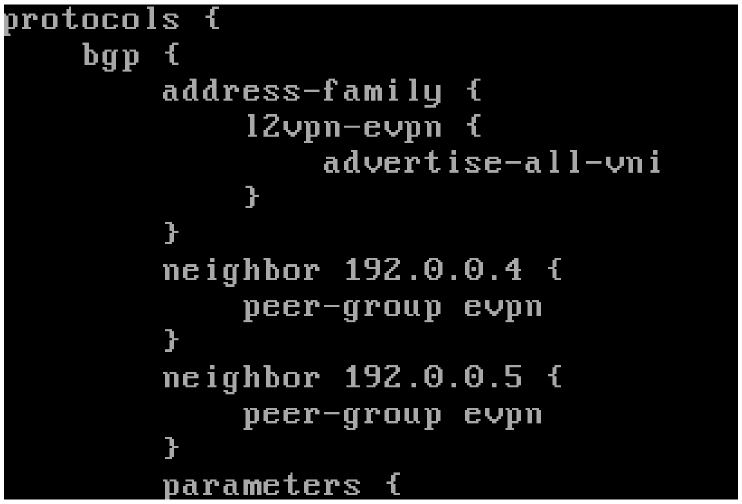
Figure 4.4: RC1 Bridge 1030 and VIFs



```
    vxlan vxlan1010 {
        mtu 1500
        source-address 192.0.0.6
        vni 1010
    }
    vxlan vxlan1020 {
        mtu 1500
        source-address 192.0.0.6
        vni 1020
    }
    vxlan vxlan1030 {
        mtu 1500
        source-address 192.0.0.6
        vni 1030
    }
```

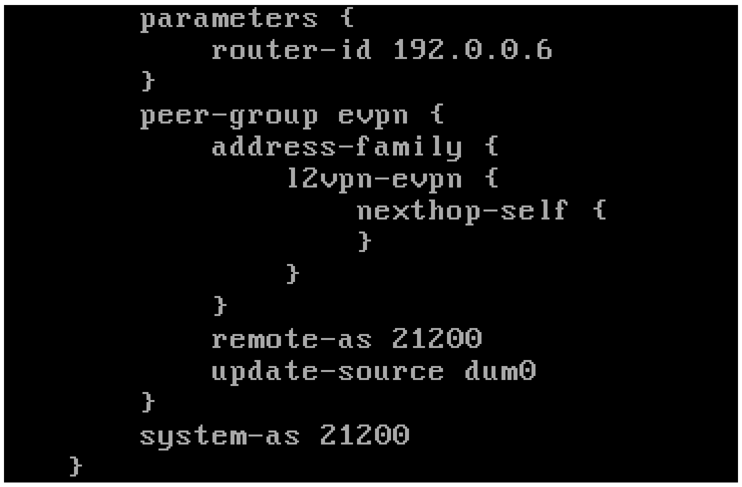
Figure 4.5: RC1 VXLANs configurations

This interconnection of the different VLAN PoPs is possible because of the established BGP mesh with other VyOS routers, for BGP peering relations, we used the Loopback interfaces of each router since they are less likely to go down, we assigned the IP **192.0.0.6** to router **RC1**, **192.0.0.5** to router **RP1** and IP **192.0.0.4** to router **RL1**. The VNIs will then be advertised through BGP allowing other PoPs to be aware of the presence of the same VLANs in different networks. In this way we can use the protocol EVPN to transport Layer 2 traffic over a Layer 3 infrastructure.



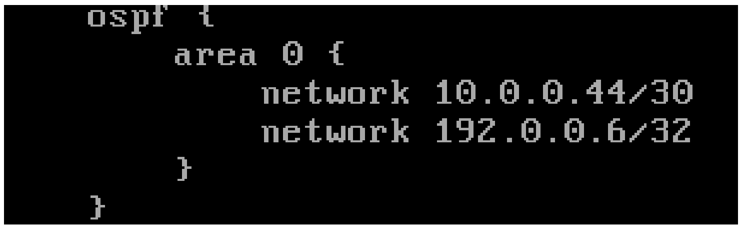
```
protocols {
    bgp {
        address-family {
            l2vpn-evpn {
                advertise-all-vni
            }
        }
        neighbor 192.0.0.4 {
            peer-group evpn
        }
        neighbor 192.0.0.5 {
            peer-group evpn
        }
        parameters {
```

Figure 4.6: RC1 BGP neighbors



```
    parameters {
        router-id 192.0.0.6
    }
    peer-group evpn {
        address-family {
            l2vpn-evpn {
                nexthop-self {
                }
            }
        }
        remote-as 21200
        update-source dum0
    }
    system-as 21200
}
```

Figure 4.7: RC1 L2VPN/EVPN



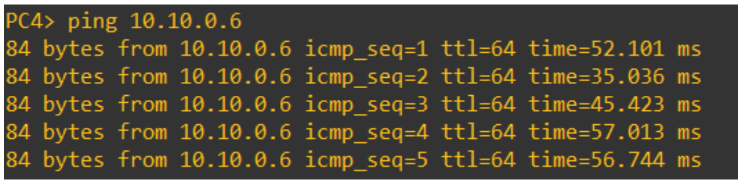
```
ospf {
    area 0 {
        network 10.0.0.44/30
        network 192.0.0.6/32
    }
}
```

Figure 4.8: RC1 OSPF Process

The IP **10.0.0.44** corresponds to the network between the RC1 VyOS router and the Chicago router, the **192.0.0.6** IP is the address of the Loopback interface of the router.

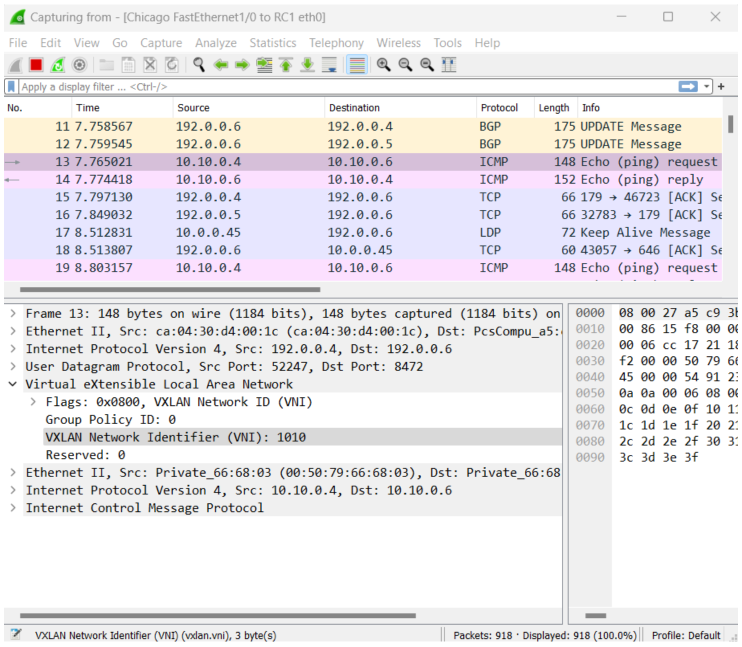
Note: These configurations were applied to the RC1 router. The configurations on the other VyOS routers are nearly identical, with the only differences being the IP addresses of the Loopback interfaces and the BGP neighbor settings.

When performing a ping from PC4 (10.10.0.4) to PC8 (10.10.0.6), the ping is successful. Additionally, the Wireshark capture confirms that the VNI is being correctly assigned to VXLAN 1010.



```
PC4> ping 10.10.0.6
84 bytes from 10.10.0.6 icmp_seq=1 ttl=64 time=52.101 ms
84 bytes from 10.10.0.6 icmp_seq=2 ttl=64 time=35.036 ms
84 bytes from 10.10.0.6 icmp_seq=3 ttl=64 time=45.423 ms
84 bytes from 10.10.0.6 icmp_seq=4 ttl=64 time=57.013 ms
84 bytes from 10.10.0.6 icmp_seq=5 ttl=64 time=56.744 ms
```

Figure 4.9: PC4 to PC8 ping


Capturing from - [Chicago FastEthernet1/0 to RC1 eth0]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time        Source      Destination  Protocol  Length  Info
   11 7.758567    192.0.0.6   192.0.0.4    BGP          175  UPDATE Message
   12 7.759545    192.0.0.6   192.0.0.5    BGP          175  UPDATE Message
   13 7.765021    10.10.0.4   10.10.0.6    ICMP         148  Echo (ping) request
   14 7.774418    10.10.0.6   10.10.0.4    ICMP         152  Echo (ping) reply
   15 7.797130    192.0.0.4   192.0.0.6    TCP           66  179 → 46723 [ACK] Se
   16 7.849032    192.0.0.5   192.0.0.6    TCP           66  32783 → 179 [ACK] Se
   17 8.512831    10.0.0.45   192.0.0.6    LDP           72  Keep Alive Message
   18 8.513807    192.0.0.6   10.0.0.45    TCP           60  43057 → 646 [ACK] Se
   19 8.803157    10.10.0.4   10.10.0.6    ICMP         148  Echo (ping) request
```

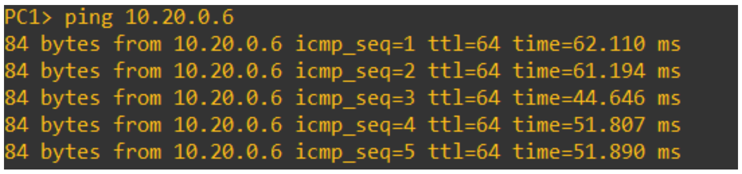
```
> Frame 13: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on
> Ethernet II, Src: ca:04:30:d4:00:1c (ca:04:30:d4:00:1c), Dst: PcsCompu_a5:
> Internet Protocol Version 4, Src: 192.0.0.4, Dst: 192.0.0.6
> User Datagram Protocol, Src Port: 52247, Dst Port: 8472
v Virtual eXtensible Local Area Network
    > Flags: 0x0800, VXLAN Network ID (VNI)
      Group Policy ID: 0
      VXLAN Network Identifier (VNI): 1010
      Reserved: 0
> Ethernet II, Src: Private_66:68:03 (00:50:79:66:68:03), Dst: Private_66:68
> Internet Protocol Version 4, Src: 10.10.0.4, Dst: 10.10.0.6
> Internet Control Message Protocol
```

```
0000  08 00 27 a5 c9
0010  00 86 15 f8 00
0020  00 06 cc 17 21
0030  f2 00 00 50 79
0040  45 00 00 54 91
0050  0a 0a 00 06 08
0060  0c 0d 0e 0f 10
0070  1c 1d 1e 1f 20
0080  2c 2d 2e 2f 30
0090  3c 3d 3e 3f
```

VXLAN Network Identifier (VNI) (vxlan.vni), 3 byte(s) || Packets: 918 · Displayed: 918 (100.0%) || Profile: Default

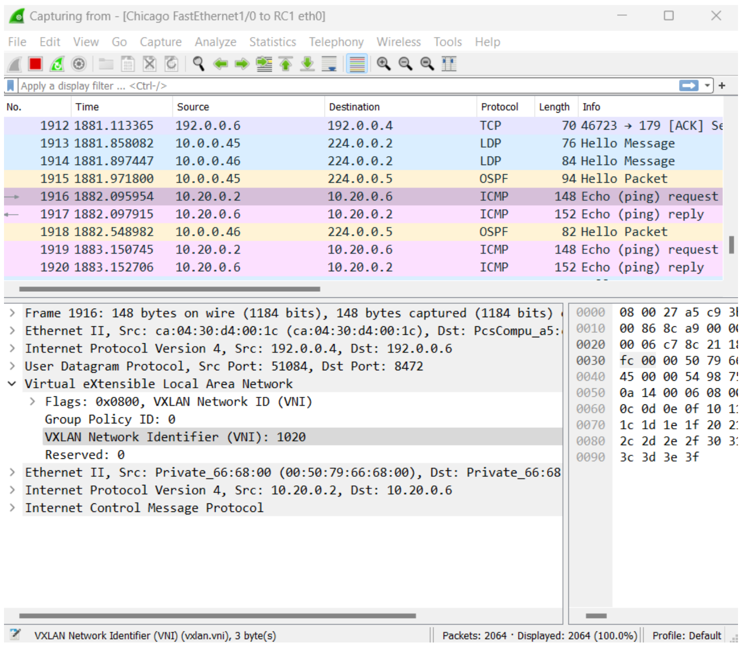
Figure 4.10: Wireshark capture from VLAN 10 ping

Performing a ping from PC1 (10.20.0.2) to PC9 (10.20.0.6) also shows there is connection, and VXLAN 1020 is correctly identified.



```
PC1> ping 10.20.0.6
84 bytes from 10.20.0.6 icmp_seq=1 ttl=64 time=62.110 ms
84 bytes from 10.20.0.6 icmp_seq=2 ttl=64 time=61.194 ms
84 bytes from 10.20.0.6 icmp_seq=3 ttl=64 time=44.646 ms
84 bytes from 10.20.0.6 icmp_seq=4 ttl=64 time=51.807 ms
84 bytes from 10.20.0.6 icmp_seq=5 ttl=64 time=51.890 ms
```

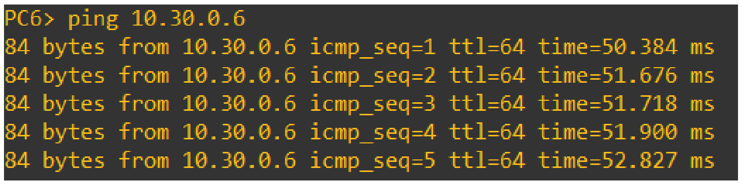
Figure 4.11: PC1 to PC9 ping


Capturing from - [Chicago FastEthernet1/0 to RC1 eth0]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time         Source      Destination  Protocol  Length  Info
 1912 1881.113365  192.0.0.6   192.0.0.4    TCP        70 46723 → 179 [ACK] Se
 1913 1881.858082  10.0.0.45   224.0.0.2    LDP        76 Hello Message
 1914 1881.897447  10.0.0.46   224.0.0.2    LDP        84 Hello Message
 1915 1881.971800  10.0.0.45   224.0.0.5    OSPF       94 Hello Packet
 1916 1882.095954  10.20.0.2   10.20.0.6    ICMP      148 Echo (ping) request
 1917 1882.097915  10.20.0.6   10.20.0.2    ICMP      152 Echo (ping) reply
 1918 1882.548982  10.0.0.46   224.0.0.5    OSPF       82 Hello Packet
 1919 1883.150745  10.20.0.2   10.20.0.6    ICMP      148 Echo (ping) request
 1920 1883.152706  10.20.0.6   10.20.0.2    ICMP      152 Echo (ping) reply

> Frame 1916: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
> Ethernet II, Src: ca:04:30:d4:00:1c (ca:04:30:d4:00:1c), Dst: PcsCompu_a5:
> Internet Protocol Version 4, Src: 192.0.0.4, Dst: 192.0.0.6
> User Datagram Protocol, Src Port: 51084, Dst Port: 8472
v Virtual eXtensible Local Area Network
  > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 1020
    Reserved: 0
> Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: Private_66:68
> Internet Protocol Version 4, Src: 10.20.0.2, Dst: 10.20.0.6
> Internet Control Message Protocol

0000  08 00 27 a5 c9 3
0010  00 86 8c a9 00 0
0020  00 06 c7 8c 21 1
0030  fc 00 00 50 79 6
0040  45 00 00 54 98 7
0050  0a 14 00 06 08 0
0060  0c 0d 0e 0f 10 1
0070  1c 1d 1e 1f 20 2
0080  2c 2d 2e 2f 30 3
0090  3c 3d 3e 3f

VXLAN Network Identifier (VNI) (vxlan.vni), 3 byte(s)   Packets: 2064 · Displayed: 2064 (100.0%)   Profile: Default
```

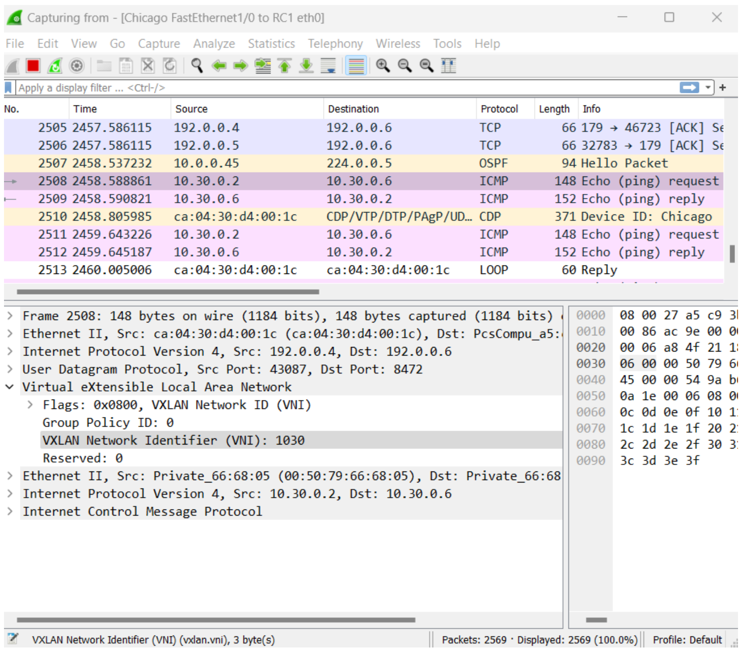
Figure 4.12: Wireshark capture from VLAN 20 ping

Performing a ping from PC6 (10.30.0.2) to PC2 (10.30.0.6) also yields the same results, but with a different VNI.



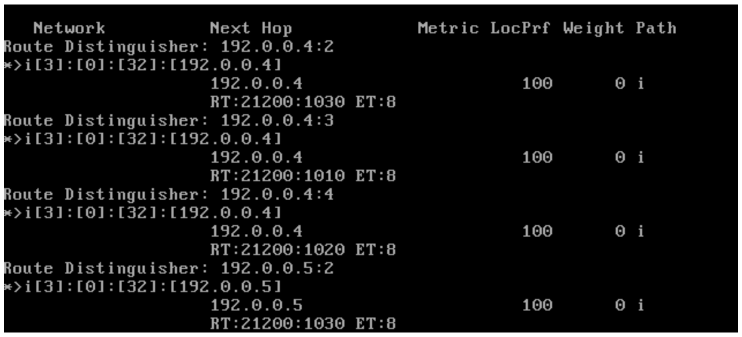
```
PC6> ping 10.30.0.6
84 bytes from 10.30.0.6 icmp_seq=1 ttl=64 time=50.384 ms
84 bytes from 10.30.0.6 icmp_seq=2 ttl=64 time=51.676 ms
84 bytes from 10.30.0.6 icmp_seq=3 ttl=64 time=51.718 ms
84 bytes from 10.30.0.6 icmp_seq=4 ttl=64 time=51.900 ms
84 bytes from 10.30.0.6 icmp_seq=5 ttl=64 time=52.827 ms
```

Figure 4.13: PC6 to PC2 ping



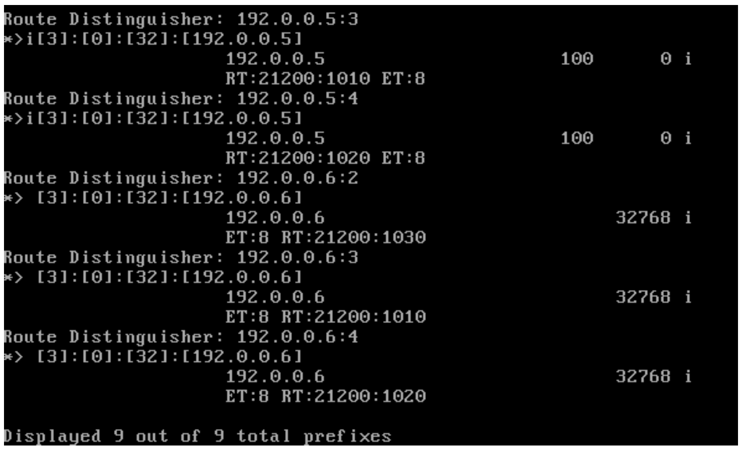
```
Capturing from - [Chicago FastEthernet1/0 to RC1 eth0]
No.    Time          Source              Destination          Protocol  Length Info
  2505 2457.586115   192.0.0.4           192.0.0.6            TCP          66 179 → 46723 [ACK] Se
  2506 2457.586115   192.0.0.5           192.0.0.6            TCP          66 32783 → 179 [ACK] Se
  2507 2458.537232   10.0.0.45           224.0.0.5            OSPF         94 Hello Packet
  2508 2458.588861   10.30.0.2           10.30.0.6            ICMP        148 Echo (ping) request
  2509 2458.590821   10.30.0.6           10.30.0.2            ICMP        152 Echo (ping) reply
  2510 2458.805985   ca:04:30:d4:00:1c   CDP/VTP/DTP/PAgP/UD… CDP         371 Device ID: Chicago
  2511 2459.643226   10.30.0.2           10.30.0.6            ICMP        148 Echo (ping) request
  2512 2459.645187   10.30.0.6           10.30.0.2            ICMP        152 Echo (ping) reply
  2513 2460.005006   ca:04:30:d4:00:1c   ca:04:30:d4:00:1c    LOOP         60 Reply

> Frame 2508: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)     0000  08 00 27 a5 c9 3
> Ethernet II, Src: ca:04:30:d4:00:1c (ca:04:30:d4:00:1c), Dst: PcsCompu_a5:    0010  00 86 ac 9e 00 0
> Internet Protocol Version 4, Src: 192.0.0.4, Dst: 192.0.0.6                   0020  00 06 a8 4f 21 1
> User Datagram Protocol, Src Port: 43087, Dst Port: 8472                       0030  06 00 00 50 79 6
v Virtual eXtensible Local Area Network                                         0040  45 00 00 54 9a b
   > Flags: 0x0800, VXLAN Network ID (VNI)                                      0050  0a 1e 00 06 08 0
     Group Policy ID: 0                                                         0060  0c 0d 0e 0f 10 1
     VXLAN Network Identifier (VNI): 1030                                       0070  1c 1d 1e 1f 20 2
     Reserved: 0                                                                0080  2c 2d 2e 2f 30 3
> Ethernet II, Src: Private_66:68:05 (00:50:79:66:68:05), Dst: Private_66:68   0090  3c 3d 3e 3f
> Internet Protocol Version 4, Src: 10.30.0.2, Dst: 10.30.0.6
> Internet Control Message Protocol

VXLAN Network Identifier (VNI) (vxlan.vni), 3 byte(s)   Packets: 2569 · Displayed: 2569 (100.0%)   Profile: Default
```

Figure 4.14: Wireshark capture from VLAN 30 ping



```
    Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 192.0.0.4:2
*>i[3]:[0]:[32]:[192.0.0.4]
                     192.0.0.4                  100      0 i
                     RT:21200:1030 ET:8
Route Distinguisher: 192.0.0.4:3
*>i[3]:[0]:[32]:[192.0.0.4]
                     192.0.0.4                  100      0 i
                     RT:21200:1010 ET:8
Route Distinguisher: 192.0.0.4:4
*>i[3]:[0]:[32]:[192.0.0.4]
                     192.0.0.4                  100      0 i
                     RT:21200:1020 ET:8
Route Distinguisher: 192.0.0.5:2
*>i[3]:[0]:[32]:[192.0.0.5]
                     192.0.0.5                  100      0 i
                     RT:21200:1030 ET:8
```

Figure 4.15: RC1 EVPN prefixes part 1



```
Route Distinguisher: 192.0.0.5:3
*>i[3]:[0]:[32]:[192.0.0.5]
                    192.0.0.5                100      0 i
                    RT:21200:1010 ET:8
Route Distinguisher: 192.0.0.5:4
*>i[3]:[0]:[32]:[192.0.0.5]
                    192.0.0.5                100      0 i
                    RT:21200:1020 ET:8
Route Distinguisher: 192.0.0.6:2
*> [3]:[0]:[32]:[192.0.0.6]
                    192.0.0.6                     32768 i
                    ET:8 RT:21200:1030
Route Distinguisher: 192.0.0.6:3
*> [3]:[0]:[32]:[192.0.0.6]
                    192.0.0.6                     32768 i
                    ET:8 RT:21200:1010
Route Distinguisher: 192.0.0.6:4
*> [3]:[0]:[32]:[192.0.0.6]
                    192.0.0.6                     32768 i
                    ET:8 RT:21200:1020

Displayed 9 out of 9 total prefixes
```

Figure 4.16: RC1 EVPN prefixes part 2

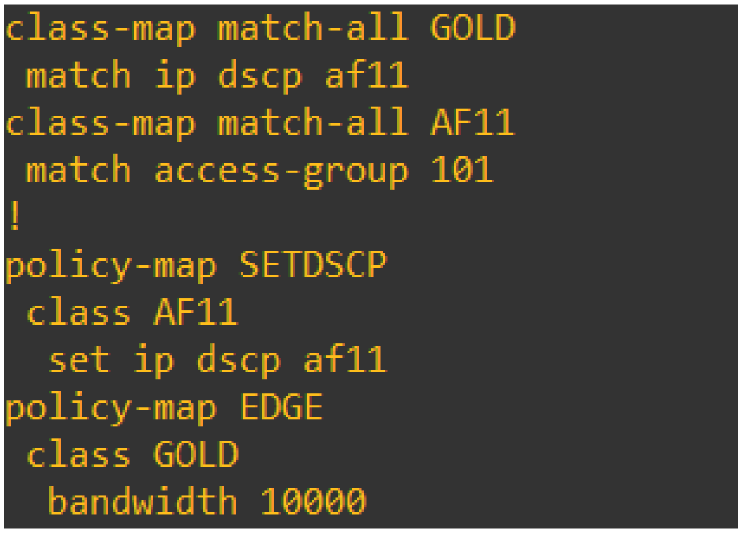
Note: The pings were performed only between the Lisbon and Chicago PoPs. Including the results of pings to the Porto PoP would be redundant, as the connection between the PoPs can be easily demonstrated during the project presentation.

Chapter 5

Client LA Layer 2 traffic differentiation

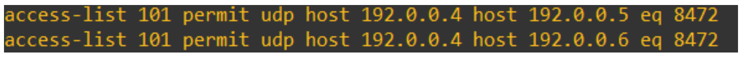
This part of the project aims to differentiate traffic (at the core of the network) generated by client LA by defining an Assured Forwarding policy, that should be guaranteed up to 10 Mbps.

We applied several configurations on routers from Lisbon, Porto, Chicago and Core1, but we will only cover the configurations made on routers from Lisbon and Core1.



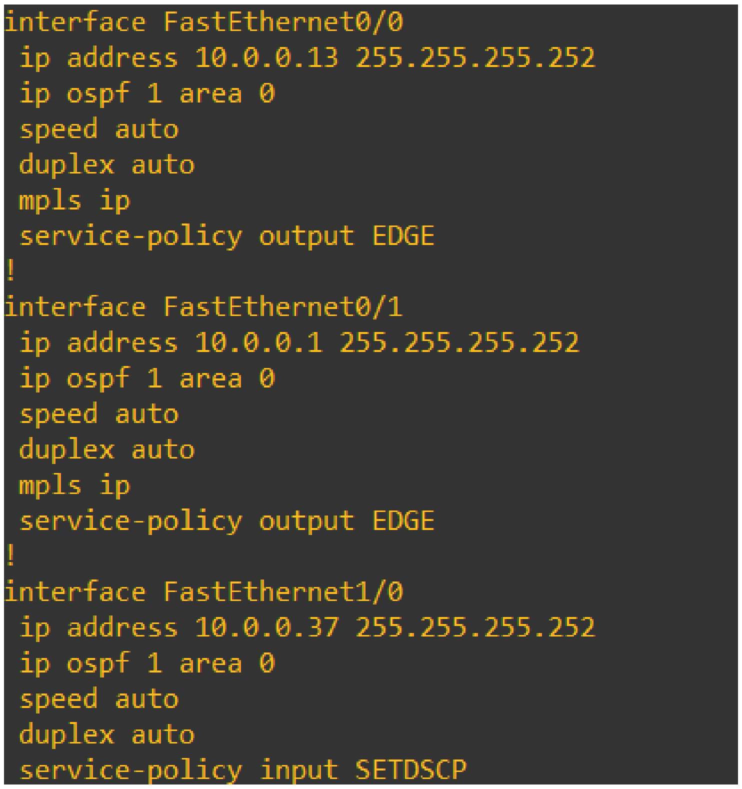
```
class-map match-all GOLD
 match ip dscp af11
class-map match-all AF11
 match access-group 101
!
policy-map SETDSCP
 class AF11
  set ip dscp af11
policy-map EDGE
 class GOLD
  bandwidth 10000
```

Figure 5.1: Lisbon router class and policy maps



```
access-list 101 permit udp host 192.0.0.4 host 192.0.0.5 eq 8472
access-list 101 permit udp host 192.0.0.4 host 192.0.0.6 eq 8472
```

Figure 5.2: Access-list 101 from Lisbon router



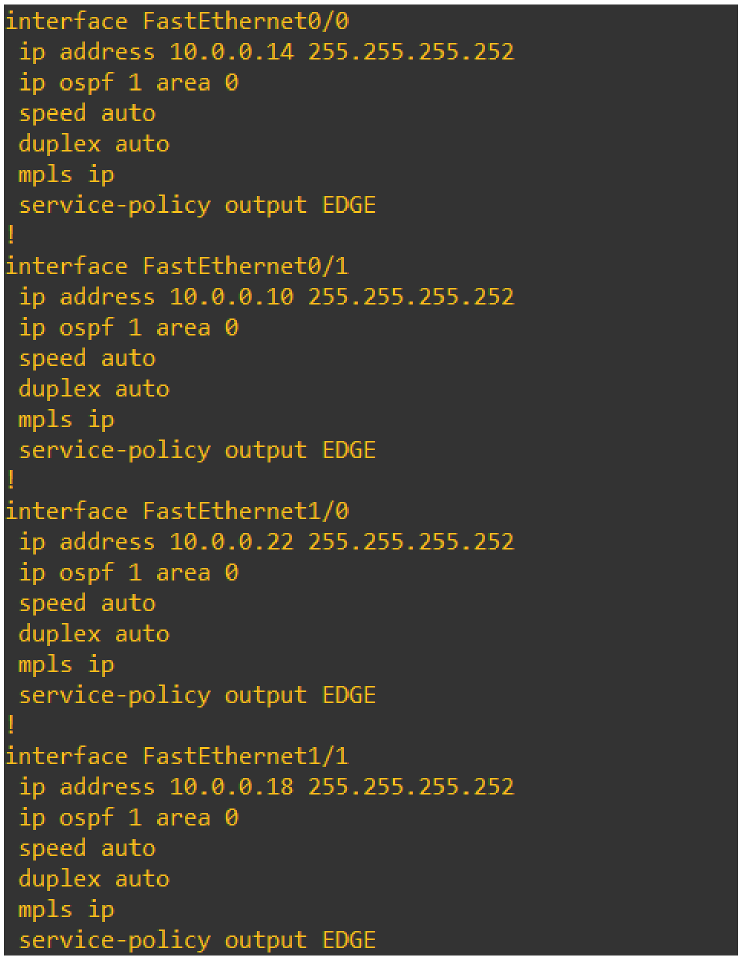
```
interface FastEthernet0/0
 ip address 10.0.0.13 255.255.255.252
 ip ospf 1 area 0
 speed auto
 duplex auto
 mpls ip
 service-policy output EDGE
!
interface FastEthernet0/1
 ip address 10.0.0.1 255.255.255.252
 ip ospf 1 area 0
 speed auto
 duplex auto
 mpls ip
 service-policy output EDGE
!
interface FastEthernet1/0
 ip address 10.0.0.37 255.255.255.252
 ip ospf 1 area 0
 speed auto
 duplex auto
 service-policy input SETDSCP
```

Figure 5.3: Lisbon router interfaces

Basically, all traffic that will be forwarded to port 8472, will belong to the GOLD class which uses assured forwarding (AF11) with low drop probability (dscp 10). The DSCP value should be attributed to the packet in the input interface of the router which in this case is "FastEthernet1/0" and the policy is enforced in the output interfaces which can be either "FastEthernet0/0" or "FastEthernet0/1".

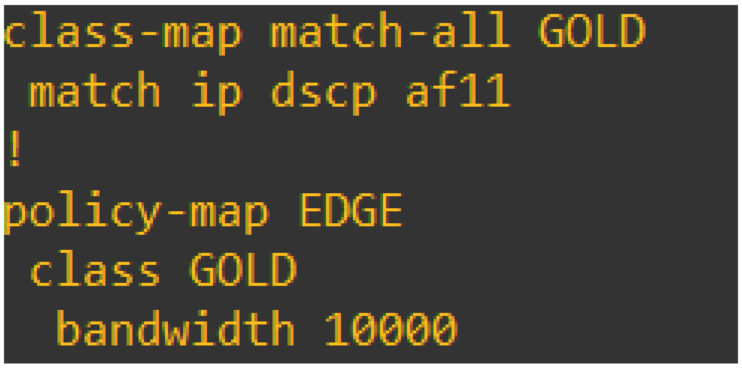
We will exclude the Porto and Chicago routers from our detailed configuration, as their setups are identical to the Lisbon router. The only difference lies in the access control lists (ACLs) defined on each router. For example, the Porto router has two ACLs: one to filter traffic from Porto to Lisbon and another to filter traffic from Porto to Chicago. Similarly, the Chicago router has two ACLs: one for traffic from Chicago to Porto and another for traffic from Chicago to Lisbon.

Moreover, Core1 plays a crucial role in extending the scope of QoS for client LA. This router ensures that QoS continues to be enforced across the network by identifying packets belonging to the GOLD class and allocating up to 10 Mbps on the output interfaces for client LA traffic.



```
interface FastEthernet0/0
 ip address 10.0.0.14 255.255.255.252
 ip ospf 1 area 0
 speed auto
 duplex auto
 mpls ip
 service-policy output EDGE
!
interface FastEthernet0/1
 ip address 10.0.0.10 255.255.255.252
 ip ospf 1 area 0
 speed auto
 duplex auto
 mpls ip
 service-policy output EDGE
!
interface FastEthernet1/0
 ip address 10.0.0.22 255.255.255.252
 ip ospf 1 area 0
 speed auto
 duplex auto
 mpls ip
 service-policy output EDGE
!
interface FastEthernet1/1
 ip address 10.0.0.18 255.255.255.252
 ip ospf 1 area 0
 speed auto
 duplex auto
 mpls ip
 service-policy output EDGE
```

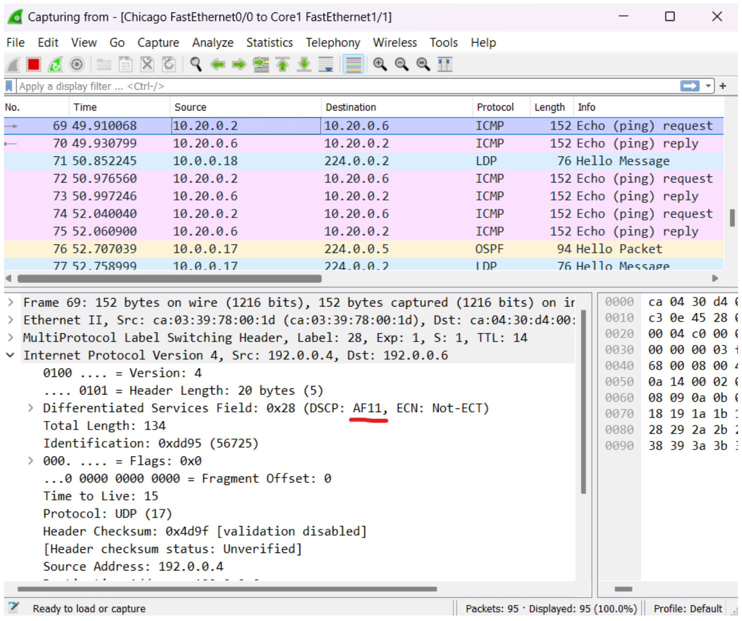
Figure 5.4: Core1 router interfaces



```
class-map match-all GOLD
 match ip dscp af11
!
policy-map EDGE
 class GOLD
  bandwidth 10000
```

Figure 5.5: Core1 router class and policy maps

In conclusion, the following screenshot verifies that the "diffserv" functionality has been correctly implemented for client LA. To test this, we initiated a ping from PC1 (10.20.0.2) to PC9 (10.20.0.6) and captured the traffic using Wireshark on the link between Chicago and Core1.



Capturing from - [Chicago FastEthernet0/0 to Core1 FastEthernet1/1]

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time       Source      Destination  Protocol  Length Info
   69 49.910068  10.20.0.2   10.20.0.6    ICMP      152 Echo (ping) request
   70 49.930799  10.20.0.6   10.20.0.2    ICMP      152 Echo (ping) reply
   71 50.852245  10.0.0.18   224.0.0.2    LDP        76 Hello Message
   72 50.976560  10.20.0.2   10.20.0.6    ICMP      152 Echo (ping) request
   73 50.997246  10.20.0.6   10.20.0.2    ICMP      152 Echo (ping) reply
   74 52.040040  10.20.0.2   10.20.0.6    ICMP      152 Echo (ping) request
   75 52.060900  10.20.0.6   10.20.0.2    ICMP      152 Echo (ping) reply
   76 52.707039  10.0.0.17   224.0.0.5    OSPF       94 Hello Packet
   77 52.758999  10.0.0.17   224.0.0.2    LDP        76 Hello Message
```

```
> Frame 69: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on ir
> Ethernet II, Src: ca:03:39:78:00:1d (ca:03:39:78:00:1d), Dst: ca:04:30:d4:00:
> MultiProtocol Label Switching Header, Label: 28, Exp: 1, S: 1, TTL: 14
v Internet Protocol Version 4, Src: 192.0.0.4, Dst: 192.0.0.6
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x28 (DSCP: AF11, ECN: Not-ECT)
    Total Length: 134
    Identification: 0xdd95 (56725)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 15
    Protocol: UDP (17)
    Header Checksum: 0x4d9f [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.0.0.4
```

```
0000  ca 04 30 d4
0010  c3 0e 45 28
0020  00 04 c0 00
0030  00 00 00 03
0040  68 00 08 00
0050  0a 14 00 02
0060  08 09 0a 0b
0070  18 19 1a 1b
0080  28 29 2a 2b
0090  38 39 3a 3b
```

Ready to load or capture | Packets: 95 · Displayed: 95 (100.0%) | Profile: Default

Figure 5.6: Performed ping from PC1 to PC9

Important Note: During the project demonstration with Professor Paulo Salvador, we had not yet implemented this functionality correctly. However, Professor Paulo provided valuable advice on how to properly implement it. After the demonstration, we successfully applied the necessary changes to ensure the feature was correctly implemented. While we are unsure if this will positively impact our grade, we would greatly appreciate it if it did.

Chapter 6

SME1, SME2, and SME3 private networks traffic reserves

Not implemented.

Chapter 7

Conclusion

In this project, we successfully designed, configured, and tested a CDN network infrastructure capable of meeting the requirements of multiple enterprise clients. Key implementations included the setup of private networks for SME clients, advanced service routing for CDN servers, the deployment of a Layer 2 private network for Client LA and traffic differentiation.

However, due to time management issues, the implementation of traffic reservation for SME clients was not implemented. This feature remains as an opportunity for future development to further enhance the network's performance and client-specific quality of service (QoS).

Overall, the project demonstrates a scalable and robust approach to managing multi-client network architectures while addressing diverse connectivity requirements.

Author's Contribution

Each member contributed equally to the development of this project, therefore, we decided to allocate 50% to each.