



University of Lleida

Master's Degree in Informatics Engineering

Higher Polytechnic School

Exercise 4

ICT Project: Communication Services and Security

Cèsar Fernández Camón

Albert Pérez Datsira

May 1, 2021

Table of contents

1	Introduction	1
2	Problem 1	2
2.1	Setup	3
2.2	Part 1	6
2.3	Part 2	8
3	Problem 2	9
3.1	Setup	10
3.2	Results	15

List of Tables

1	Cases to evaluate	2
2	Tap interface configuration commands	3
3	Network configuration commads	4
4	R1 router configuration commands	4
5	Packet lost by cases	8
6	R1 router configuration commands	11
7	R2 router configuration commands	12
8	R3 router configuration commands	12
9	R4 router configuration commands	13

List of Figures

1	Problem 1 traffic plot	2
2	Problem 1 network topology	2
3	Check tap interfaces configuration	3
4	Problem 1 ip route configuration	4
5	Problem 1 NAT pings	5
6	Traffic shapping comparison for 1000 kbps average rate	6
7	Traffic shapping comparison for 800 kbps average rate	7
8	Problem 2 network topology	9
9	Traffic pattern 1 shell script code	13
10	Traffic pattern 1 execution	14
11	Traffic pattern 2 execution	14
12	Traffic pattern 3 execution	14
13	R1 router rsvp configuration	15
14	R2 router rsvp configuration	15
15	R3 router rsvp configuration	15
16	R4 router rsvp configuration	16
17	Traffic pattern 1 at 1s	16
18	Traffic pattern 1 at 10ms	17
19	Traffic pattern 1 at 100ms	17
20	3 Patterns active traffic at 1s	18
21	3 Patterns active traffic at 10ms	18
22	3 Patterns active traffic at 100ms	19

1 Introduction

This assignment is focused on the one hand on learning how to install, configure and use *GNS3* (Graphical Network Simulator) and defining scenarios to be used through the example.

On the other hand, to put into practice the knowledge about the *Quality of Service* (QoS) and to be able to draw results from the analysis and reading of the network frames using tools such as *Wireshark*, but also to draw conclusions about the behavior of the simulations implemented.

2 Problem 1

Download [this video](#) (ElecCard 4K video about Tomsk, part2, Bit rate: 0.8Mbps, 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.

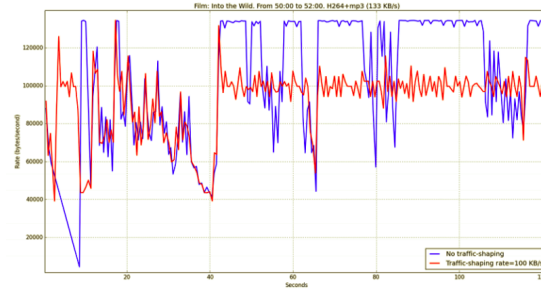


Figure 1: Problem 1 traffic plot

2. Compute the number of lost frames at the media player during the first 30 seconds in the following cases:

Average rate (bps)	$bc = be$ (bits)
10^6	$4 \cdot 10^5$
10^6	$4 \cdot 10^4$
10^6	$4 \cdot 10^3$
$8 \cdot 10^5$	$4 \cdot 10^5$
$8 \cdot 10^5$	$4 \cdot 10^4$
$8 \cdot 10^5$	$4 \cdot 10^3$
$5 \cdot 10^5$	$4 \cdot 10^5$
$5 \cdot 10^5$	$4 \cdot 10^4$
$5 \cdot 10^5$	$4 \cdot 10^3$

Table 1: Cases to evaluate

All points based on the following topology scenario:

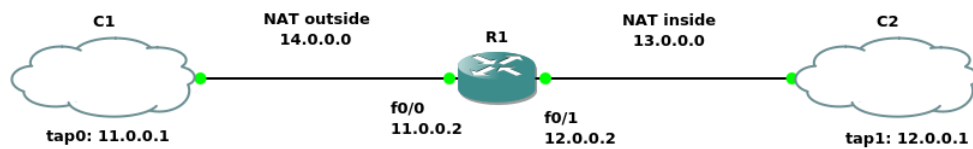


Figure 2: Problem 1 network topology

2.1 Setup

As can be seen, the problem is intended to demonstrate how traffic formation works. The given topology is showed in the figure 2 on page 2. We can see there are two *tap* interfaces acting as clouds, each one connected to the same router (R1) through a FastEthernet link.

In addition, the *Network Address Translation* (NAT) is configured to allow traffic between both interfaces through the router.

Furthermore, the token bucket configuration is set to shape the traffic that exceeds the burst size. In this case, the exceed bucket has the same size of the conforming one.

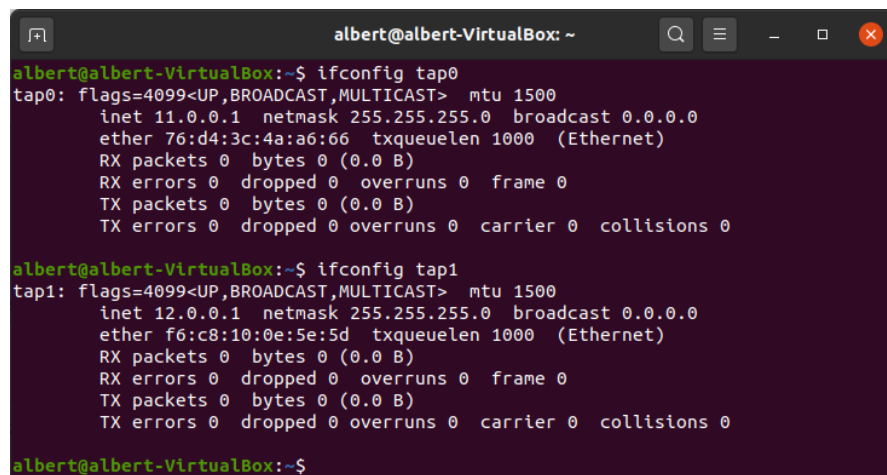
Nextly, there are the configurations followed to build the topology scenario as well as the way how was traffic generation done.

2.1.1 Host computer

Tap interfaces

	Commands
Configure Tap0	<pre>sudo tuncctl -t tap0 -u <username> sudo ip link set tap0 ip sudo ip add add 11.0.0.1/24 dev tap0</pre>
Configure Tap1	<pre>sudo tuncctl -t tap1 -u <username> sudo ip link set tap1 ip sudo ip add add 12.0.0.1/24 dev tap1</pre>

Table 2: Tap interface configuration commands



```
albert@albert-VirtualBox: ~
albert@albert-VirtualBox:~$ ifconfig tap0
tap0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 11.0.0.1 netmask 255.255.255.0 broadcast 0.0.0.0
    ether 76:d4:3c:4a:a6:66 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

albert@albert-VirtualBox:~$ ifconfig tap1
tap1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 12.0.0.1 netmask 255.255.255.0 broadcast 0.0.0.0
    ether f6:c8:10:0e:5e:5d txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

albert@albert-VirtualBox:~$
```

Figure 3: Check tap interfaces configuration

Network routes

	Commands
Add ip route gateways	<pre>sudo route add -net 13.0.0.0/24 gw 11.0.0.2 sudo route add -net 14.0.0.0/24 gw 12.0.0.2</pre>

Table 3: Network configuration commads

```

albert@albert-VirtualBox: ~/Desktop
albert@albert-VirtualBox:~/Desktop$ ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
11.0.0.0/24 dev tap0 proto kernel scope link src 11.0.0.1
12.0.0.0/24 dev tap1 proto kernel scope link src 12.0.0.1
13.0.0.0/24 via 11.0.0.2 dev tap0
14.0.0.0/24 via 12.0.0.2 dev tap1
169.254.0.0/16 dev enp0s3 scope link metric 1000
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 linkdown
albert@albert-VirtualBox:~/Desktop$

```

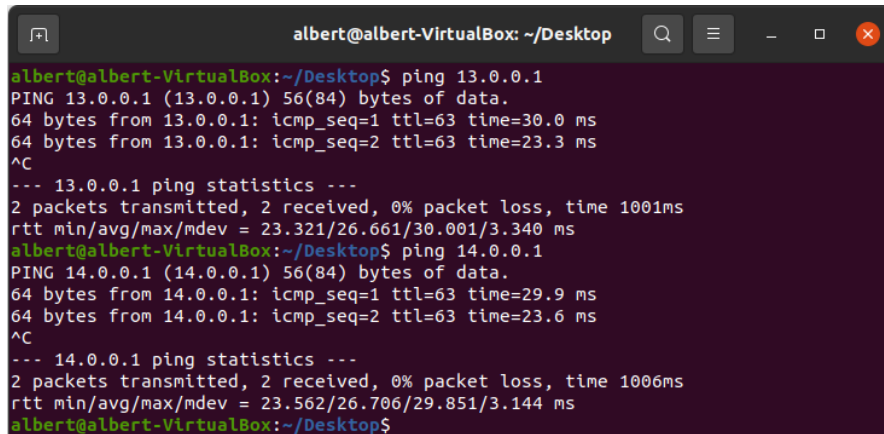
Figure 4: Problem 1 ip route configuration

2.1.2 Router 1

	Commands
Interface FastEthernet 0/0	<pre>ip address 11.0.0.2 255.255.255.0 no shutdown</pre>
Interface FastEthernet 0/1	<pre>ip address 12.0.0.2 255.255.255.0 no shutdown traffic-shape rate <Avg rate (bps)> <bc (bits)> <be (bits)></pre>
NAT configuration	<pre>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</pre>
Add ip routes	<pre>ip route 13.0.0.0 255.255.255.0 12.0.0.1 ip route 14.0.0.0 255.255.255.0 11.0.0.1</pre>

Table 4: R1 router configuration commands

where *traffic-shape* values will depend on the problem point sentence you may see at the begining.

A terminal window titled 'albert@albert-VirtualBox: ~/Desktop' showing the output of two ping commands. The first command is 'ping 13.0.0.1', which shows two successful pings with times of 30.0 ms and 23.3 ms, followed by statistics: 2 packets transmitted, 2 received, 0% packet loss, time 1001ms, and rtt min/avg/max/mdev = 23.321/26.661/30.001/3.340 ms. The second command is 'ping 14.0.0.1', which also shows two successful pings with times of 29.9 ms and 23.6 ms, followed by statistics: 2 packets transmitted, 2 received, 0% packet loss, time 1006ms, and rtt min/avg/max/mdev = 23.562/26.706/29.851/3.144 ms.

```
albert@albert-VirtualBox:~/Desktop$ ping 13.0.0.1
PING 13.0.0.1 (13.0.0.1) 56(84) bytes of data.
64 bytes from 13.0.0.1: icmp_seq=1 ttl=63 time=30.0 ms
64 bytes from 13.0.0.1: icmp_seq=2 ttl=63 time=23.3 ms
^C
--- 13.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 23.321/26.661/30.001/3.340 ms
albert@albert-VirtualBox:~/Desktop$ ping 14.0.0.1
PING 14.0.0.1 (14.0.0.1) 56(84) bytes of data.
64 bytes from 14.0.0.1: icmp_seq=1 ttl=63 time=29.9 ms
64 bytes from 14.0.0.1: icmp_seq=2 ttl=63 time=23.6 ms
^C
--- 14.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 23.562/26.706/29.851/3.144 ms
albert@albert-VirtualBox:~/Desktop$
```

Figure 5: Problem 1 NAT pings

Once, the network route and R1 router configuration are set up, it is possible to check if all is running as expected by carrying out specific pings showed in the previous figure 5.

2.1.3 Traffic generation

To generate the traffic flow from the video provided we will use *ffmpeg*¹, a multimedia framework to record, convert and stream audio and video.

Once this software is installed in our machine, is able to run the following command in order to transmute the video to the 13.0.0.1 ip destination over the UDP² 5004 port:

- **ffmpeg -re -i <video infile> -vcodec copy -an -sdp_file <outfile .sdp> -f rtp rtp://13.0.0.1:5004**

Moreover, we must use for the second part of the problem a portable media player included into the *FFmpeg* libraries called *ffplay*, mostly used as a testbed.

Hence, will be able to compute the packets lost over the transmissions done, as we will be able to record the frames by using the following command:

- **ffplay -protocol_whitelist file,udp,rtp -i <file .sdp> 2> <log file>**

take a look the output must be redirected to a log file in order to keep the frames record for getting afterwards the packets lost.

¹<http://ffmpeg.org/>

²User Datagram Protocol

2.2 Part 1

In order to obtain the results, the video set for the problem was transmitted to C2 via the R1 router, by running the traffic generation *ffmpeg* command specified previously.

Then, to test the traffic formation, the video was transmitted in different token bucket configurations, the ones specified in the problem statement.

The configuration for the token bucket algorithms for traffic shaping with the requested average bit rates (1Mbps and 0.8Mbps), were set up by using the *traffic-shape* cisco-router command on the R1 router FastEthernet 0/1 interface, where its usage is specified in the table 4 on page 4.

Also, was obtained the no shapping record to compare with the shaped results obtained as you may see in the following resulting plots.

All by using *Wireshark* to capture the transmission traffic at R1 router FastEthernet 0/1 interface, a shell script using the *tshark* application to calculate the rate Bytes/second from the capture files for each flow and finally a python script to create the corresponding plots to better visualize them in a timeline as the Wireshark I/O graphs functionality doesn't let you combine multiple captures.

Must be said, all the plots are using Bits per second (bps) as it is the standard to measure data speed.

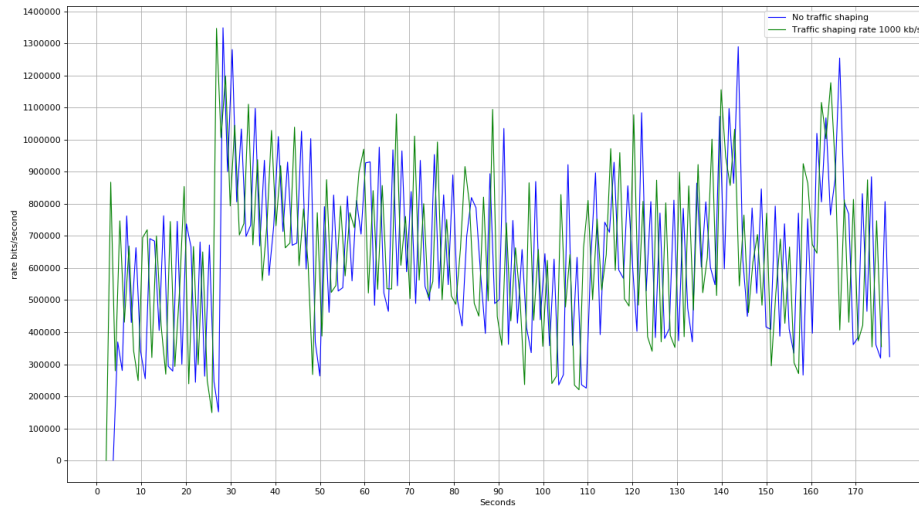


Figure 6: Traffic shapping comparison for 1000 kbps average rate

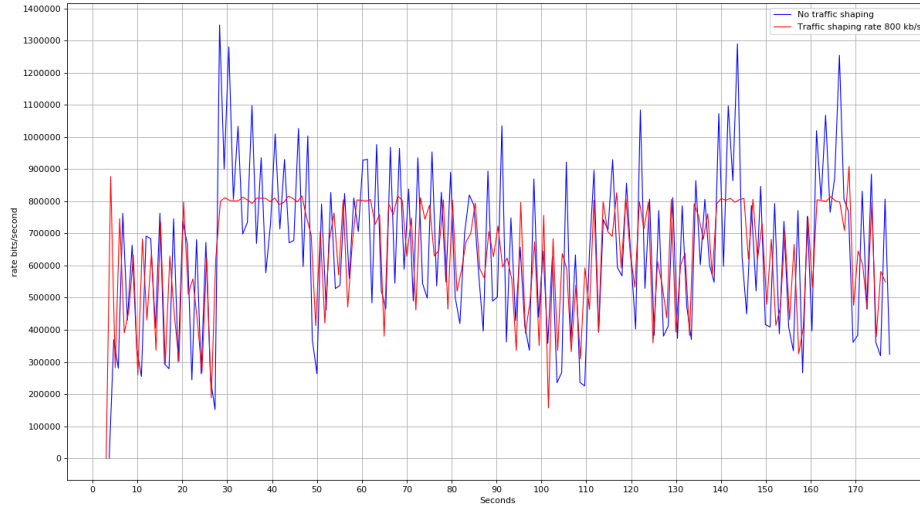


Figure 7: Traffic shapping comparison for 800 kbps average rate

Once the results were obtained and by comparing then, we can see that without the traffic shaping the transmission rates are inconsistent and unstable with a lot of dispersion, as seen in figure 6 on page 6.

In a different way it matters if we have the traffic modeling activated because the average rate tends to be around the value configured, because of as expected traffic shapping allows for much more constant rates.

Thus we observe that the shape of the plots tends to moderate and be more uniform when the traffic is more shapped as seen in the figure 7, which implies more stability over the transmission.

2.3 Part 2

To carry out this part, as mentioned in the traffic generation section, we have used the *ffplay* media player and redirected its output to get the log files in order to count the lost packets.

So basically, we executed both commands the *ffmpeg* to generate the transmission traffic and the *ffplay* to get the streaming, and finally once obtained the logs files a python script was used to count the packets lost by getting the RTP³ missed packets.

You may consult the results for each case on the following table:

Average rate (bps)	bc=be (bist)	Packets lost
10^6	$4 * 10^5$	85
10^6	$4 * 10^4$	86
10^6	$4 * 10^3$	86
$8 * 10^5$	$4 * 10^5$	112
$8 * 10^5$	$4 * 10^4$	161
$8 * 10^5$	$4 * 10^3$	166
$5 * 10^5$	$4 * 10^5$	280
$5 * 10^5$	$4 * 10^4$	372
$5 * 10^5$	$4 * 10^3$	377

Table 5: Packet lost by cases

If we take a closer look at the results, we can conclude that its performance is as expected.

Mainly because as traffic becomes more shaped, therefore the average bit rate is adjusted, more packets are lost because of this.

³Real-time Transport Protocol

3 Problem 2

Build a network as in figure 8.

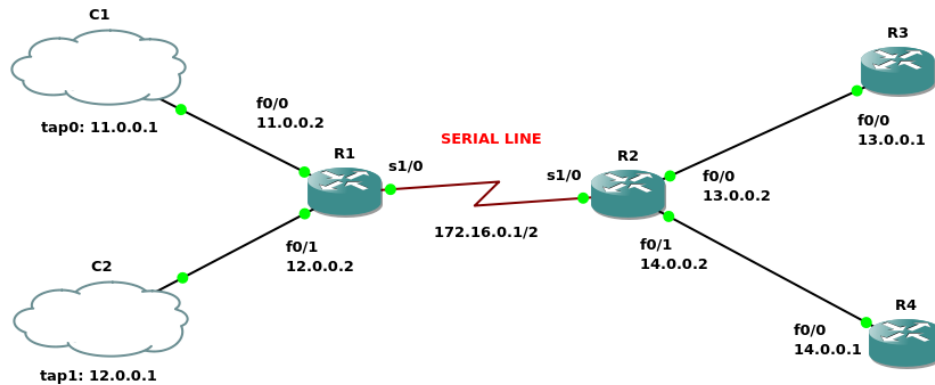


Figure 8: Problem 2 network topology

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)⁴
3. 1 Mbps continuous video streaming from 12.0.0.1:40258 to 14.0.0.1:5004 (Use packETHcli with captured packet)

Deliver

- R1 console screenshot showing RSVP⁵ reservations
- Show the command to generate traffic pattern 1. Capture on Serial Link (only this traffic active) and plot throughput at different averaging slots (1", 0.1" and 0.0.1"). Use the same **ping** command for all the three plots.
- Repeat previous item having 3 traffic patterns active
- Comment each point

⁴Find de captured packets on folder Lab Work → 2 QoS → Captures

⁵Resource Reservation Protocol

3.1 Setup

This problem aims to demonstrate, test, and comprheends how RSPV works, which is an implementation of *Integrated Services* as a transport layer protocol featured as a *signaling* one, used to reserve network resources and enable running Internet applications to gain *Quality of Service*.

As you can see there are two *tap* interfaces (C1 and C2), connected through a router (R1) via FastEthernet interface each one.

Then, R1 router is connected to R2 via a serial link. And finally, R2 it is also connected through FastEthernet link to two more routers (R3 and R4).

3.1.1 Host computer

Tap interfaces

The configuration of the *tap* interfaces (clouds) is the same as in problem 1.

You can consult the commands in the table 2 on page 3, as well as how should like the host interface configuration in the figure 3 on page 3.

Network routes

Regards the net configuration, corresponds exactly as the one implemented in problem 1.

You can consult the commands in the table 3 on page 4, as well as how should look like the resulting configuration in the figure 4 on page 4.

3.1.2 Routers

Router 1

	Commands
Interface FastEthernet 0/0	ip address 11.0.0.2 255.255.255.0 no shutdown ip rsvp bandwidth 1200 1200
Interface FastEthernet 0/1	ip address 12.0.0.2 255.255.255.0 no shutdown ip rsvp bandwidth 1200 1200
Interface Serial 1/0	ip address 172.16.0.1 255.255.255.0 ip rsvp bandwidth 1150 1150 fair-queue 4096 4096 1000 serial restart-delay 0 no shutdown
Ip route configuration	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
RSVP sender configuration	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

Table 6: R1 router configuration commands

Look at the *rsvp bandwidth* on the serial interface, that must be used because of the maximum amount of reservable bandwidth per interface, which is 75% of interface rate as default and as a maximum (Kbps).

This phenomenon happens also on the serial interface of R2 router, you may see its configuration on the next page.

Router 2

	Commands
Interface FastEthernet 0/0	ip address 13.0.0.2 255.255.255.0 no shutdown ip rsvp bandwidth 1200 1200
Interface FastEthernet 0/1	ip address 14.0.0.2 255.255.255.0 no shutdown ip rsvp bandwidth 1200 1200
Interface Serial 1/0	ip address 172.16.0.2 255.255.255.0 ip rsvp bandwidth 1150 1150 fair-queue 4096 4096 1000 serial restart-delay 0 no shutdown
Ip route configuration	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

Table 7: R2 router configuration commands

Router 3

	Commands
Interface FastEthernet 0/0	ip address 13.0.0.1 255.255.255.0 no shutdown ip rsvp bandwidth 1200 1200
Default gateway	ip route 0.0.0.0 0.0.0.0 13.0.0.2
Reservation	ip rsvp reservation-host 13.0.0.1 11.0.0.1 udp 5004 48823 ff load 400 10

Table 8: R3 router configuration commands

Router 4

	Commands
Interface FastEthernet 0/0	ip address 14.0.0.1 255.255.255.0 no shutdown ip rsvp bandwidth 1200 1200
Default gateway	ip route 0.0.0.0 0.0.0.0 14.0.0.2
Reservation	ip rsvp reservation-host 14.0.0.1 12.0.0.1 udp 5004 40258 ff load 200 10

Table 9: R4 router configuration commands

3.1.3 Traffic generation

Three different traffic patterns should be generated. So regarding the first one, was build a shell script as it is not enough to make a traffic flow because of must be assured the on-off pulses.

That's the reason why, the shell script must provide an infinite *loop* with a *sleep* of 1 second between traffic transmissions.

As regards the traffic command, it uses the *packETHcli* to generate at each step one packet transmission by applying the optional argument *-t* set as 1 packet.

```
while true;
do
  sudo ./packETHcli -i tap0 -d 6000 -m 2 -f ping-1000-tap0.pcap -t 1;
  sleep 1;
done;
```

Figure 9: Traffic pattern 1 shell script code

On the other hand, there are the two last patterns which stands in the same way and also using the *packETHcli*. The following are the commands used:

- `sudo ./packETHcli -i tap0 -d 9100 -n 0 -m 2 -f udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap`
- `sudo ./packETHcli -i tap1 -d 9800 -n 0 -m 2 -f udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap`

Take a look that must be adjust for each pattern command the *delay* option (-d) as must be assure the 1Mbps statement requirement, because it will affect directly on the resulting traffic rate, and its value will depend on the machine.

In this specific case, that a virtual machine was used to run up all the problems, the delays showed are the ones fitting perfectly in the environment. You may see the 1Mbps rate for each traffic pattern in the figures below:


```
albert@albert-VirtualBox: ~/Desktop/shared/packETH-mast...
albert@albert-VirtualBox:~/Desktop/shared/packETH-master/cli$ ./pattern1.sh
Sent 118 packets on tap0; 1042 bytes packet length; 118 packets/s; 0.983 Mbit/s data rate; 1.006 Mbit/s link utilization
-----
Sent 118 packets on tap0 in 1.001235 second(s).
-----
Sent 116 packets on tap0; 1042 bytes packet length; 116 packets/s; 0.966 Mbit/s data rate; 0.989 Mbit/s link utilization
-----
Sent 116 packets on tap0 in 1.003993 second(s).
-----
Sent 121 packets on tap0; 1042 bytes packet length; 121 packets/s; 1.008 Mbit/s data rate; 1.031 Mbit/s link utilization
```

Figure 10: Traffic pattern 1 execution

```
albert@albert-VirtualBox: ~/Desktop/shared/packETH-mast...
albert@albert-VirtualBox:~/Desktop/shared/packETH-master/cli$ sudo ./packETHcli
-i tap0 -n 0 -d 9100 -m 2 -f udp-size_1356-port_dest_5004_ip_13.0.0.1.pcap
Sent 90 packets on tap0; 1370 bytes packet length; 90 packets/s; 0.986 Mbit/s data rate; 1.003 Mbit/s link utilization
Sent 182 packets on tap0; 1370 bytes packet length; 92 packets/s; 1.008 Mbit/s data rate; 1.025 Mbit/s link utilization
Sent 276 packets on tap0; 1370 bytes packet length; 94 packets/s; 1.030 Mbit/s data rate; 1.048 Mbit/s link utilization
Sent 371 packets on tap0; 1370 bytes packet length; 95 packets/s; 1.041 Mbit/s data rate; 1.059 Mbit/s link utilization
Sent 461 packets on tap0; 1370 bytes packet length; 90 packets/s; 0.986 Mbit/s data rate; 1.003 Mbit/s link utilization
Sent 551 packets on tap0; 1370 bytes packet length; 90 packets/s; 0.986 Mbit/s
```

Figure 11: Traffic pattern 2 execution

```
albert@albert-VirtualBox: ~/Desktop/shared/packETH-mast...
albert@albert-VirtualBox:~/Desktop/shared/packETH-master/cli$ sudo ./packETHcli
-i tap1 -n 0 -d 9800 -m 2 -f udp-size_1500-port_dest_5004_ip_14.0.0.1.pcap
Sent 90 packets on tap1; 1514 bytes packet length; 90 packets/s; 1.090 Mbit/s data rate; 1.107 Mbit/s link utilization
Sent 178 packets on tap1; 1514 bytes packet length; 88 packets/s; 1.065 Mbit/s data rate; 1.082 Mbit/s link utilization
Sent 264 packets on tap1; 1514 bytes packet length; 86 packets/s; 1.041 Mbit/s data rate; 1.058 Mbit/s link utilization
Sent 348 packets on tap1; 1514 bytes packet length; 84 packets/s; 1.017 Mbit/s data rate; 1.033 Mbit/s link utilization
Sent 433 packets on tap1; 1514 bytes packet length; 85 packets/s; 1.029 Mbit/s data rate; 1.045 Mbit/s link utilization
Sent 522 packets on tap1; 1514 bytes packet length; 89 packets/s; 1.077 Mbit/s
```

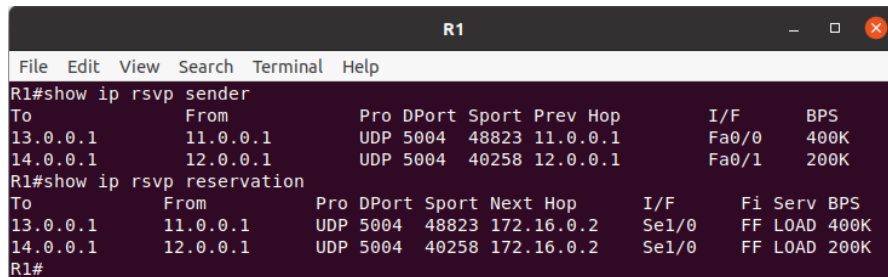
Figure 12: Traffic pattern 3 execution

3.2 Results

RSVP reservations

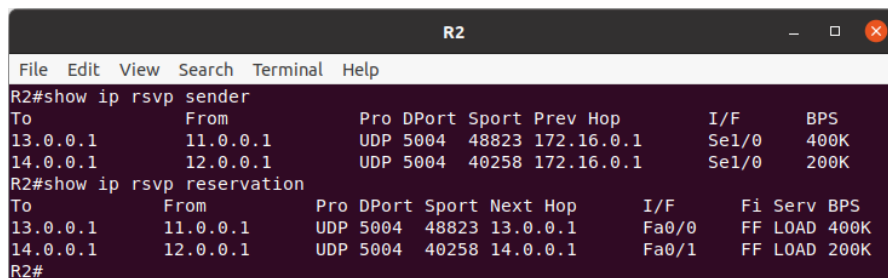
As described in the statement, two reservations must be made. So, after configuring them all, the following figures show the expected RSVP configuration for each router to confirm which active flows exist that have made a resource reservation.

In addition, you may see how all the *Next Hops* are correctly set for each case, as well as the R2 router getting the sender and reservations, but also the R3 and R4 respectively their ones too.



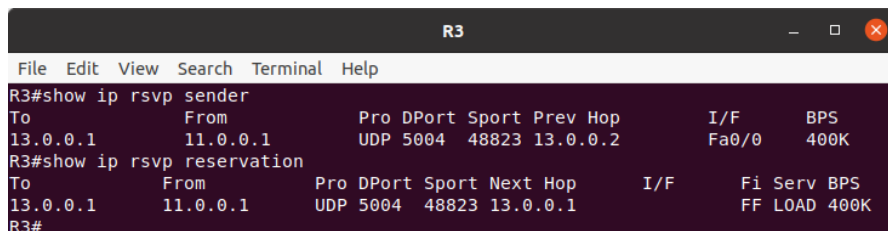
```
R1
File Edit View Search Terminal Help
R1#show ip rsvp sender
To          From          Pro DPort Sport Prev Hop      I/F      BPS
13.0.0.1    11.0.0.1    UDP 5004 48823 11.0.0.1    Fa0/0    400K
14.0.0.1    12.0.0.1    UDP 5004 40258 12.0.0.1    Fa0/1    200K
R1#show ip rsvp reservation
To          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  Se1/0    FF LOAD 400K
14.0.0.1    12.0.0.1    UDP 5004 40258 172.16.0.2  Se1/0    FF LOAD 200K
R1#
```

Figure 13: R1 router rsvp configuration



```
R2
File Edit View Search Terminal Help
R2#show ip rsvp sender
To          From          Pro DPort Sport Prev Hop      I/F      BPS
13.0.0.1    11.0.0.1    UDP 5004 48823 172.16.0.1  Se1/0    400K
14.0.0.1    12.0.0.1    UDP 5004 40258 172.16.0.1  Se1/0    200K
R2#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 13.0.0.1    Fa0/0    FF LOAD 400K
14.0.0.1    12.0.0.1    UDP 5004 40258 14.0.0.1    Fa0/1    FF LOAD 200K
R2#
```

Figure 14: R2 router rsvp configuration



```
R3
File Edit View Search Terminal Help
R3#show ip rsvp sender
To          From          Pro DPort Sport Prev Hop      I/F      BPS
13.0.0.1    11.0.0.1    UDP 5004 48823 13.0.0.2    Fa0/0    400K
R3#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 13.0.0.1    Fa0/0    FF LOAD 400K
R3#
```

Figure 15: R3 router rsvp configuration

```

R4
File Edit View Search Terminal Help
R4#show ip rsvp sender
To      From      Pro DPort Sport Prev Hop      I/F      BPS
14.0.0.1 12.0.0.1      UDP 5004 40258 14.0.0.2 Fa0/0     200K
R4#show ip rsvp reservation
To      From      Pro DPort Sport Next Hop      I/F      Fi Serv BPS
14.0.0.1 12.0.0.1      UDP 5004 40258 14.0.0.1 FF LOAD 200K
R4#

```

Figure 16: R4 router rsvp configuration

Traffic pattern 1

To be able to visualize the traffic and its operation, was used the statistics I/O graphic Wireshark functionality, so the next plots where made from the serial link capture when only the pattern 1 activated.

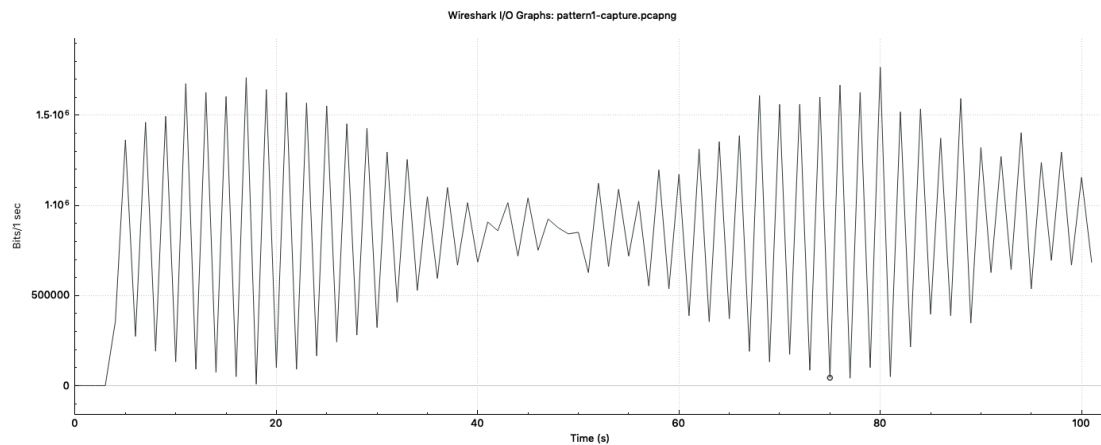


Figure 17: Traffic pattern 1 at 1s

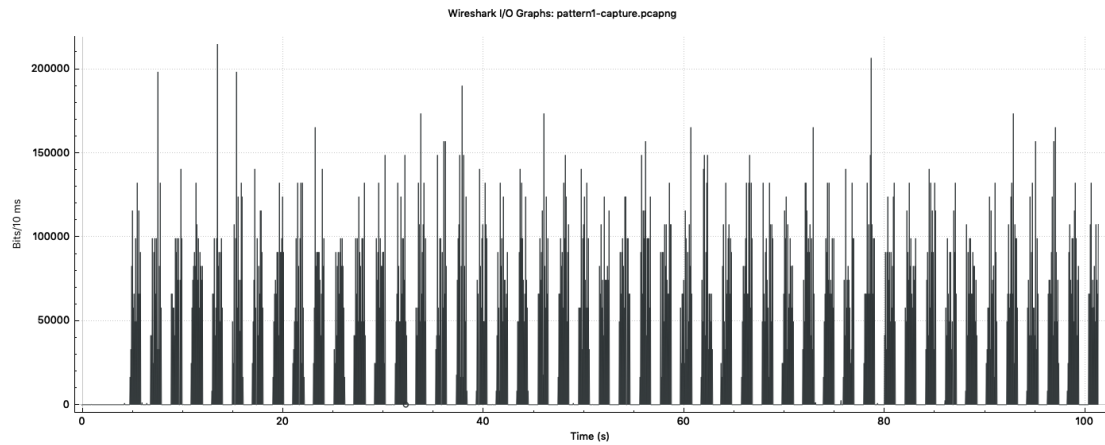


Figure 18: Traffic pattern 1 at 10ms

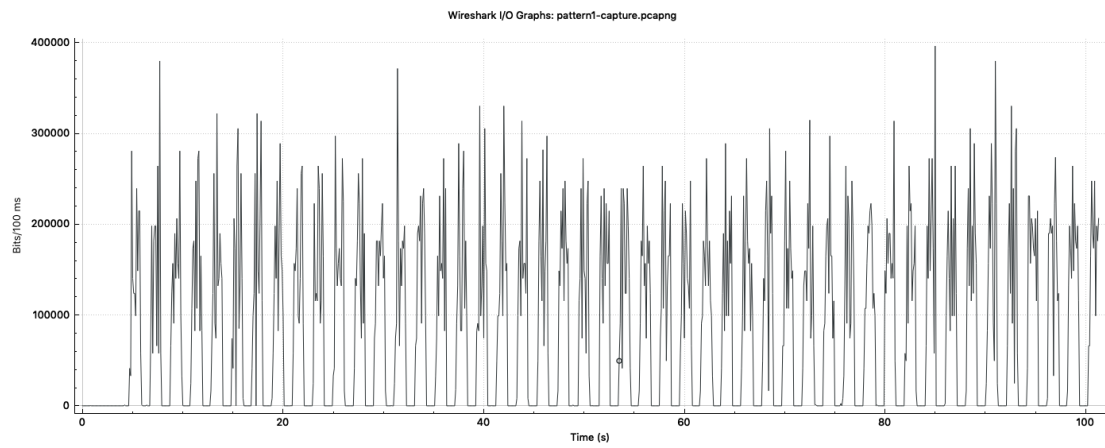


Figure 19: Traffic pattern 1 at 100ms

As can be seen from the previous plots, the traffic pattern flow performed works correctly as the average rate tends to be 1Mbps approximately as it corresponds to 1×10^6 bits showed perfectly on figure 17.

Moreover, better on figure 18, it can be seen how the on-off pulses performs well too because of the traffic stops every second during the traffic transmission.

3 Patterns active

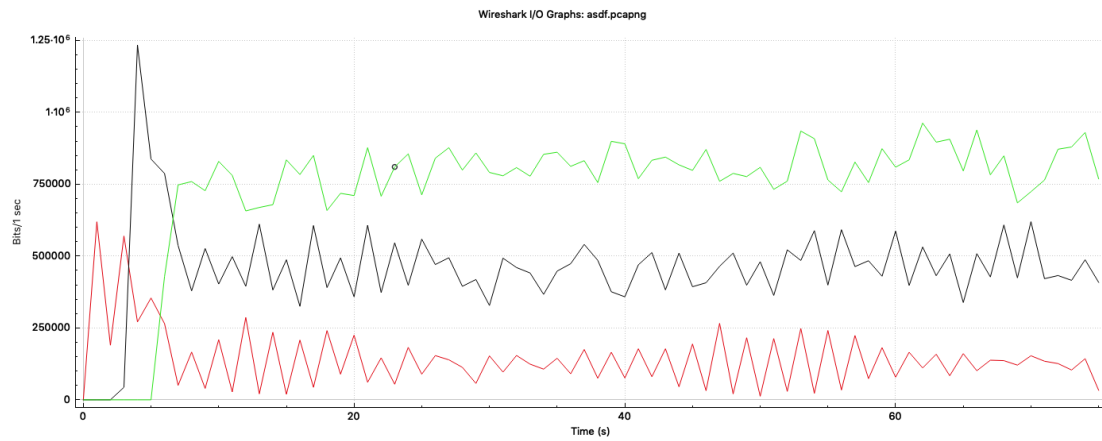


Figure 20: 3 Patterns active traffic at 1s

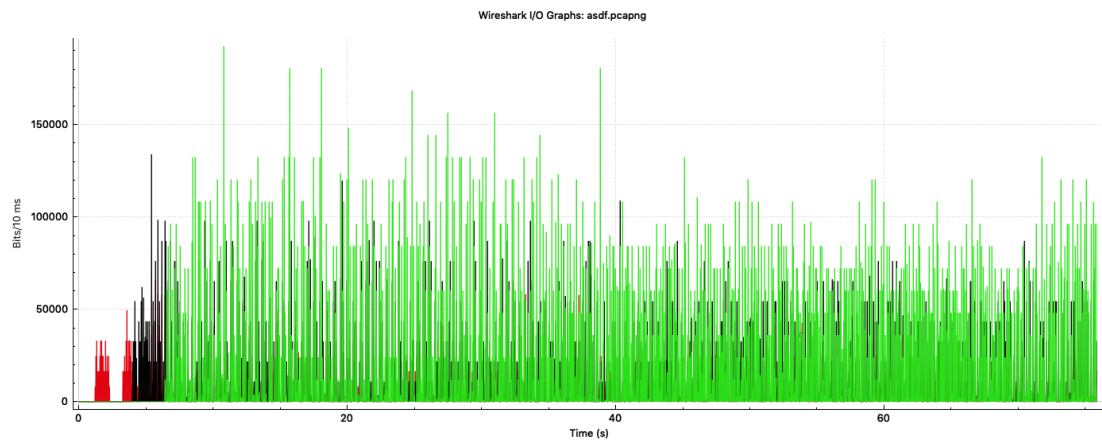


Figure 21: 3 Patterns active traffic at 10ms

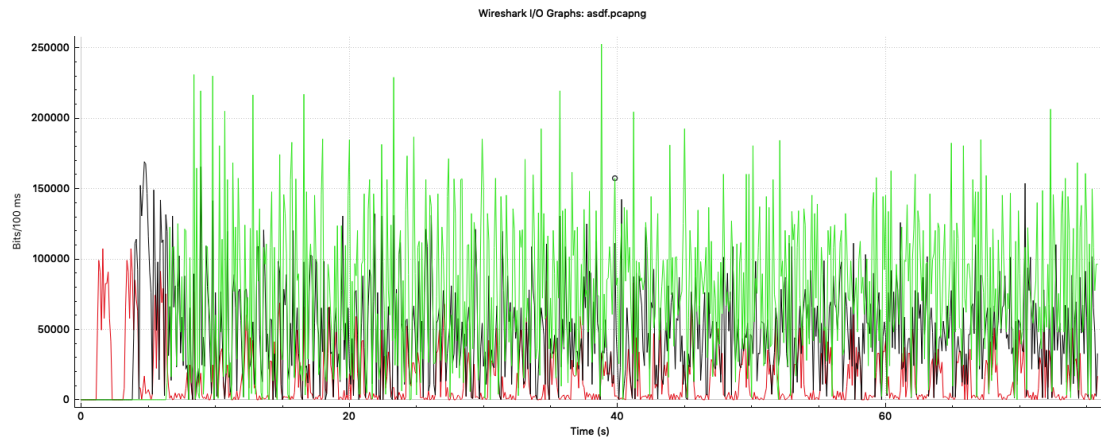


Figure 22: 3 Patterns active traffic at 100ms

After carried out the plots, we can see that at first sight thanks to the first of the plots, which is the one that best represents this exercise, it is showed the reserves work correctly.

There can be seen how the average minimum bit rate of the traffic is above its reserve and the remaining bandwidth is divided among the three traffics.

However, it can also be seen that, in some points, the performance is declining. One of the reasons could be due to the inestability of the virtual environment used.

References

- [1] [Quality of Service - Communicatino Services and Security - César Fernández Camón](#)
- [2] [GNS3 — The software that empowers network professionals](#)
- [3] [Wireshark · Go Deep](#)
- [4] [tshark - The Wireshark Network Analyzer 3.4.4](#)
- [5] [PackETH 2.1 Release Repository by jemcek](#)
- [6] [PackETH home web page](#)