Universitat de Lleida

MÀSTER en ENGINYERIA INFORMÀTICA

ESCOLA POLITÈCNICA SUPERIOR

CURS 2020/2021

Communication Services and Security
Exercise 4

Authors:
Lluís Mas
Radu Spaimoc

02/05/21

Universitat de Lleida

Contents

List of Figures

1 Problem

Download this video (Elecard 4K video about Tomsk, part 2, Bit rate: 0.8 Mbps, Size: 16 MB).

1. **Considering 2 values for the token bucket filter:**
   - Average rate: 10^6 (bps) and bc=be=$4 \cdot 10^5$ (bits)
   - Average rate: $8 \cdot 10^5$ (bps) and bc=be=$4 \cdot 10^3$ (bits)

   Reproduce the plot on slide 86 (QoS, Traffic shaping) for the total video duration.
2. **Compute the number of lost frames at the player during the first 30 seconds in the attached cases.**

In the following sections we will explain the steps followed to solve the different parts of this problem.

1.1 Topology

Figure 1 shows the structure of the implemented topology.

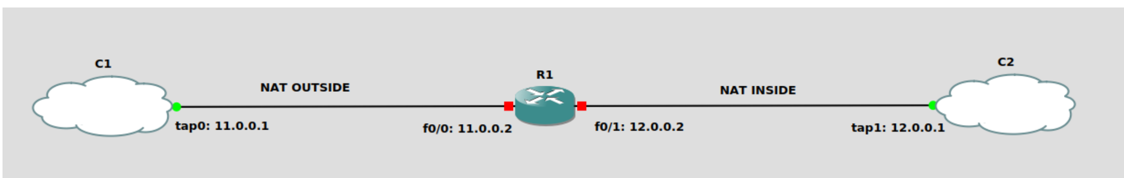


Figure 1: Implemented Topology

1.2 Configuration

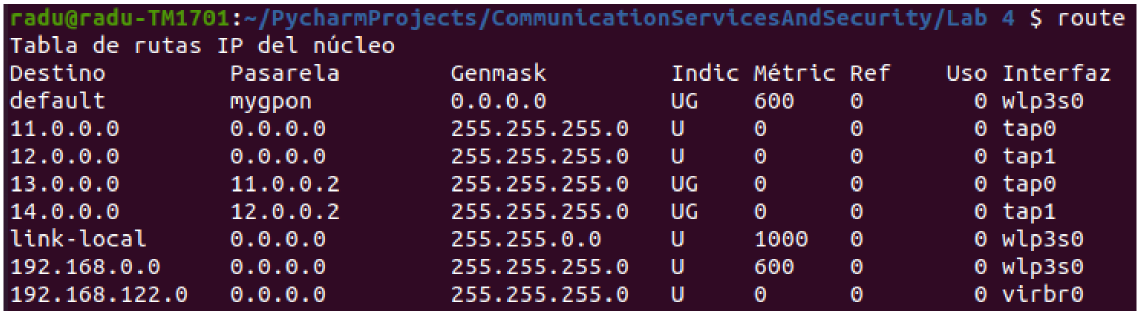
To implement the presented topology where added the following commands on the different elements.

1.2.1 Host

Is attached as “tab-script.sh” and configures the taps and adds routes.

```
#!/bin/bash
sudo tunctl -t tap0 -u radu
sudo tunctl -t tap1 -u radu
sudo ip link set tap0 up
sudo ip link set tap1 up
ip add add 11.0.0.1/24 dev tap0
ip add add 12.0.0.1/24 dev tap1
route add -net 13.0.0.0/24 gw 11.0.0.2 dev tap0
route add -net 14.0.0.0/24 gw 12.0.0.2 dev tap1
```

We used the following command for cheking:



```
radu@radu-TM1701:~/PycharmProjects/CommunicationServicesAndSecurity/Lab 4 $ route
Tabla de rutas IP del núcleo
Destino         Pasarela        Genmask         Indic Métric Ref    Uso Interfaz
default         mygpon          0.0.0.0         UG    600    0        0 wlp3s0
11.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 tap0
12.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 tap1
13.0.0.0        11.0.0.2        255.255.255.0   UG    0      0        0 tap0
14.0.0.0        12.0.0.2        255.255.255.0   UG    0      0        0 tap1
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 wlp3s0
192.168.0.0     0.0.0.0         255.255.255.0   U     600    0        0 wlp3s0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

Figure 2: Route command.

1.2.2 R1

1. Default Routes

```
ip route 13.0.0.0 255.0.0.0 12.0.0.1
ip route 14.0.0.0 255.0.0.0 11.0.0.1
```

2. Nat Configuration

```
ip nat inside source static 12.0.0.1 13.0.0.1
ip nat outside source static 11.0.0.1 14.0.0.1
```

3. Interaface Fast Etherent 0/0

```
ip address 11.0.0.2 255.255.255.0
ip nat outside
no shutdown
```

4. Interaface Fast Ethernet 0/1

```
ip address 12.0.0.2 255.255.255.0
ip nat inside
no shutdown
traffic-shape rate [Average rate (bps)] [bc (bits)] [be (bits)]
```

We used the following command for cheking:

```
R1#show traffic-shape

Interface   Fa0/1
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
-             1000000   100000 400000    400000    400       50000     -
R1#
```

Figure 3: Show traffic-shape command.

1.3 Traffic Generation

To generate the flow firs we donloaded from the provided link the following video:

Elecard 4K video about Tomsk, part 2

Encode CodecWorks
Decode MPEG Player
Analyze StreamEye

Duration, minutes: 3:33

Resolution	Format	Video bit rate	Recommended Hardware	Size	Link
3840x2160	MP4, HEVC	4.8 mbps	4-core Intel i7 2.5 GHz	100Mb	Download
1920x1080	MP4, HEVC	1.5 mbps	4-core Intel i7 2.5 GHz	31Mb	Download
1280x720	MP4, HEVC	0.8 mbps	4-core Intel i7 2.5 GHz	16Mb	Download

Figure 4: Transmitted video.

Then the transmission was done with the following commands in 2 different terminals on the host computer:

```
term1$ ffmpeg -re -i video_720p.mp4 -vcodec copy -an -sdp_file s.sdp
-f rtp rtp://13.0.0.1:5004
term2$ ffplay -protocol_whitelist file,udp,rtp -i s.sdp
```

1.4 Traffic Analysis

1.4.1 Traffic Shapping Plots

To solve the first part of this problem, the previous shown command on the interface FastEthernet 0/1:

```
traffic-shape rate [Average rate (bps)] [bc (bits)] [be (bits)]
```

Was modified with the following values on different iterations:

```
traffic-shape rate 1000000 400000 40000
traffic-shape rate 800000 4000 4000
```

And after without traffic shape. The captured files are processed in the shell script “rate.sh” and the results are used in the “Plot Generator.ipynb” which generates the following plots:

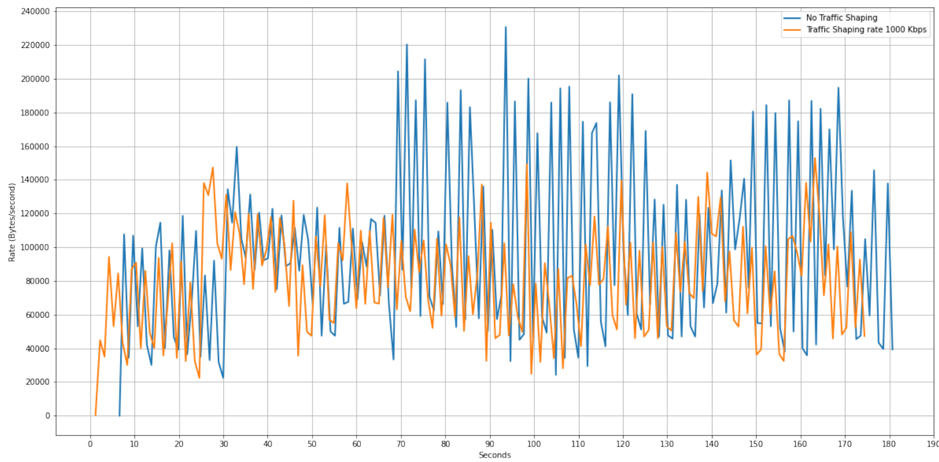


Figure 5: Traffic-Shaping 1000 Kbps.

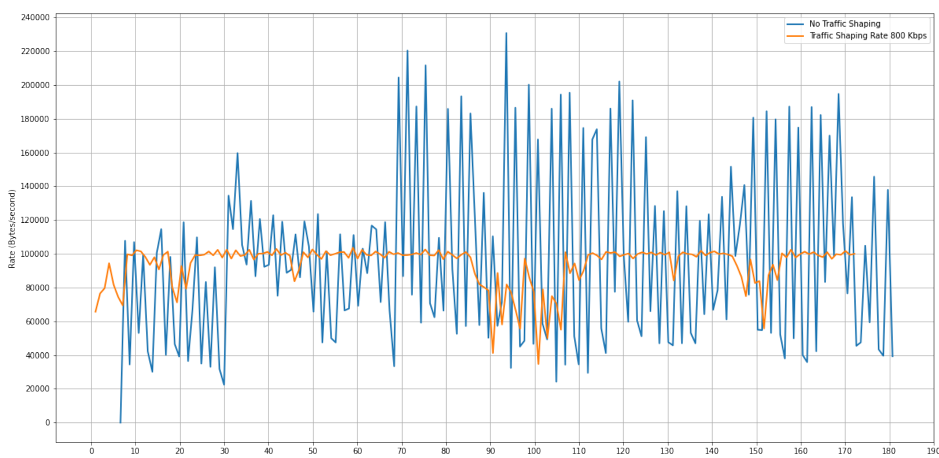


Figure 6: Traffic-Shaping 800 Kbps.

From the previous plot we can observe that both traffic-shaping have a more consistent rate than without traffic-shapping. On the other hand, in the second plott which shows the result of applying traffic-shaping with an average rate of 800 Kbps, has a more consistent average (on

100.000 bytes/second) than traffic-shapping with an average rate of 1000 Kbps.

1.4.2 Lost Frames

In order to compute the lost frames at the player during the first 30 seconds in the different cases, we also modified the following command for each case:

```
traffic-shape rate [Average rate (bps)] [bc (bits)] [be (bits)]
```

Then, the output of the **ffplay** was saved in different text files for this exercices requirements we implement a script "rtp_script.py" which computes the total of RTP missed packets from the files generated.

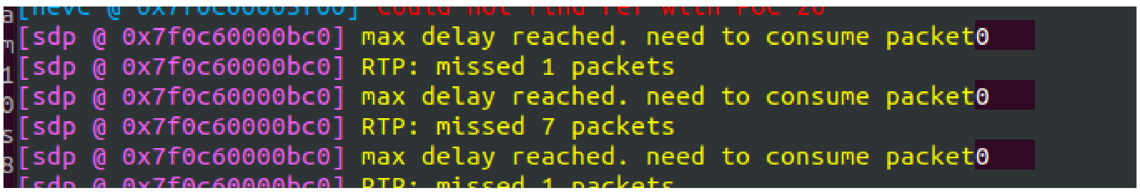


Figure 7: RTP missed packets example.

The following table shows the lost frames at the player for the different proposed cases:

Average rate (bps)	**bc=be (bits)**	**Lost Frames**
10^6	$4 \cdot 10^5$	13
10^6	$4 \cdot 10^4$	18
10^6	$4 \cdot 10^3$	34
$8 \cdot 10^5$	$4 \cdot 10^5$	44
$8 \cdot 10^5$	$4 \cdot 10^4$	86
$8 \cdot 10^5$	$4 \cdot 10^3$	128
$5 \cdot 10^5$	$4 \cdot 10^5$	287
$5 \cdot 10^5$	$4 \cdot 10^4$	332
$5 \cdot 10^5$	$4 \cdot 10^3$	352

2 Problem

RSVP configuration

- **Reserve 400 Kbps FF style from 11.0.0.1:48823 to 13.0.0.1:5004**
- **Reserve 200 Kbps FF style from 12.0.0.1:40258 to 14.0.0.1:5004**

Traffic patterns. Create the following streams:

1. **A 1 Mbps ping flow from C1 to R3, with on-off pulses of 1 second each at Serial Line**
2. **1 Mbps continuous video streaming from 11.0.0.1:48823 to 13.0.0.1:5004 (Use packETHcli with captured packet)**
3. **1 Mbps continuous video streaming from 12.0.0.1:40258 to 14.0.0.1:5004 (Use packETHcli with captured packet)**

In the following sections we will explain the steps followed to solve the different parts of this problem.

2.1 Topology

Figure 8 shows the structure of the implemented topology.

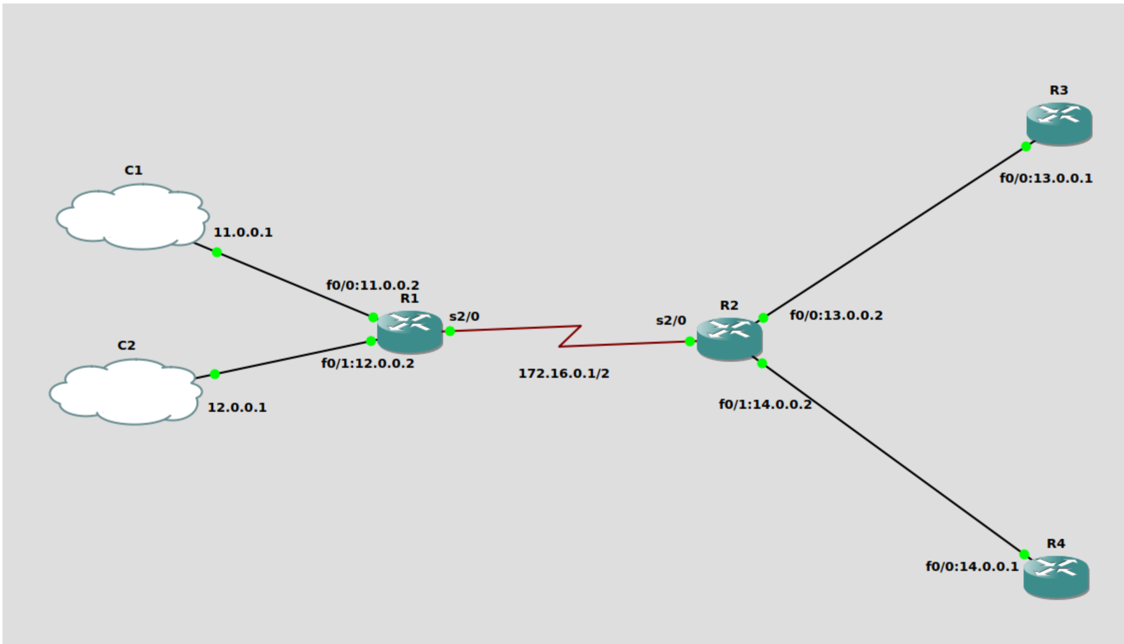


Figure 8: Implemented Topology

2.2 Configuration

To implement the presented topology where added the following commands on the different elements.

2.2.1 Host

Is attached as “tab-script.sh” and configures the taps and adds routes.

```
#!/bin/bash
sudo tunctl -t tap0 -u radu
sudo tunctl -t tap1 -u radu
sudo ip link set tap0 up
sudo ip link set tap1 up
ip add add 11.0.0.1/24 dev tap0
ip add add 12.0.0.1/24 dev tap1
route add -net 13.0.0.0/24 gw 11.0.0.2 dev tap0
route add -net 14.0.0.0/24 gw 12.0.0.2 dev tap1
```

As in the previous exercice, we used the “route” command for cheking.

2.2.2 R1

1. Default Routes

```
ip route 13.0.0.0 255.255.255.0 172.16.0.2
ip route 14.0.0.0 255.255.255.0 172.16.0.2
```

2. RSVP (Sender)

```
ip rsvp sender 13.0.0.1 11.0.0.1 UDP 5004 48823 11.0.0.1 FastEthernet0/0 400 10
ip rsvp sender 14.0.0.1 12.0.0.1 UDP 5004 40258 12.0.0.1 FastEthernet0/1 200 10
```

3. Interaface Fast Etherent 0/0

```
ip address 11.0.0.2 255.255.255.0
ip rsvp bandwidth 1200 1200
no shutdown
```

4. Interaface Fast Ethernet 0/1

```
ip address 12.0.0.2 255.255.255.0
ip rsvp bandwidth 1200 1200
no shutdown
```

5. Interaface Serial 2/0

```
ip address 172.16.0.1 255.255.255.0
fair-queue 4096 4096 1000
ip rsvp bandwidth 1200 1200
no shutdown
```

2.2.3 R2

1. Default Routes

```
ip route 11.0.0.0 255.255.255.0 172.16.0.1
ip route 12.0.0.0 255.255.255.0 172.16.0.1
```

2. Interaface Fast Etherent 0/0

```
ip address 13.0.0.2 255.255.255.0
ip rsvp bandwidth 1200 1200
no shutdown
```

3. Interaface Fast Ethernet 0/1

```
ip address 14.0.0.2 255.255.255.0
ip rsvp bandwidth 1200 1200
no shutdown
```

4. Interaface Serial 2/0

```
ip address 172.16.0.2 255.255.255.0
ip rsvp bandwidth 1200 1200
fair-queue 4096 4096 1000
no shutdown
```

2.2.4 R3

1. Default Routes

```
ip route 0.0.0.0 0.0.0.0 13.0.0.2
```

2. RSVP (Reservation)

```
ip rsvp reservation-host 13.0.0.1 11.0.0.1 UDP 5004 48823 FF LOAD 400 10
```

3. Interaface Fast Etherent 0/0

```
ip address 13.0.0.1 255.255.255.0
ip rsvp bandwidth 1200 1200
no shutdown
```

2.2.5 R4

1. Default Routes

```
ip route 0.0.0.0 0.0.0.0 14.0.0.2
```

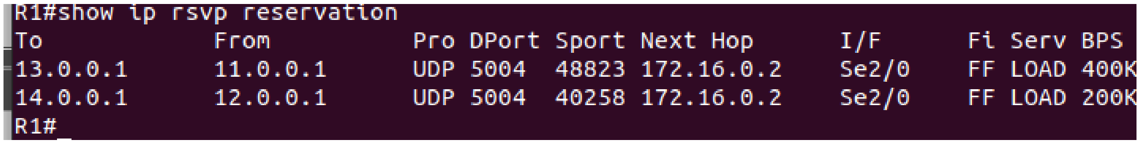
2. RSVP (Reservation)

```
ip rsvp reservation-host 14.0.0.1 12.0.0.1 UDP 5004 40258 FF LOAD 200 10
```

3. Interaface Fast Etherent 0/0

```
ip address 14.0.0.1 255.255.255.0
ip rsvp bandwidth 1200 1200
no shutdown
```

The following image shows the results of the command that displays the RSVP reservations.



```
R1#show ip rsvp reservation
To             From           Pro DPort Sport Next Hop       I/F      Fi Serv BPS
13.0.0.1       11.0.0.1       UDP 5004  48823 172.16.0.2     Se2/0    FF LOAD 400K
14.0.0.1       12.0.0.1       UDP 5004  40258 172.16.0.2     Se2/0    FF LOAD 200K
R1#_
```

Figure 9: Show ip rsvp reservation command.

3 Traffic Patterns

3.1 Patterns generation

To generate the traffic pattern 1 we used the following script wich is attached as "pattern1.sh":

```
while true;
do
  sudo ./packETHcli -i tap0 -d 8336 -m 2 -f 'ping-1000-tap0.pcap' -t 1;
   sleep 1;
done
```

It generates traffic from tap0 to R3 of 1Mbps. To generate traffic of 1Mbps it needs to send packets every 8336 ms. The used formula:

$$RATE = \frac{1048 \times 8}{8336 \times 10^-6}$$

To achieve the on-off pulses we used version 2.1 of the **packETHcli** which allows us to control the execution time with the **-t** parameter. The version is free available at Github. To generate the patterns 2 and 3 the following files were donwloaded from Captures:

- udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap
- udp-size_1500-port_dest_5004_ip_14.0.0.1.pcap

And the following command using **packETHcli**:

- sudo ./packETHcli -i tap0 -m 2 -d 10500 -n 0 -f udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap
- sudo ./packETHcli -i tap1 -m 2 -d 12000 -n 0 -f udp-size_1500-port_dest_5004_ip_14.0.0.1.pcap

3.2 Traffic Pattern 1

We captured the traffic between R1 and R2 throw the serial interface s2/0 and stored it into "Paterrn1.pcap" while running the "pattern1.sh" script. To generate the required plots we used the **Statistics - I/O Graph** function of Wireshark. The averaging slots time was changed using the interval option:

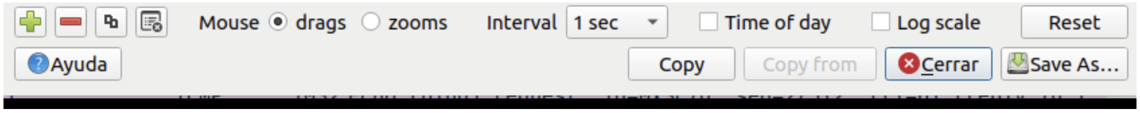


Figure 10: Interval option.

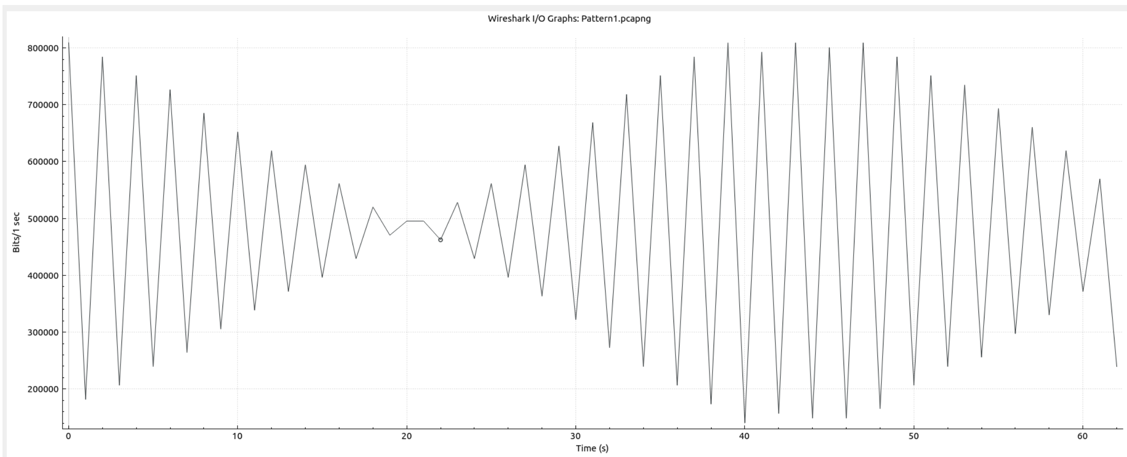


Figure 11: Traffic Pattern 1 - Avg slot 1s.

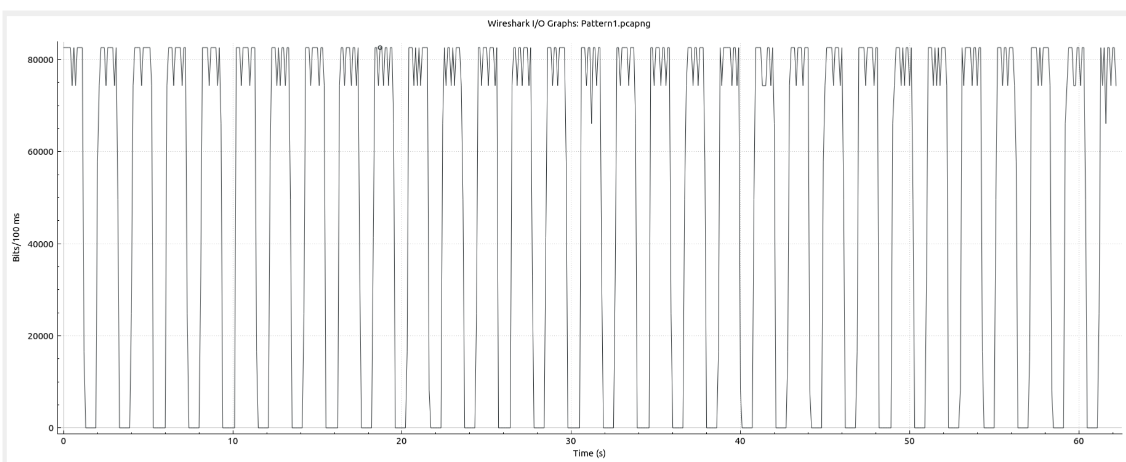


Figure 12: Traffic Pattern 1 - Avg slot 100ms.

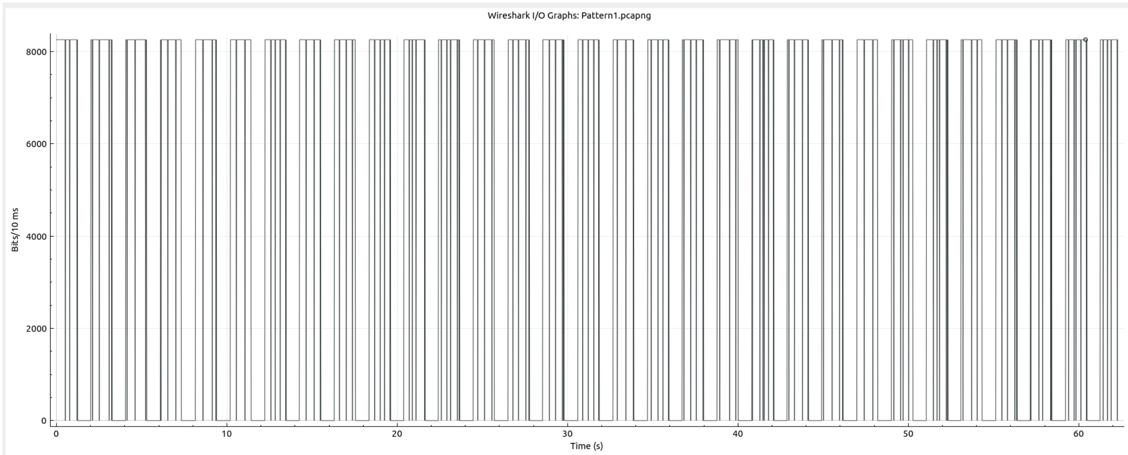


Figure 13: Traffic Pattern 1 - Avg slot 10ms.

In the previous plots we can see how the ping performed correctly obtaining the desired average bit rate. Also, the plots show that the on-off pulses using the **-t** param from the packETHcli and the "sleep 1" in the pattern script were implemented correctly.

3.3 Traffic All Patterns

We stored the captured traffic with the 3 patterns running into "AllPatterns.pcapng". To generate the plots we used the same method as in the previous section but is shown in the Figure 14 were used the following filters to differentiate between patterns:

- Pattern 1

```
icmp && ip.addr == 11.0.0.1 && ip.dst == 13.0.0.1
```

- Pattern 2

```
udp && ip.addr == 11.0.0.1 && ip.dst == 13.0.0.1
```

- Pattern 3

```
ip.addr == 12.0.0.1 && ip.dst == 14.0.0.1
```

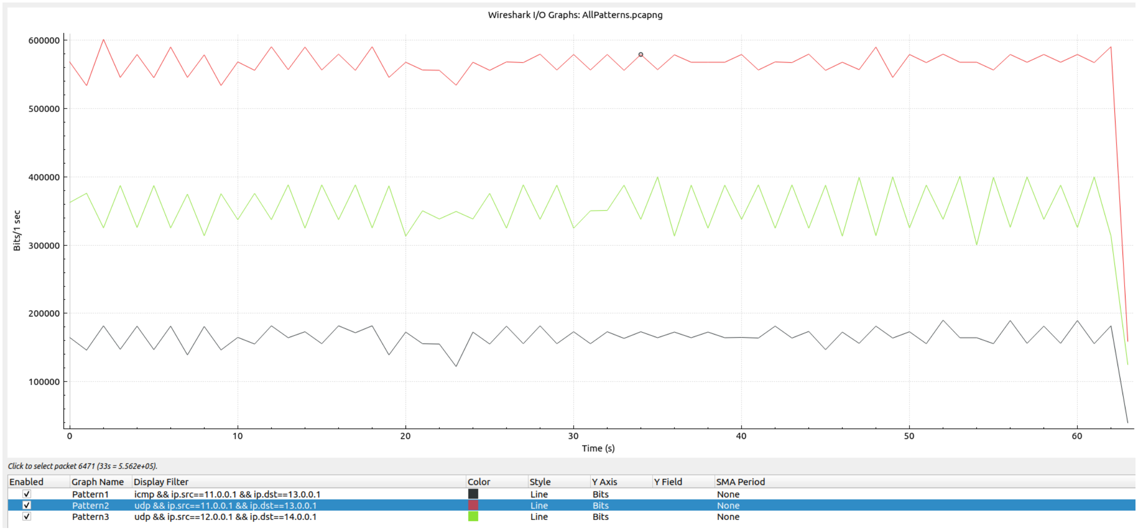


Figure 14: Traffic All Patterns - Avg slot 1s.

Figure 15: Traffic All Patterns - Avg slot 100ms.

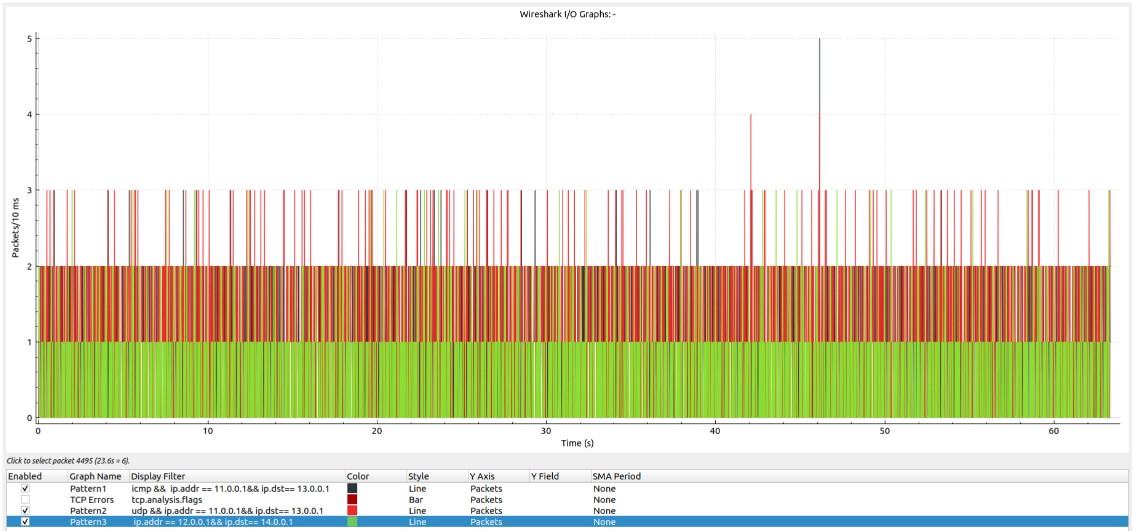


Figure 16: Traffic All Patterns - Avg slot 10ms.

From the previous plots, according to the implemented patterns and network topology, we can see that the reservation protocol is working as it's desired according to the topology specifications. The plots show as that the average minimum bit rate of the traffic is over the reserve and the remaining bandwidth is divided between the 3 traffic patterns.