# 103087 - ICT Project Communication Services and Security

## Activity 3

Jordi Rafael Lazo Florensa
Alejandro Clavera Poza

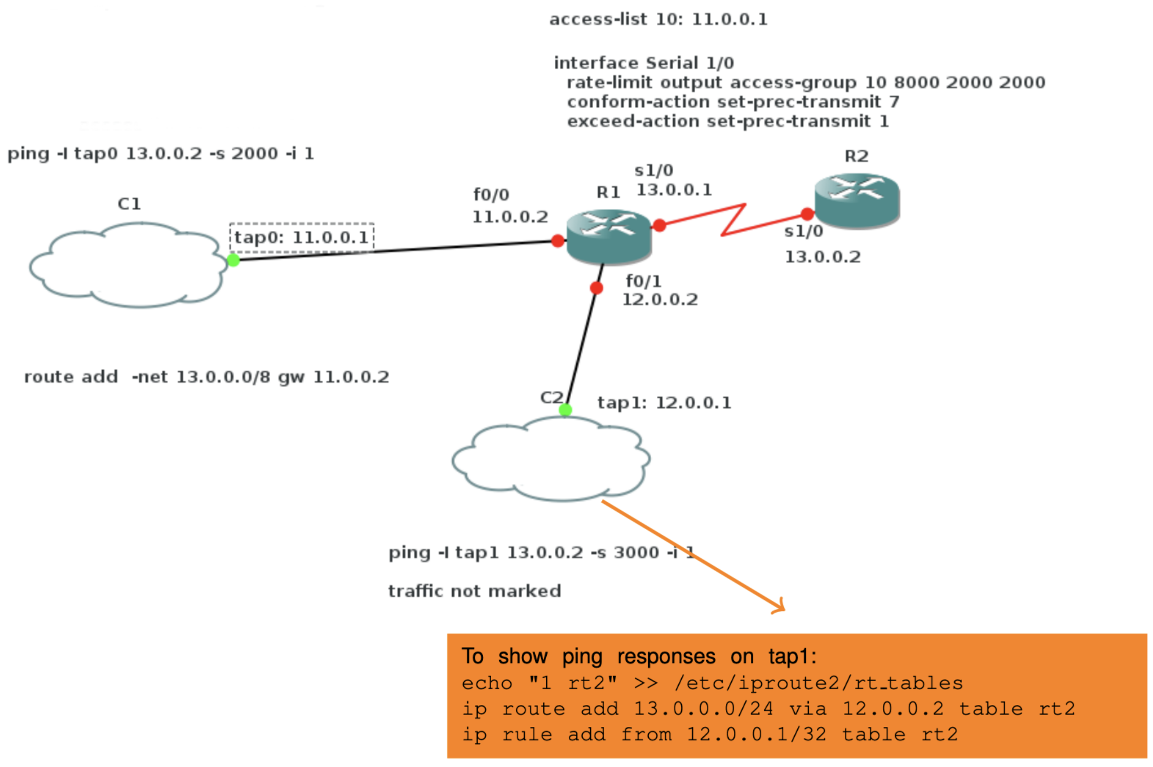
2 April 2023

Master's degree in Computer Engineering

**Universitat de Lleida**
Escola Politècnica Superior

## 1 Problem 1

Based on topology from slide page 22 (QoS, CAR), write a shell script that computes the average rate for the following traffic flows:


access-list 10: 11.0.0.1
interface Serial 1/0
rate-limit output access-group 10 8000 2000 2000
conform-action set-prec-transmit 7
exceed-action set-prec-transmit 1
ping -I tap0 13.0.0.2 -s 2000 -i 1
C1
tap0: 11.0.0.1
f0/0
11.0.0.2
R1
s1/0
13.0.0.1
R2
s1/0
13.0.0.2
f0/1
12.0.0.2
route add -net 13.0.0.0/8 gw 11.0.0.2
C2
tap1: 12.0.0.1
ping -I tap1 13.0.0.2 -s 3000 -i 1
traffic not marked

```
To show ping responses on tap1:
echo "1 rt2" >> /etc/iproute2/rt_tables
ip route add 13.0.0.0/24 via 12.0.0.2 table rt2
ip rule add from 12.0.0.1/32 table rt2
```

Figure 1: Network topology problem 1

- From 11.0.0.1 with precedence 7
- From 11.0.0.1 with precedence 1
- 12.0.0.1

captured at interface R1-s1/0. Use tshark application or wireshark statistics.

| Flow | Average rate |
|---|---|
| From 11.0.0.1 with precedence 7 | 8,21 Kbps |
| From 11.0.0.1 with precedence 1 | 8,41 Kbps |
| From 12.0.0.1 | 24,79 Kbps |

Table 1: Average rate of each flow

After capturing the traffic of the serial 1/0 interface with wireshark for approximately 90 seconds, it can be seen that the traffic comes from IP 12.0.0.1 (which generates traffic at a constant speed of approximately 24 Kbps). Since no QOS policy is applied to this traffic, it can be seen that the average rate is around 24 Kbps. On the other hand, it can be seen how the traffic coming from IP 11.0.0.1 with origin 1 and origin 7 have a very similar average rate of around 8 Kbps.

This is due to the fact that after the implementation of the CAR algorithm on the 11.0.0.1 traffic, approximately half of the packets are classified at a precedence of 1, which causes the transmission rate to be affected in such a way that the traffic of both classes share the same flow rate. This can be seen in the following figure:

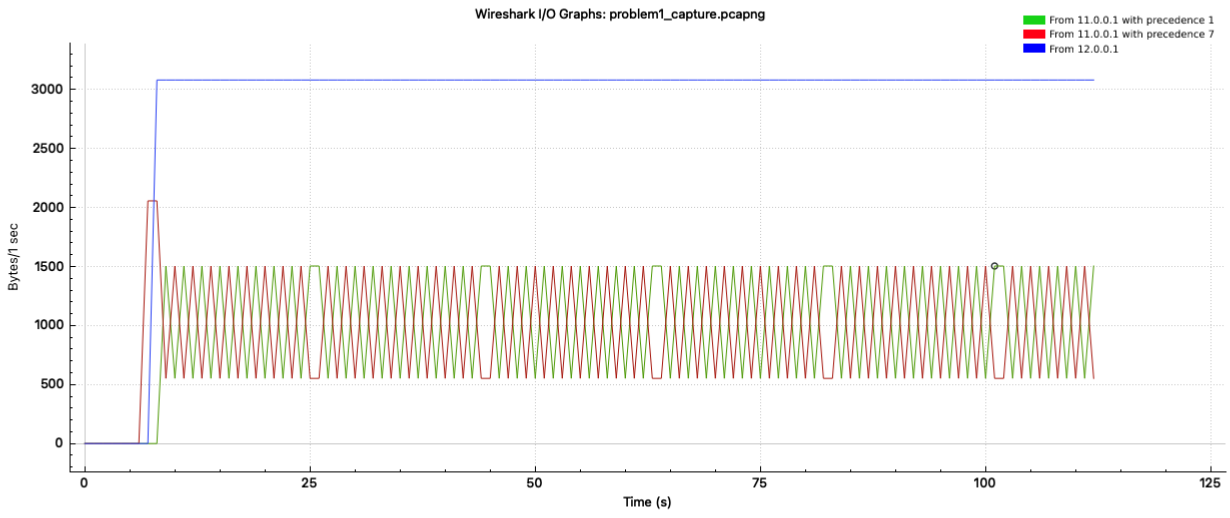


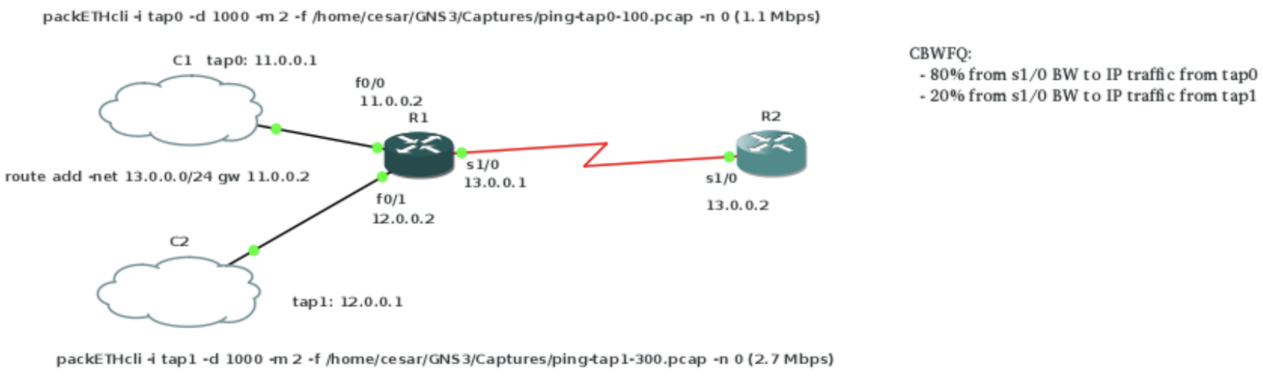
Figure 2: Problem 1 traffic capture

The figure 2 shows how the packets from origin 1 and 7 share the channel. In such a way that the average rate oscillates between the values 12 and 4 Kbps, which implies an average rate of 8 Kbps.

In this way, it can be seen that thanks to this application of the algorithm, this average rate is reduced by half, which would go from 16 to 8 Kbps.

## 2 Problem 2

Build the same topology as in slide page 40 (QoS, CBWFQ) but using Custom Queuing (CQ). Measure the bandwidth occupation at serial link R1-R2 for each of the streams coming from C1-tap0 and C2-tap1. Use tshark application or wireshark statistics. (Remember: avoid to user queue 0).



```
packETHcli -i tap0 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap0-100.pcap -n 0 (1.1 Mbps)
route add -net 13.0.0.0/24 gw 11.0.0.2
packETHcli -i tap1 -d 1000 -m 2 -f /home/cesar/GNS3/Captures/ping-tap1-300.pcap -n 0 (2.7 Mbps)
```

Figure 3: Network topology problem 2

After configuring the topology of problem 2 with the Custom Queuing, to verify that the configuration was working correctly, two ping flows of around 1.7 Mbps were generated, obtaining the result shown in the Figure 4. Where it can be seen that the traffic that comes from tap0, in red colour, occupies 80% of the bandwidth while the traffic that comes from tap1, in green colour, occupies the remaining 20%.

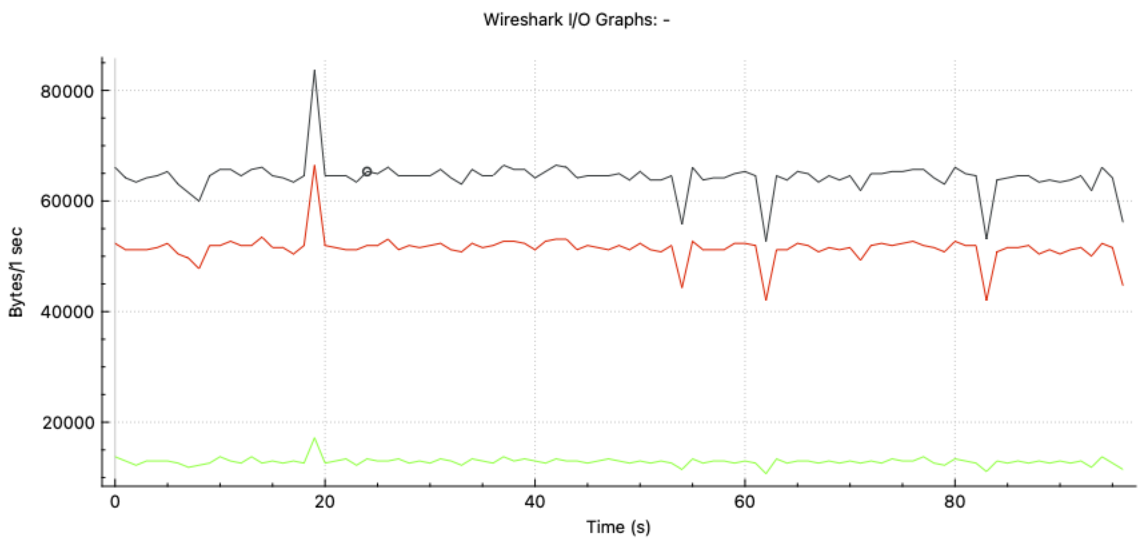
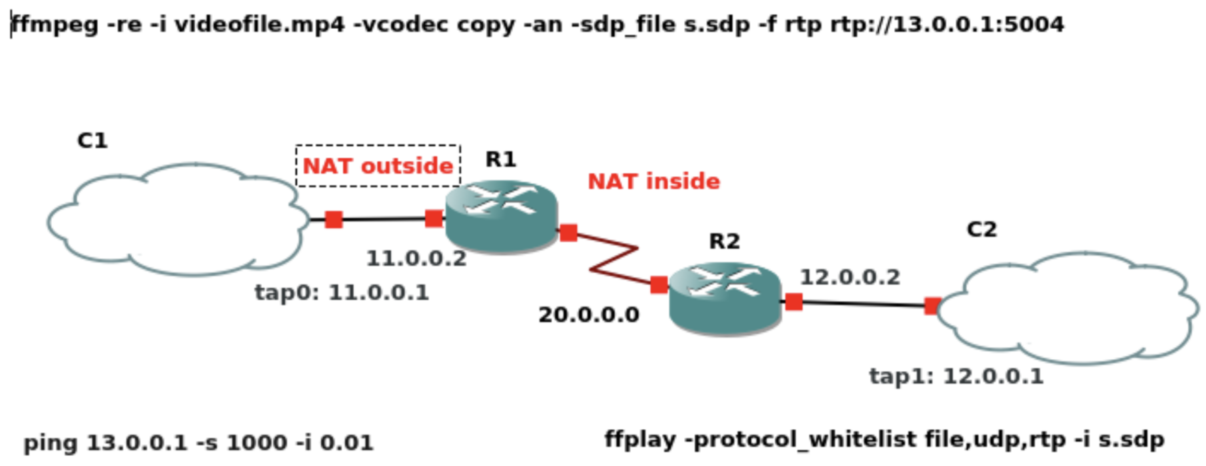


Figure 4: Problem 2 traffic capture

## 3 Problem 3

Considering the following topology:



```
ffmpeg -re -i videofile.mp4 -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004
```

```
ping 13.0.0.1 -s 1000 -i 0.01
```

```
ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

Figure 5: Network topology problem 3

Answer the following questions:

1. **Detail the required NAT related R1 configuration as well as the correct routing directives on your PC. Note that video stream must be delivered from 11.0.0.1 to 13.0.0.1.**

   The NAT configuration of router R1 is as follows:

```
interface Serial1/0
    ip address 20.0.0.1 255.255.255.0
    ip nat inside
ip nat inside source static 12.0.0.1 13.0.0.1
```

And the configuration of the path on the PC is as follows: (taking into account that it is working on the mac operating system)

```
route add -host 13.0.0.1 11.0.0.2
route add -host 11.0.0.1 12.0.0.2
```

2. **Detail the required priority queuing (PQ) related R1 configuration in order to provide high priority to video streaming and low priority to default traffic. Which is the maximum allowed speed transmission rate along the path?**

```
interface Serial1/0
    priority-group 1
priority-list 1 protocol ip high udp 5004
priority-list 1 default low
```

   Assuming that the bottleneck is the Serial1/0 connection, the maximum transmission limit is: 1,5 Mbps.

3. **Download from here a video sample with the corresponding video bit rate encoding slightly below your maximum transmission rate. Detail the complete ping to achieve the same transmission rate that your video streaming.**

   We have chosen to send the 500 Kbps video, because it is the one that represents a number of acceptable losses at the same time that it had an acceptable bit rate.

   To send a ping with the same transmission rate we have used the following command(we are working on mac):

```
sudo ping -S 11.0.0.1 13.0.0.1 -s 150 -i 0.001
```

4. **While streaming video, measure the number of RTP missed packets (during 1 minute of transmission) by ffplayer when using PQ and without PQ. Explain how you obtain those measurements.**

   To obtain these results, the retransmission has first been generated using the following commands:

```
ffmpeg  -re -i man500Kbps.mp4 -vcodec copy -an -sdp_file
s.sdp -f rtp rtp://13.0.0.1:5004

ffplay -protocol_whitelist file,udp,rtp -i s.sdp -report
```

After executing these commands, a *.log* file has been generated that shows the logs generated by the *ffplay* command. Among this information is how many packets have been lost. With the format:

```
RTP missed x packets
```

Therefore we have used the following python script that reads this file and extracts the total number of packets lost during transmission.

```python
import sys
import re

# Check if file name was provided as an argument
if len(sys.argv) != 2:
    print("Usage: python script.py <log_file>")
    sys.exit(1)

filename = sys.argv[1]

# Open the file and read its contents
try:
    with open(filename, 'r') as file:
        file_contents = file.read()
except FileNotFoundError:
    print("File not found.")
    sys.exit(1)

# Use regular expressions to extract missed packet counts and sum them
pattern = r"RTP: missed (\d+) packets"
matches = re.findall(pattern, file_contents)
sum = 0
for match in matches:
    sum += int(match)

# Print out the sum of missed packet counts
print("Total RTP missed packets :", sum)
```

Once this process has been applied to a video broadcast, using and without using the PQ, the results obtained are the following:

| **Execution** | **Number of missed packets** |
|---|---|
| Not using PQ | 341 |
| Using PQ | 387 |

Table 2: Missed packets while the video was running.

As can be seen in the table, it is not possible to observe a great difference in the number of missed packets because, since there is no traffic that intercepts the video retransmission, the number of missed packets is included due to the saturation of the PQ generated by the same transmission.

5. **Repeat the last item adding a ping transmission at the same rate that the video streaming.**

| **Execution** | **Number of missed packets** |
|---|---|
| Not using PQ | 1694 |
| Using PQ | 662 |

Table 3: Missed packets while the video and ping was running.

As can be seen in the table, a great improvement can be seen in the missed packets thanks to the use of the PQ, since it gives greater preference to the video transmission packets, avoiding the saturation of the assigned queue because it has a higher priority.