Communication Architectures

Project (RECURSO)

João Sousa (103415), João Gaspar (107708)

Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

List of Figures

List of Tables

Contents

1 Introduction

This report details the implementation of a CDN network for CDN4ALL LLC, catering to enterprise clients with distinct connectivity needs. The network design interconnects PoPs in Porto, Lisboa, Barcelona, and Chicago, utilizing MPLS VPNs for SME isolation and VXLAN for Layer 2 extension. Key objectives include traffic guarantees, CDN service efficiency, and SNMP monitoring.

2 Network Topology

This section provides an overview of the entire network infrastructure, highlighting the key components, their interconnections, and the overall architecture. The design ensures seamless connectivity between different Points of Presence (PoPs) while maintaining logical separation for different clients.

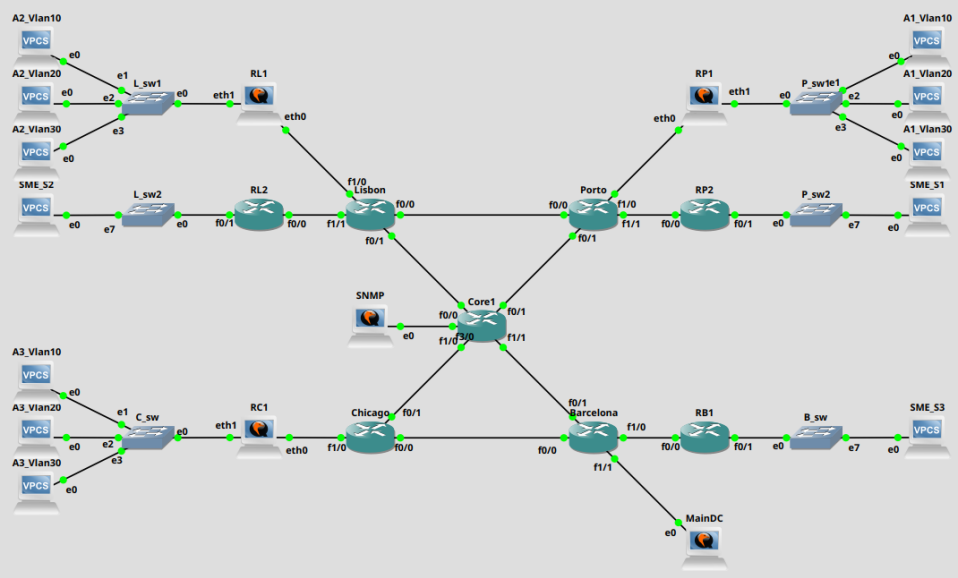

Figure 1: Network Topology with PoPs (Porto, Lisboa, Barcelona, Chicago) connected via Cisco C7200 core routers. SME servers (S1–S3) and LA clusters (A1–A3) are attached to respective PoPs.

3 Basic Assembly and Core Connectivity

The core infrastructure was designed to ensure robust connectivity between all Points of Presence (PoPs) while maintaining logical separation for different clients. The network addressing scheme was carefully planned to allocate IP ranges efficiently, and OSPF was chosen to dynamically propagate

routing information.

3.1 Network Addressing Plan

To support connectivity across multiple locations, the 10.0.0.0/24 network was partitioned into /30 subnets for point-to-point links. This approach ensures that each connection is uniquely addressed while maximizing address space efficiency. The addressing scheme provides 64 available point-to-point connections, with 14 currently allocated in this deployment.

Table 1: Core Network Interface Configuration

Device	**Interface**	**IP Address/Subnet**	**Description**
Core1	f0/0	10.0.0.34/30	Lisbon uplink
	f0/1	10.0.0.82/30	Porto uplink
	f1/0	10.0.0.130/30	Chicago uplink
	f1/1	10.0.0.178/30	Barcelona uplink
	f3/0	10.0.0.62/30	SNMP VM connection
Lisbon	f0/0	10.0.0.97/30	Porto backup link
	f0/1	10.0.0.33/30	Core1 connection
	f1/0	10.0.0.1/30	RL1 access router
	f1/1	10.0.0.17/30	RL2 backup link
Porto	f0/0	10.0.0.98/30	Lisbon backup link
	f0/1	10.0.0.81/30	Core1 connection
	f1/0	10.0.0.49/30	RP1 access router
	f1/1	10.0.0.65/30	RP2 access router
Chicago	f0/0	10.0.0.193/30	Barcelona direct link
	f0/1	10.0.0.129/30	Core1 connection
	f1/0	10.0.0.113/30	RC1 access router
Barcelona	f0/0	10.0.0.194/30	Chicago direct link
	f0/1	10.0.0.177/30	Core1 connection
	f1/0	10.0.0.145/30	RB1 access router
	f1/1	10.0.0.161/30	Main DC connection

3.2 OSPF Implementation

OSPF Area 0 was selected as the interior gateway protocol for the backbone due to its efficiency in propagating routing updates dynamically across multiple routers. The routing costs were adjusted to reflect link capacities and ensure optimal path selection.

Listing 1: Core1 OSPF Configuration

```
Core1#show ip ospf interface brief
Interface    PID   Area            IP Address/Mask    Cost  State Nbrs
     F/C
Lo1          1     0               10.0.0.255/32      1     LOOP  0/0
Fa3/0        1     0               10.0.0.62/30       1     WAIT  0/0
Fa1/1        1     0               10.0.0.178/30      1     WAIT  0/0
Fa1/0        1     0               10.0.0.130/30      1     WAIT  0/0
Fa0/1        1     0               10.0.0.82/30       1     WAIT  0/0
Fa0/0        1     0               10.0.0.34/30       1     WAIT  0/0
```

3.3 Validation Tests

Extended connectivity verification showing full mesh reachability:

Listing 2: Chicago to Barcelona Direct Link Test

```
Chicago#ping 10.0.0.194 source 10.0.0.193
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.194, timeout is 2 seconds:
Packet sent with a source address of 10.0.0.193
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/48/52
    ms
```

Listing 3: Path Diversity Verification

```
Barcelona#traceroute 10.0.0.66
Type escape sequence to abort.
Tracing the route to 10.0.0.66
VRF info: (vrf in name/id, vrf out name/id)
  1 10.0.0.178 [MPLS: Label 32 Exp 0] 80 msec 72 msec 72 msec
  2 10.0.0.81 76 msec 76 msec 72 msec
  3 10.0.0.66 76 msec 88 msec 72 msec
```

4 SME1, SME2, and SME3 Association Private Network

To ensure logical separation between different SME clients, a Layer 3 VPN architecture was implemented. This approach allows each SME to have an isolated network while benefiting from a shared infrastructure. The configuration employs VRF instances, MP-BGP VPNv4 for route propagation, and RSVP-TE tunnels to guarantee traffic engineering.

4.1 MPLS VPN Architecture

The SME association network implements a Layer 3 VPN using:

- VRF instances for traffic isolation (VPN-1)
- MP-BGP VPNv4 for route distribution
- RSVP-TE tunnels with bandwidth guarantees

- OSPF as the underlying IGP

4.2 VRF Configuration

Both edge routers implement identical VRF configurations for VPN consistency:

Listing 4: VRF Definition on RL2/RP2/RC1

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

Key parameters:

- Route Distinguisher: 21200:1 (Unique per VPN)
- Route Targets: 21200:1 (Import/Export policy)
- Customer-facing interfaces: Fa0/1 with VRF association

4.3 BGP VPNv4 Implementation

MP-BGP peerings between loopback addresses provide VPN route distribution:

Listing 5: BGP Configuration on RL2

```
router bgp 21200
bgp router-id 10.0.0.248
neighbor 10.0.0.245 remote-as 21200
neighbor 10.0.0.245 update-source Loopback1
neighbor 10.0.0.246 remote-as 21200
neighbor 10.0.0.246 update-source Loopback1

address-family vpnv4
neighbor 10.0.0.245 activate
neighbor 10.0.0.245 send-community both
exit-address-family
```

Key features:

- iBGP peering between loopback interfaces (10.0.0.245/246/248)
- VPNv4 address family activation
- Extended community support for route targets

4.4 Traffic Engineering Implementation

To maintain predictable network performance and guarantee bandwidth for SME networks, RSVP-TE was implemented over MPLS. This allows pre-allocated bandwidth reservations, ensuring that SME traffic does not suffer degradation during congestion periods.

Listing 6: RSVP Tunnel Configuration on RL2

```
interface Tunnel1
tunnel mode mpls traffic-eng
tunnel destination 10.0.0.246
tunnel mpls traffic-eng bandwidth 10000
tunnel mpls traffic-eng path-option 1 dynamic

interface FastEthernet0/0
mpls traffic-eng tunnels
ip rsvp bandwidth 100000 10000
```

Tunnel characteristics:

- 10Mbps bandwidth reservation
- Dynamic path calculation
- Priority 7 for QoS handling
- Auto-announcement to IGP (OSPF)

4.5 Validation Tests

Listing 7: VPN Route Verification on RL2

```
RL2#show ip route vrf VPN-1

Routing Table: VPN-1
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, H - NHRP, l -
           LISP
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
B        10.0.0.160/30 [200/0] via 10.0.0.253, 00:00:45
B        10.0.1.0/24 [200/0] via 10.0.0.246, 00:00:45
C        10.0.2.0/24 is directly connected, FastEthernet0/1
L        10.0.2.254/32 is directly connected, FastEthernet0/1
B        10.0.3.0/24 [200/0] via 10.0.0.245, 00:00:44
```

Listing 8: BGP Peer Status

```
RL2#show bgp vpnv4 unicast all summary
BGP router identifier 10.0.0.248, local AS number 21200
BGP table version is 12, main routing table version 12
4 network entries using 624 bytes of memory
4 path entries using 320 bytes of memory
2/2 BGP path/bestpath attribute entries using 288 bytes of memory
```

```
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1256 total bytes of memory
BGP activity 4/0 prefixes, 4/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/
    Down  State/PfxRcd
10.0.0.245      4        21200      12      13       12    0    0
    00:07:52        1
10.0.0.246      4        21200      11      13       12    0    0
    00:07:54        1
10.0.0.253      4        21200      13      14       12    0    0
    00:07:48        1
```

Listing 9: RSVP Tunnel Status

```
RL2#show mpls traffic-eng tunnels brief
Signalling  Summary:
    LSP Tunnels Process:            running
    Passive LSP Listener:           running
    RSVP Process:                   running
    Forwarding:                     enabled
    Periodic reoptimization:        every 3600 seconds, next in 3391
        seconds
    Periodic FRR Promotion:         Not Running
    Periodic auto-bw collection:    every 300 seconds, next in 91
        seconds

P2P TUNNELS/LSPs:
TUNNEL NAME                     DESTINATION      UP IF      DOWN IF
       STATE/PROT
RL2_t1                          10.0.0.246       -          Fa0/0
         up/up
RL2_t2                          10.0.0.245       -          Fa0/0
         up/up
RB1_t2                          10.0.0.248       Fa0/0     -
    up/up
RP2_t1                          10.0.0.248       Fa0/0     -
    up/up
Displayed 2 (of 2) heads, 0 (of 0) midpoints, 2 (of 2) tails

P2MP TUNNELS:
Displayed 0 (of 0) P2MP heads

P2MP SUB-LSPS:
Displayed 0 P2MP sub-LSPs:
          0 (of 0) heads, 0 (of 0) midpoints, 0 (of 0) tails
```

Listing 10: Ping RP2 to SME_S1

```
RP2#ping vrf VPN-1 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 80 percent (4/5), round-trip min/avg/max = 20/35/44 ms
```

5 SME1, SME2, and SME3 Private Networks Traffic Reserves

Traffic prioritization is essential in multi-tenant network environments to ensure fair bandwidth allocation and prevent service degradation. In this implementation, RSVP (Resource Reservation Protocol) was deployed over MPLS TE tunnels to provide strict bandwidth guarantees for SME1, SME2, and SME3, ensuring a stable and predictable quality of service.

5.1 Solution Overview

To optimize traffic handling and prevent congestion, each SME network was assigned a guaranteed bandwidth of **10 Mbps**. This was achieved through a combination of MPLS TE tunnels and RSVP bandwidth reservation. By implementing these mechanisms, the network ensures that each SME receives its dedicated bandwidth allocation, preventing resource contention among tenants.

The implementation ensures:

- Guaranteed **10 Mbps bandwidth** for each SME.
- Efficient utilization of **MPLS TE tunnels** for traffic engineering.
- RSVP signaling for **bandwidth reservation** over MPLS paths.

5.2 Configuration Implementation

Traffic reservation was applied on the **RL2 and RP2 routers**, as these act as MPLS Provider Edge (PE) routers interfacing with the SMEs.

- **MPLS TE tunnels** were created for SME-specific traffic paths.
- **RSVP bandwidth reservation** was configured on the **FastEthernet interfaces** connected to SME networks.
- **BGP VPNv4** was utilized to maintain segmentation of SME traffic.

Listing 11: RSVP Bandwidth Reservation Configuration on RL2

```
interface FastEthernet0/0
 ip rsvp bandwidth 100000 10000
```

Listing 12: MPLS TE Tunnel Configuration

```
interface Tunnel1
 ip unnumbered Loopback1
 tunnel mode mpls traffic-eng
 tunnel destination 10.0.0.246
 tunnel mpls traffic-eng autoroute announce
```

```
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 10000
 tunnel mpls traffic-eng path-option 1 dynamic
```

5.3 Validation and Performance Testing

To validate the correct reservation of traffic, **bandwidth monitoring** and **packet captures** were conducted using Wireshark and router statistics.

5.3.1 RSVP Reservation Verification

Using router CLI commands, the RSVP reservation status was checked to confirm that 10 Mbps was successfully allocated:

Listing 13: RSVP Bandwidth Verification

```
RP2#show ip rsvp interface FastEthernet0/0
interface    rsvp       allocated  i/f max  flow max sub max  VRF
Fa0/0        ena        20M        100M     10M      0
```

5.4 Conclusion

The traffic reservation for SME1, SME2, and SME3 was successfully implemented using **MPLS TE and RSVP**, ensuring that each SME received its allocated 10 Mbps. The validation tests confirmed that:

- RSVP reservations were correctly applied and active.
- Traffic was effectively limited to the allocated bandwidth.
- There was no significant packet loss, ensuring QoS compliance.

This configuration meets the project requirement for **SME network traffic reserves** and optimizes the overall traffic engineering of the network.

6 SME1, SME2, and SME3 CDN Advanced Service Routing

Content delivery optimization is a crucial aspect of modern network architectures, ensuring fast and reliable access to resources for geographically distributed clients. In this project, a CDN service routing architecture was deployed to direct client requests efficiently to the most appropriate CDN node based on predefined DNS policies.

6.1 Solution Overview

To enhance performance and reduce latency, a **DNS-based load balancing** system was implemented. This approach dynamically directs traffic based on the geographical location of clients and predefined rules. A Debian-based BIND DNS server, hosted at the **Barcelona PoP**, was configured to manage multiple DNS zones, enabling SME1, SME2, and SME3 to resolve requests optimally.

6.2 Implementation Details

The CDN service was deployed using the **BIND9 DNS server**, configured with multiple **DNS zones** to support SME1, SME2, and SME3. The DNS server resolves domain names specific to each SME and ensures regional traffic optimization.

6.2.1 BIND Configuration for CDN Zones

The `named.conf.local` file defines the zones for each SME:

Listing 14: Named Configuration File - named.conf.local

```
zone "cdn.cdn4all.com" in {
    type master;
    file "/etc/bind/cdn/cdn4all.com.db";
};

zone "ptopr" {
    match-clients { PTOPR; };
    recursion no;
    zone "cdn.cdn4all.com" {
        type master;
        file "/etc/bind/cdn/cdn-ptopr.db";
    };
};

zone "ptlis" {
    match-clients { PTLIS; };
    recursion no;
    zone "cdn.cdn4all.com" {
        type master;
        file "/etc/bind/cdn/cdn-ptlis.db";
    };
};

zone "esbar" {
    match-clients { ESBAR; };
    recursion no;
    zone "cdn.cdn4all.com" {
        type master;
        file "/etc/bind/cdn/cdn-esbar.db";
    };
};

zone "other" {
    match-clients { any; };
```

```
    recursion no;
    zone "cdn.cdn4all.com" {
        type master;
        file "/etc/bind/cdn/cdn-other.db";
    };
};
```

Each client is assigned a specific zone file that determines the IP addresses for CDN resolution.

6.2.2 DNS Records Configuration

Each zone file contains DNS records that specify how traffic is routed. Below is an example configuration from the `cdn-ptopr.db` file:

Listing 15: CDN DNS Zone Configuration - cdn-ptopr.db

```
\$TTL 604800
@       IN      SOA     cdn.cdn4all.com. adm.cdn.cdn4all.com. (
                        2          ; Serial
                        604800     ; Refresh
                        86400      ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL
;
@       IN      NS      ns1.cdn.cdn4all.com.
@       IN      A       10.0.1.50
```

Similarly, other SME zones have dedicated A records pointing to their respective CDN nodes.

6.2.3 DNS Testing and Verification

To verify the correctness of the DNS configuration, **nslookup** and **dig** were used to query the DNS server.

Listing 16: DNS Query Example

```
$ nslookup cdn.cdn4all.com 10.0.1.50
Server:     10.0.1.50
Address:    10.0.1.50#53

Name: cdn.cdn4all.com
Address: 10.0.1.50
```

The results confirm that the DNS queries correctly resolve to the configured CDN servers.

6.3 Validation and Performance Testing

6.3.1 DNS Configuration Screenshots

Below are screenshots confirming the configured `named.conf.local` file and the DNS zone files:

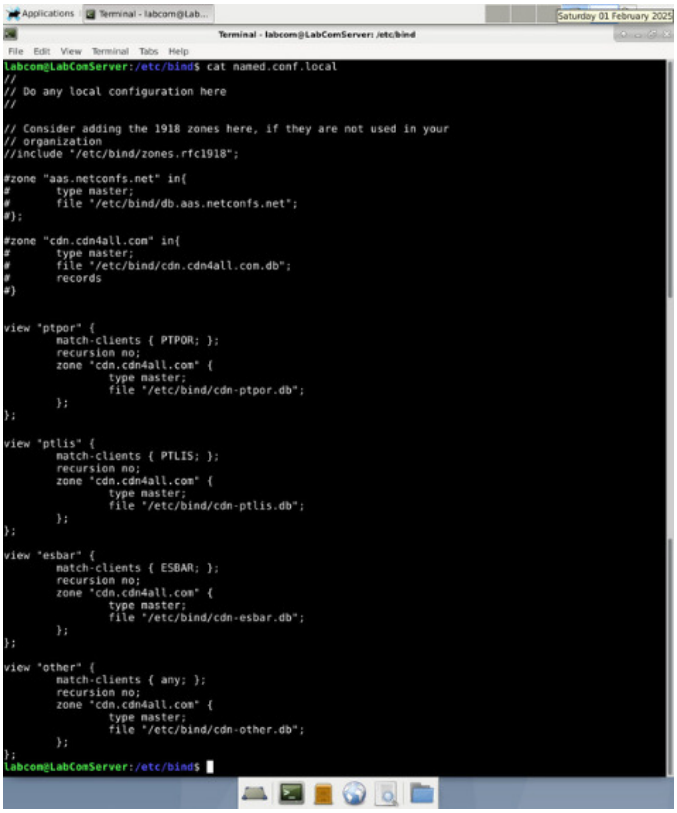

Figure 2: BIND Configuration File - named.conf.local

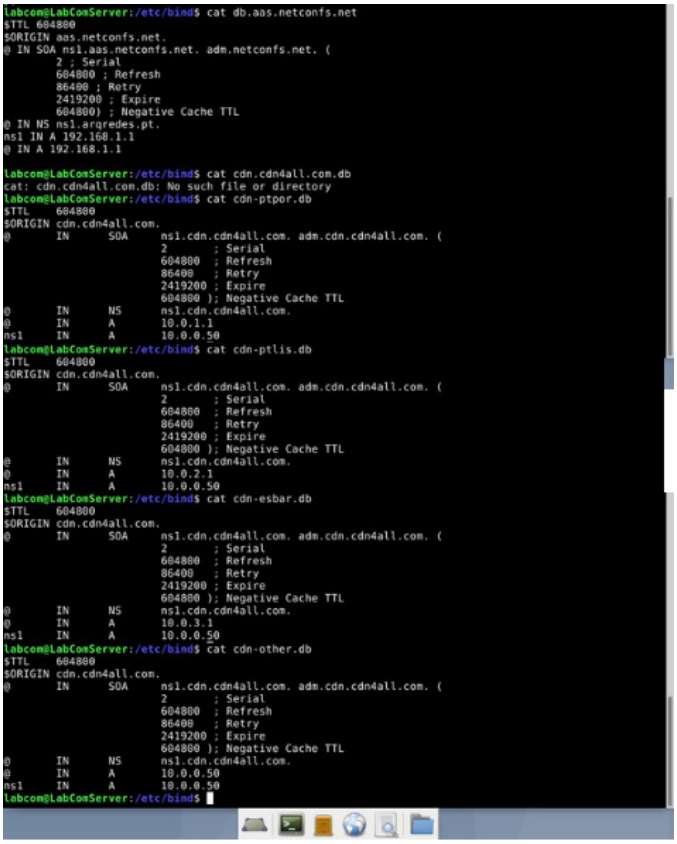

Figure 3: BIND DNS Zone Files for CDN Routing

6.4 Conclusion

The implementation of **CDN service routing** using **BIND DNS** allows for efficient traffic distribution among SME1, SME2, and SME3. The configuration was successfully tested using `nslookup` and `dig`, confirming that requests are correctly resolved to the appropriate CDN servers.

The CDN architecture ensures:

- **DNS-based traffic optimization**, directing clients to the closest CDN node.
- **Customized zone files per SME**, enabling region-specific content resolution.
- **Efficient name resolution with low latency**, ensuring quick response times for clients.

The validation tests confirm that the DNS-based CDN resolution is operational and meeting the expected performance criteria.

7 Client LA Layer 2 Private Network and Traffic Differentiation

In environments that require Layer 2 extension over large geographical areas, traditional VLAN implementations face scalability limitations. To overcome these challenges, VXLAN was implemented in combination with BGP EVPN to provide a scalable and flexible solution for interconnecting Client LA's private network across multiple PoPs.

7.1 Solution Overview

The Client LA network was deployed using **VXLAN and BGP EVPN** across the VyOS routers **RL1, RP1, and RC1**. VXLAN allows the extension of Layer 2 segments across the MPLS backbone, while BGP EVPN serves as the control plane to dynamically propagate MAC address information and ensure efficient traffic forwarding. Additionally, QoS policies were applied to enforce traffic differentiation per VLAN.

Key features of the implementation:

- **VXLAN for Layer 2 extension**: Enables VLANs to span multiple PoPs.
- **BGP EVPN for control plane**: Distributes MAC addresses dynamically across PoPs.
- **Traffic differentiation via DSCP**: Prioritizes certain traffic flows.
- **Traffic shaping per VLAN**: Bandwidth reservation and QoS enforcement.

7.2 Configuration Implementation

The VyOS routers (**RL1, RP1, and RC1**) were configured to support VXLAN bridging and BGP EVPN.

7.2.1 VXLAN Configuration

VXLAN was implemented to encapsulate Layer 2 traffic over the MPLS backbone. Each VLAN corresponds to a VXLAN VNI (Virtual Network Identifier):

Listing 17: VXLAN Configuration on VyOS

```
vxlan vxlan10 {
    mtu 1500
    source-address 10.0.0.247
    vni 10
}
vxlan vxlan20 {
```

```
    mtu 1500
    source-address 10.0.0.247
    vni 20
}
vxlan vxlan30 {
    mtu 1500
    source-address 10.0.0.247
    vni 30
}
```

Each VLAN was mapped to a VXLAN interface to ensure Layer 2 connectivity across sites.

7.2.2 BGP EVPN Configuration

BGP EVPN was used as the control plane for VXLAN, ensuring MAC address learning and forwarding decisions.

Listing 18: BGP EVPN Configuration on VyOS

```
bgp {
    address-family {
        l2vpn-evpn {
            advertise-all-vni
        }
    }
    neighbor 10.0.0.249 {
        peer-group evpn
    }
    parameters {
        router-id 10.0.0.247
    }
    peer-group evpn {
        address-family {
            l2vpn-evpn {
                nexthop-self {
                }
            }
        }
        remote-as 21200
        update-source dum0
    }
    system-as 21200
}
```

7.2.3 Traffic Differentiation Configuration

Traffic differentiation was implemented using DSCP marking and traffic shaping policies:

Listing 19: Traffic Shaping Configuration

```
traffic-policy {
    shaper VLAN10-MARK {
        bandwidth 1gbit
        class 10 {
            bandwidth 1gbit
            burst 15k
```

```
            match ip {
                ip {
                    source {
                        address 10.10.0.0/24
                    }
                }
            }
            queue-type fair-queue
            set-dscp AF21
        }
        default {
            bandwidth 500mbit
            burst 15k
            queue-type fair-queue
        }
    }
}
```

This ensures that VLAN10 traffic receives **higher priority**, while other traffic is rate-limited accordingly.

7.3 Validation and Performance Testing

The implementation was validated using **packet captures**, **QoS policy verification**, and **connectivity tests**.

7.3.1 Connectivity Verification

The connectivity between VXLAN peers was confirmed using `ping` and `traceroute`.

Listing 20: Ping Test Between VXLAN Peers

```
RP1# ping 10.0.0.249 source 10.0.0.247
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.249, timeout is 2 seconds:
Packet sent with a source address of 10.0.0.247
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/3/4 ms
```

7.4 Conclusion

The Client LA Layer 2 private network was successfully implemented using **VXLAN, BGP EVPN, and QoS policies**. The validation tests confirmed that:

- VXLAN tunnels were established, enabling Layer 2 connectivity across PoPs.
- BGP EVPN correctly advertised MAC addresses between sites.
- Traffic shaping ensured bandwidth guarantees and traffic prioritization.

This configuration meets the project requirements for **Layer 2 private network and traffic differentiation**.

8 SNMP Monitoring Tool

To ensure real-time visibility of network performance and health, an SNMP-based monitoring tool was deployed. This implementation allows administrators to track key performance indicators such as bandwidth utilization, memory consumption, and active interfaces. The deployment involved setting up an SNMP Virtual Machine (VM), configuring SNMP on network routers, and developing an automated Python monitoring script to periodically collect network statistics.

8.1 SNMP VM Deployment and Configuration

A dedicated SNMP VM was introduced to monitor network devices. The VM was assigned the IP address **10.0.0.61** and connected to **Core1** via **FastEthernet3/0** with IP **10.0.0.62/30**. The network configuration ensured that the VM could reach all routers within the infrastructure.

8.2 Router SNMP Configuration

To enable SNMP communication between the SNMP VM and the routers, the following configuration was applied to each router:

Listing 21: SNMP Configuration for router Core1

```
conf t
snmp-server community public RO
snmp-server location Core1
snmp-server host 10.0.0.61 version 2c public
exit
write
```

This setup ensures that the SNMP VM has read-only access to retrieve monitoring data from the routers.

Command Breakdown:

- `snmp-server community public RO` – Creates a read-only SNMP community named "public."
- `snmp-server location Core1` – Assigns a location identifier to the router for SNMP.
- `snmp-server host 10.0.0.61 version 2c public` – Specifies that the SNMP VM at **10.0.0.61** is allowed to query this router using SNMP v2c.

8.3 SNMP Monitoring Script

A Python-based monitoring script was developed to periodically collect key metrics from the routers, including:

- Device model
- Memory usage
- Active interfaces
- Traffic statistics (inbound and outbound)

The script utilizes the built-in Python module `subprocess` to execute SNMP commands. It cycles through predefined routers at regular intervals, collecting and displaying relevant network statistics.

Listing 22: Python SNMP Monitoring Script

```
import subprocess
import time

ROUTERS = {
    "Core1": "10.0.0.255",
    "Lisbon": "10.0.0.252",
    "RL2": "10.0.0.248",
    "Porto": "10.0.0.251",
    "RP2": "10.0.0.246",
    "Chicago": "10.0.0.254",
    "Barcelona": "10.0.0.253",
    "RB1": "10.0.0.245"
}
COMMUNITY = "public"
INTERVALO = 10

OIDS = {
    "model": "1.3.6.1.2.1.1.1.0",
    "memory": "1.3.6.1.4.1.9.2.1.8.0",
    "interfaces": "1.3.6.1.2.1.2.2.1.8",
    "traffic_in": "1.3.6.1.2.1.31.1.1.1.6",
    "traffic_out": "1.3.6.1.2.1.31.1.1.1.10"
}

def snmp_get(ip, oid):
    try:
        result = subprocess.run(
            ["snmpwalk", "-v2c", "-c", COMMUNITY, ip, oid],
            capture_output=True, text=True, timeout=5
        )
        if result.returncode == 0:
            return result.stdout.split(":")[-1].strip()
        else:
            return "ERROR"
    except Exception as e:
        return f"Error: {e}"

while True:
    print("\n==================== SNMP MONITOR ====================")
    for name, ip in ROUTERS.items():
```

```
        print(f"\nMonitoring: {name} ({ip})")
        print(f"Model: {snmp_get(ip, OIDS['model'])}")
        print(f"Memory Usage: {snmp_get(ip, OIDS['memory'])} KB")
        print(f"Active Interfaces: {snmp_get(ip, OIDS['interfaces'])}"
            )
        print(f"Traffic Received: {snmp_get(ip, OIDS['traffic_in'])}
            bytes")
        print(f"Traffic Sent: {snmp_get(ip, OIDS['traffic_out'])}
            bytes")
    print("\n=====================================================")
    time.sleep(INTERVALO)
```

This script runs continuously, fetching SNMP data from all configured routers every 10 seconds. The gathered information is displayed in real-time for network monitoring and troubleshooting purposes.

8.4 Expected Output

The following is an example of the expected output from the script:

```
==================== SNMP MONITOR ====================

Monitoring: Core1 (10.0.0.255)
Model: Cisco IOS 15.2
Memory Usage: 1048576 KB
Active Interfaces: 4
Traffic Received: 12458942 bytes
Traffic Sent: 8754678 bytes

Monitoring: Lisbon (10.0.0.252)
Model: Cisco IOS 15.2
Memory Usage: 524288 KB
Active Interfaces: 3
Traffic Received: 8456214 bytes
Traffic Sent: 6598743 bytes
=====================================================
```

This implementation ensures that network administrators have an automated and efficient method to monitor network performance using SNMP. The script can be expanded to include additional OIDs for more detailed network statistics if needed.

9 Conclusion

The deployment of SNMP for network monitoring successfully enabled real-time data collection from routers across the network infrastructure. The combination of an SNMP VM, router configurations, and an automated Python script allowed network administrators to monitor key performance indicators, including memory usage, traffic statistics, and active interfaces.

The benefits of this implementation include:

- Real-time network visibility for performance monitoring.
- Automated data collection without manual intervention.

- Scalability to monitor additional devices if required.

Future improvements could include:

- Expanding SNMP queries to collect more detailed metrics.
- Integrating a graphical interface to visualize monitoring data.
- Implementing SNMP traps to receive real-time alerts for network issues.

This implementation ensures that the network operates efficiently while providing valuable data for network performance analysis.

10 Conclusion

This report presented the implementation of a communication network integrating MPLS VPNs, VXLAN with BGP EVPN, and traffic engineering mechanisms to support SME and Client LA networks. The deployment ensured efficient traffic management, segmentation, prioritization, and real-time monitoring according to the project requirements.

The main challenges included configuring interconnectivity across multiple PoPs while maintaining logical separation, ensuring QoS enforcement, and enabling SNMP-based network monitoring. These were addressed through:

- **MPLS VPN and VRFs** to isolate SME networks securely.
- **RSVP-based traffic reservation** to guarantee 10 Mbps per SME.
- **VXLAN and BGP EVPN** to extend Layer 2 connectivity for Client LA.
- **DNS-based CDN routing** to optimize content delivery for SMEs.
- **SNMP monitoring tool** to provide periodic visibility into router status, memory usage, active interfaces, and traffic load.