University of Pisa
MSc in Computer Engineering
Advanced Network Architectures and Wireless Systems

QoS project

Daniela Comola
Sara Lotano
Eugenia Petrangeli

Academic Year 2019/2020

Contents

1 Introduction

This project considers a multimedia network in which Quality of Service requirements must be met using a Differentiated Services Architecture and MPLS TE.

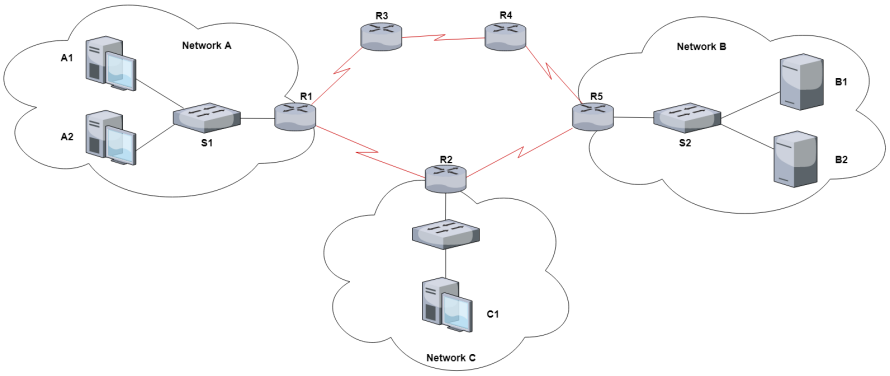

Figure 1: Network Architecture

The network consists of three different access networks (Network A, B and C) interconnected through a core network of five routers (R1 - R5).
The communications happens from clients to servers according to the data flow characteristics shown below:

Source	Destination	Priority	Bandwidth
A1	B1	High	0.9 Mbps
A2	B2	Low	1.2 Mbps
C1	B1	High	0.9 Mbps

Table 1: Data flow characteristics

1.1 Requirements

- Each flow has to be conformed with the characteristics shown in the table. Moreover:
  - for high priority flows, the excess traffic must be downgraded to low priority.
  - for low priority traffic, the excess traffic must be dropped.
- For each high priority flow must be granted at worst 60% of the bandwidth for the link it traverses in the core network.

2 Design

The network was built using the network software emulator **GNS3**. The resulting topology is shown below:

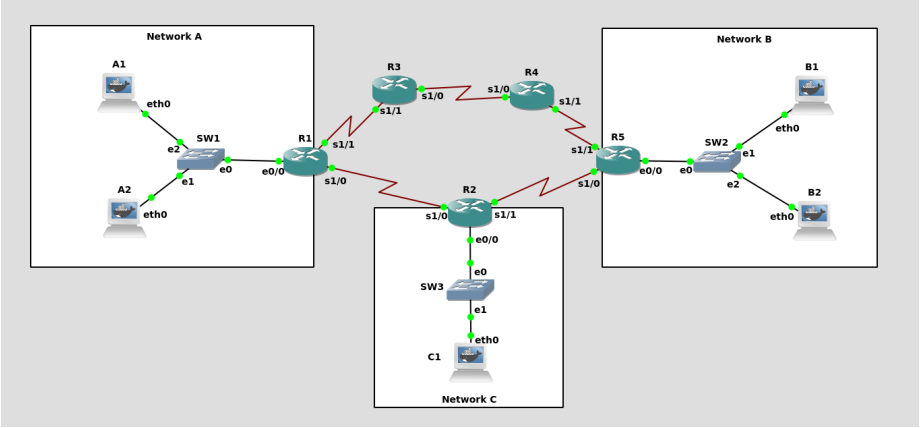

Figure 2: GNS3 network topology

To ensure basic communication between all hosts belonging to the network, all network interfaces have been configured using the static addressing plan shown in the table below and, on all routers, OSPF has been configured as the routing protocol.

2.1 Address planning

Network	Device	Interface	IP address	Network mask
Core	**R1**	Serial 1/0	10.1.1.1	/30
		Serial 1/1	10.2.2.1	/30
		Loopback 0	192.168.4.1	/32
	R2	Serial 1/0	10.1.1.2	/30
		Serial 1/1	10.5.5.1	/30
		Loopback 0	192.168.4.2	/32
	R3	Serial 1/0	10.2.2.2	/30
		Serial 1/1	10.3.3.1	/30
		Loopback 0	192.168.4.3	/32
	R4	Serial 1/0	10.3.3.2	/30
		Serial 1/1	10.4.4.1	/30
		Loopback 0	192.168.4.4	/32
	R5	Serial 1/0	10.5.5.2	/30
		Serial 1/1	10.4.4.2	/30
		Loopback 0	192.168.4.5	/32
A	**R1**	Ethernet 0/0	192.168.1.1	/24
	A1	Ethernet 0	192.168.1.2	/24
	A2	Ethernet 0	192.168.1.3	/24
B	**R2**	Ethernet 0/0	192.168.2.1	/24
	B1	Ethernet 0	192.168.2.2	/24
	B2	Ethernet 0	192.168.2.3	/24
C	**R5**	Ethernet 0/0	192.168.3.1	/24
	C1	Ethernet 0	192.168.3.2	/24

3 Configuration

The configuration of routers has been made considering the following steps:

3.1 Classification

Inside the R1 router the following access-lists have been defined:

```
1    access-list 100 permit ip host 192.168.1.2 host 192.168.2.2
2    access-list 101 permit ip host 192.168.1.3 host 192.168.2.3
```

Both activated on the Ethernet0/0 input interface.
Each access-list has been associated with a class-map:

- access-list 100 with class-map **A1_TO_B1**
- access-list 101 with class-map **A2_TO_B2**

Inside the R2 router, on Ethernet0/0 interface, the following access-list has been defined:

```
1    access-list 100 permit ip host 192.168.3.2 host 192.168.2.2
```

and it has been associated with class-map **C1_TO_B1**

3.2 Marking

The following DSCP values were used to mark the traffic:

- **EF** (Expedited Forwarding): used to mark high-priority flows.
- **AF11** (Assured Forwarding): used to mark low-priority flows.

The marking is done only at the ingress of the core network, within the routers R1 and R2.

To check the correct marking of high-priority packets, we can use Wireshark to inspect one of the packets belonging to the data flow generated from A1 to B1:

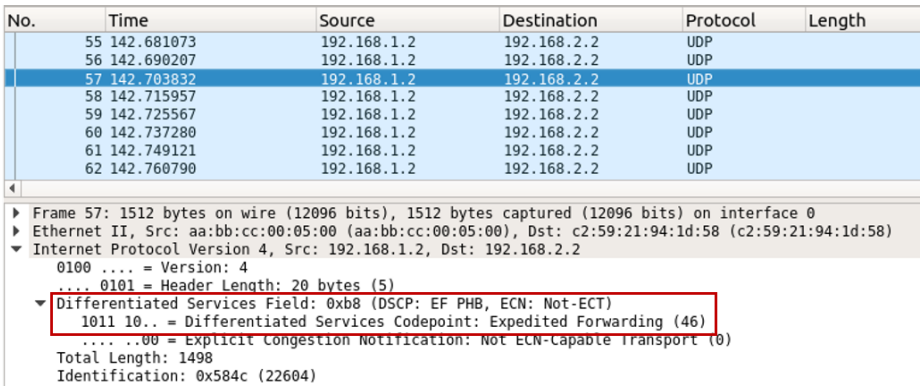

Figure 3: Example of packet marked with DSCP value: ef

The same type of control can be performed for low-priority packets. For example, analyzing those generated from A2 to B2:

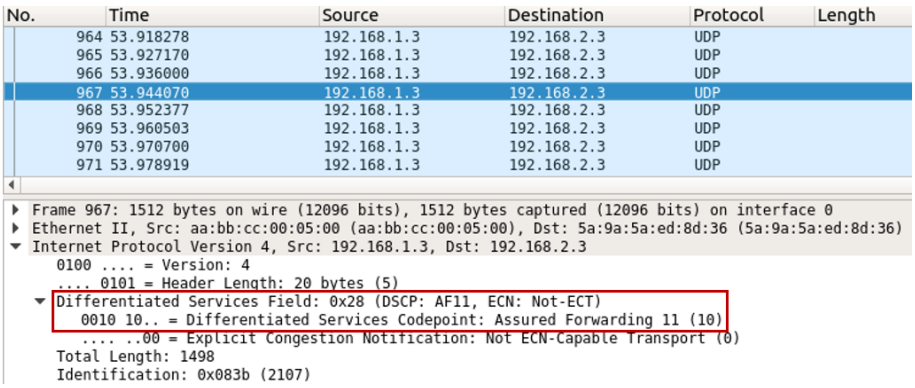

Figure 4: Example of packet marked with DSCP value: af11

3.2.1 Class-Based WFQ

In order to guarantee the requirement that each high-priority flow must be guaranteed in the worst case for 60% of the bandwidth on each link it crosses within the core network, the first thing to do is to configure the following access-list on all routers:

```
1    access-list 102 permit ip any any dscp ef
```

which is associated with the class-map **HIGH_PRIO_FLOW**.
In order to set the bandwidth share for the high-priority traffic, we define:

```
1    policy-map OUT
```

```
        class HIGH_PRIO_FLOW
        bandwidth percent 60
```

and we apply this configuration as output service-policy.

3.3 Policing

To meet the bandwidth characteristic of the data flow, we have used **CAR** (Committed Access Rate) that is adopted at the ingress routers to limit ingress traffic of a flow.

We define a policing function, applied on the R1 and R2 Ethernet0/0 interfaces, whose purpose is to limit high-priority traffic to 0.9 Mbps and to remark the exceeded traffic. This behavior is obtained with the option exceed-action set-dscp-transmit 10 (the value 10 has been chosen because it corresponds to the value AF11 used for low-priority traffic).
For example, the configuration on the R1 router is as follows:

```
    rate-limit input access-group 100 896000 5000 5000 conform-
    action transmit exceed-action set-dscp-transmit 10
```

To check the correct policy configuration we can, for example, generate 1M traffic from host A1 to host B1 and check the correct application of the rules set on the R1 router. Looking at the R1 router console, we can see that part of the packets have been transmitted while the remaining part has been remarked with the new DSCP value of 10, as we expected.

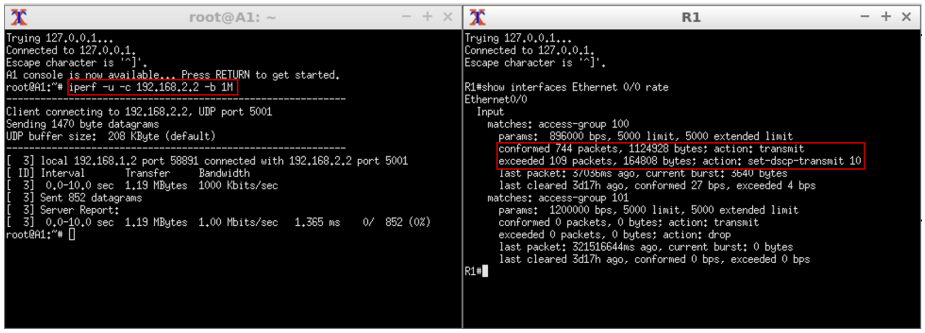

Figure 5: Example of packet remarking

The same behavior is obtained by generating a data flow from host C1 to host B1 and analyzing the status of the Ethernet 0/0 interface of the R2 router.

A second policing function has been defined only for the R1 router that limits low-priority traffic to a rate of 1.2 Mbps and drops the exceeded traffic.

```
    rate-limit input access-group 101 1200000 5000 5000 conform-
    action transmit exceed-action drop
```

To check the correct policy configuration we generated 2M traffic from host A2 to host B2. We can see that part of the packets have been transmitted while the rest have been dropped, as we expected.

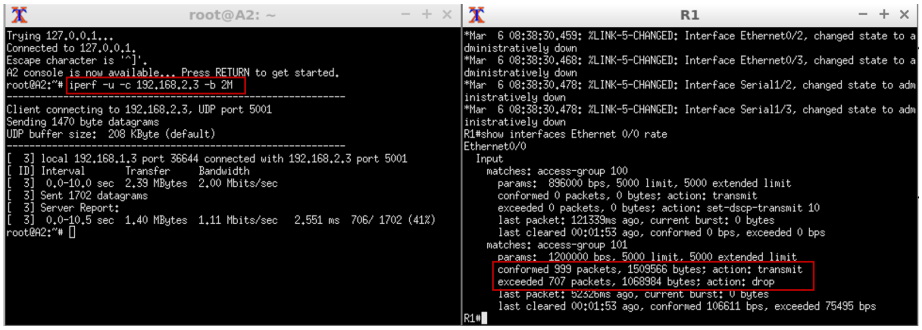

Figure 6: Example of packet dropping

3.4 MPLS - TE

To correctly configure MPLS we have enabled the layer 3 switching technology **CEF** (Cisco Express Forwarding) on all routers.
To guarantee 60% bandwidth for the high priority traffic, a tunnel has been created from router R1 to router R5, through router R3 and R4. Without any tunnel, the OSPF protocol would always forward high priority traffic to the R2 router as it is on the shortest path to reach R5.

Tunnel 0 has been created by specifying the path, called *longpath*:

```
ip explicit-path name longpath enable
next-address 10.2.2.2
next-address 10.3.3.2
next-address 10.4.4.2
```

In order to differentiate routing we can set the following static routes on R1:

```
ip route 192.168.2.2 255.255.255.255 Tunnel 0
ip route 192.168.2.3 255.255.255.255 Serial 1/0
```

We can use **traceroute** command to print the route that packets, from A1 and A2, take to reach the hosts B1 and B2 respectively.

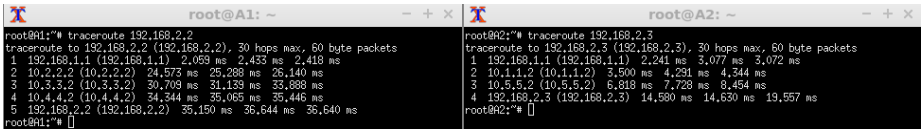

Figure 7: Traceroute used on A1 and A2 hosts

To test the correctness of the guaranteed bandwidth in the worst case at 60% we can perform some tests using **iperf**. In both cases, we must remember that the bandwidth of a serial link is equal to 1544 Kbit/s.

3.4.1 Test 1

In this first test, we generate 1 Mpbs traffic from the hosts A1 and C1.

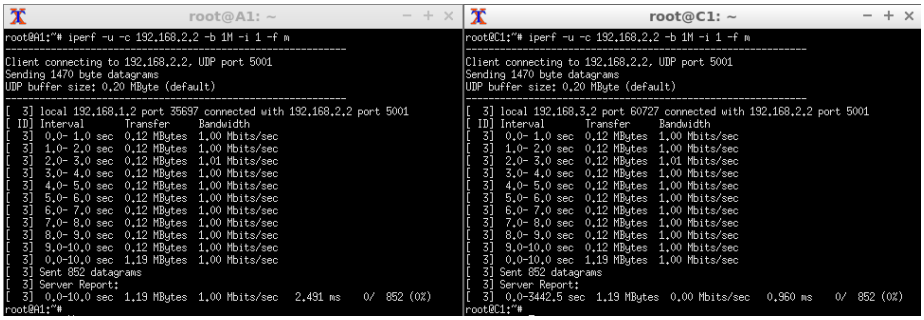

Figure 8: Data flows generated from A1 and C1

As can be seen, the percentage of lost packages is equal to 0% in both cases. This is because data flows follow different paths as shown above.

3.4.2 Test 2

In the second test, we generate 1 Mpbs traffic from hosts A2 and C1. In this case, the traffic will follow the same path through the R2 and R5.

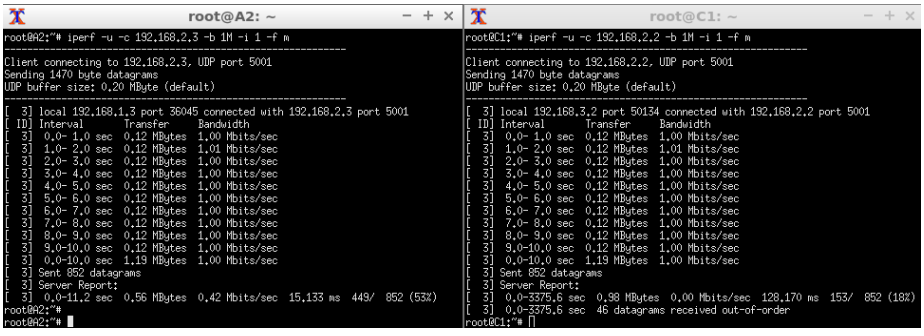

Figure 9: Data flows generated from A2 and C1

As shown by the C1 console, the percentage of lost packets is only 18%, so the bandwidth constraint is satisfied.

4 Running the project

4.1 Prerequisites

To develop the multimedia network, we used **GNS3** (Graphical Newtork Simulator) which allows the emulation of real hosts and real Cisco Routers.

4.2 Starting the network

In order to open the network, just run the following command:

```
$gns3 multimedia_network.gns3
```

Once the network is opened, all devices must be started. When the LEDs of all interfaces are green, the network is ready to be used.