University of Lleida

Master's Degree in Informatics Engineering

Higher Polythecnic School

Exercise 3

ICT Project: Communication Services and Security

Cèsar Fernández Camón

Albert Pérez Datsira

Jeongyun Lee

May 3, 2021

Table of contents

List of Tables

List of Figures

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 Simulation Environment

The GNS3 is a Graphical Network Simulator, open source and free software that offers an easy way to design, build and test networks of any size withouth the need for hardware by using a virtual environment.

Used by hundreds of thousands of network engineers worldwide to emulate, configure, test and troubleshoot virtual and real networks. Allows you to run a small topology consisting of only a few devices on your laptop, to those that have many devices hosted on multiple servers or even hosted in the cloud.

In addition, counts with an extended and active comunity where to ask questions, start discussions and see what is happenning in the world of GNS3, which helps a lot when any doubt raises.

Figure 1: GNS3 logo

3 Problem 1

Based on topology from slide page 21 (QoS, CAR), write a shell script that computes the average rate for the following traffic flows:

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

captured at interface R1-s1/0. Use *tshark* application.

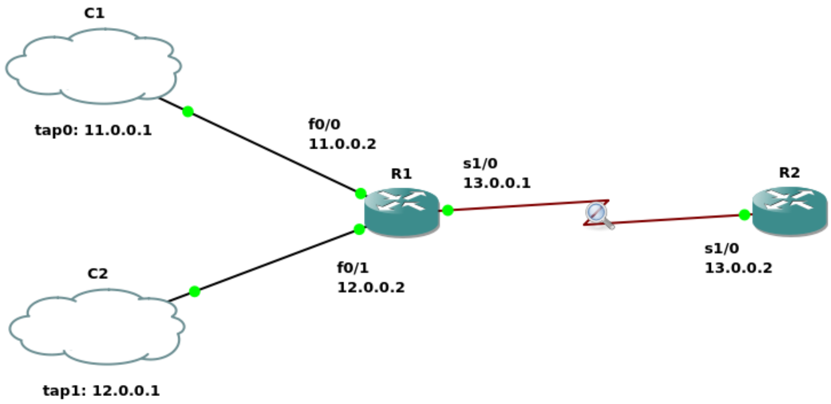


Figure 2: Problem 1 simulation topology scenario

3.1 Setup

This problem aims to show the behaviour of the Commited Acces Rate (CAR) feature. As we can see there are two *tap* interfaces that must ping in the same direction. The two traffics pass through an intermediate router (R1) that classifies the traffic (by origin). and sends it to the destination (R2) through a serial link.

By configuration the R1 router classifies the traffic from *tap0*. The conforming traffic is transmitted marked with precedence 7 and the exceeding one with precedence 1. The traffic from *tap1* is not classified (marked).

The necessary configurations to build the topology and the way to generate the traffic in order to capture the frames are specified below.

3.1.1 Host Computer

Tap interfaces configuration

First, we must focus on the two computer interfaces that will be connected/linked to our topology by representing them as clouds as shown in figure 2 on page 3.

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

Table 1: Tap interface configuration commands

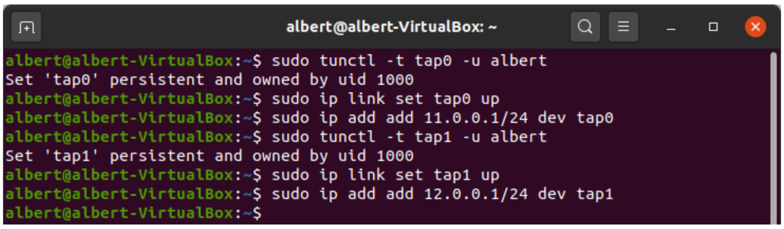


Figure 3: Set up tap interfaces

Once we have executed the previous commands we can go on to check the status of our *tap* interfaces using the *"ifconfig <interface>"* command, although we can also use other more modern commands such as *"ip address show"*.

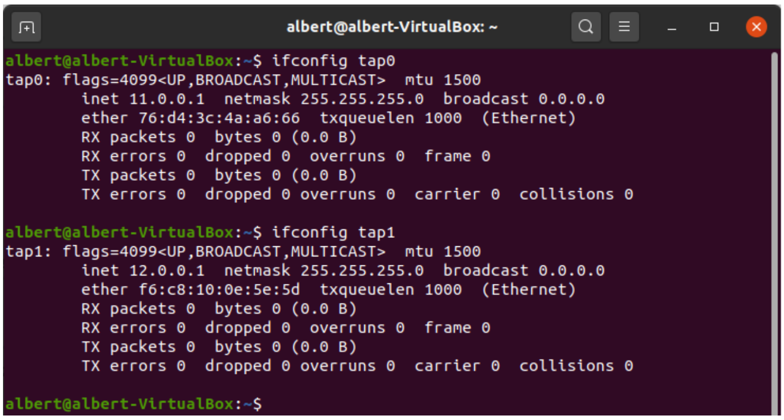


Figure 4: Check tap interfaces configuration

In addition, we can further check the established links and their status by using the command *"ip link show <interface>"*.

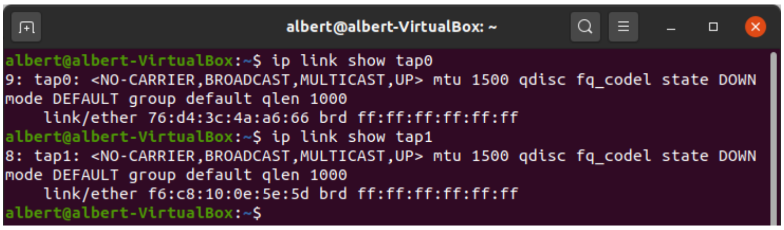


Figure 5: Check links tap interfaces

Network configuration

Then we should add the corresponding routes to our IP table so that our computer knows how to reach the network 13.0.0.0/24, and thus be able to perform the corresponding pings.

	Commands
Add IP Route Gateways	sudo route add -net 13.0.0.0/24 gw 11.0.0.2
	sudo ip route add 13.0.0.0/24 via 12.0.0.2 table rt2
	sudo rule add from 12.0.0.1/32 table rt2

Table 2: Network configuration commads

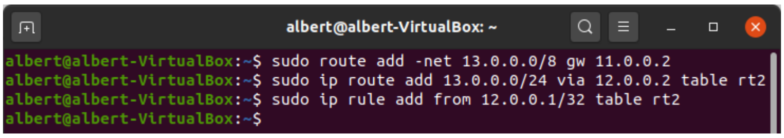


Figure 6: Set up specific network routing configuration

Hence, by means of the command *"ip route"* we will be able to observe our new routes in the computer IP table.

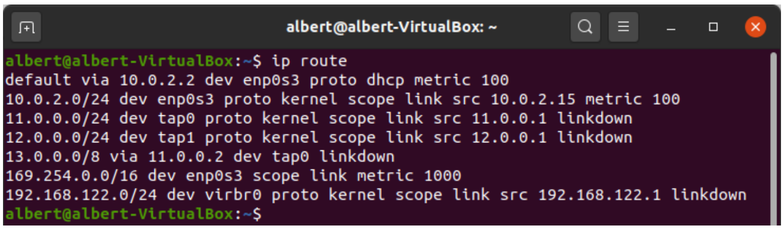


Figure 7: Host ip route table

But if we look at the commands used we will see that we would not only be using the local IP default table but we use the *rt2* one for *tap1* interface, so we can show ping responses on it.

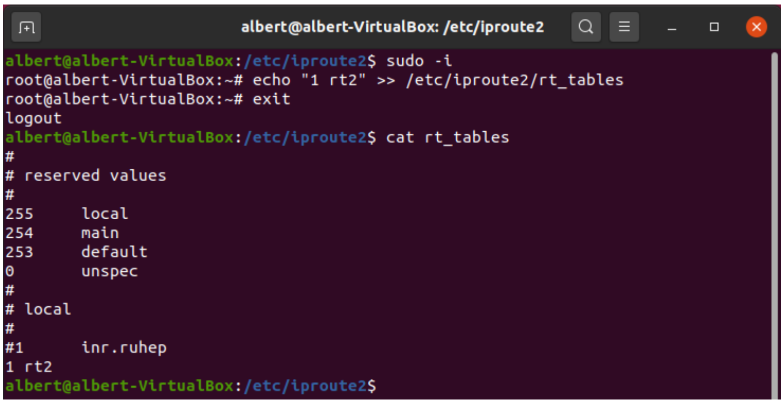


Figure 8: Configuration file /etc/iproute2/rt_tables

Although it is not necessary at all because just adding the gateway without specifying a new IP table as done with *tap0*, would be enough.

3.1.2 Routers

Regards routers, the general configurations are specified below, although if you wish to see them completely, they can be found in the corresponding *.cfg* files of the problem 1 GNS3 project.

<u>Router 1</u>

	Commands
Interface FastEthernet 0/0	ip address 11.0.0.2 255.255.255.0 duplex auto no shutdown
Interface FastEthernet 1/0	ip address 12.0.0.2 255.255.255.0 duplex auto no shutdown
Interface Serial 1/0	ip address 13.0.0.1 255.255.255.0 rate-limit output access-group 10 8000 2000 2000 set-prec-transmit 7 exceed-action set-prec-transmit 1 serial restart-delay 0 no shutdown
Access List	access-list 10 permit 11.0.0.1

Table 3: R1 router configuration commands

Router 2

	Commands
Interface Serial 1/0	ip address 13.0.0.2 255.255.255.0 serial restart-delay 0 no shutdown
Default Gateway	ip route 0.0.0.0 0.0.0.0 13.0.0.1

Table 4: R2 router configuration commands

Just comment the default gateway must be added to the R2 router so that it knows where to answer the pings.

3.1.3 Traffic generation

In order to generate traffic between interfaces the following commands were used:

- **$ ping -I tap0 13.0.0.2 -s 2000 -i 1**

  sending 2000 bytes every second at a rate of 16Kbps

- **$ ping -I tap1 13.0.0.2 -s 3000 -i 1**

  sending 3000 bytes every second at a rate of 24Kbps

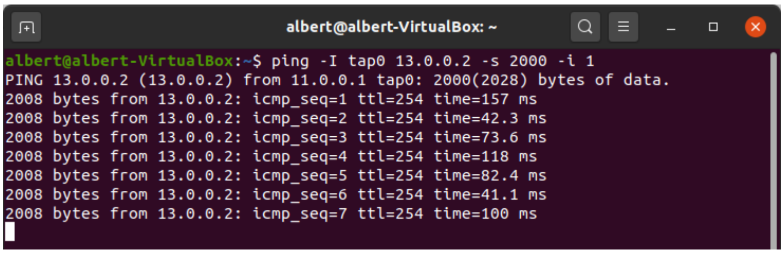


Figure 9: Ping through tap0 interface

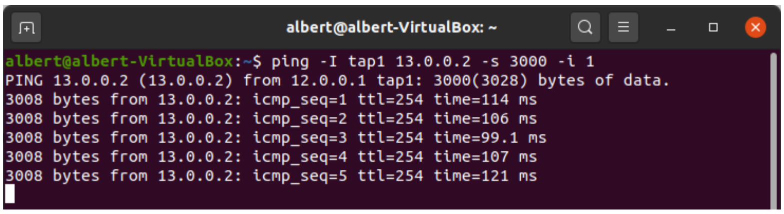
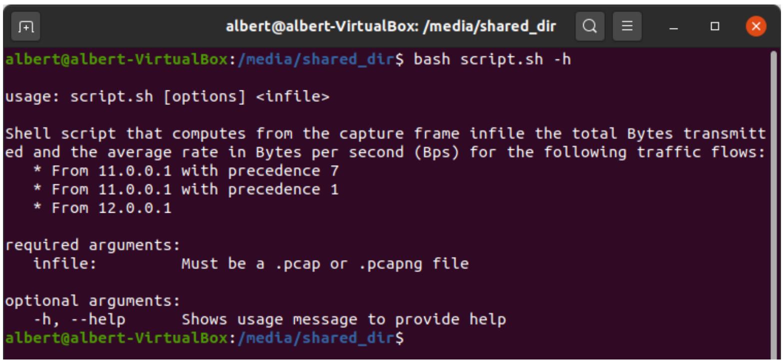


Figure 10: Ping through tap1 interface

3.2 Results

Once the topology was built and the traffic to the R1-s1/0 interface was captured using the wireshark tool, we were able to move on to analyzing the traffic to the R1-s1/0 interface.

A shell script was built that by using the *tshark* application calculates the average rate in Bytes per second (Bps) for the flows specified in the problem statement from reading the *.pcapng* capture file. But also show the bytes transmitted by each of the flows as well as the total bytes of the scenario.

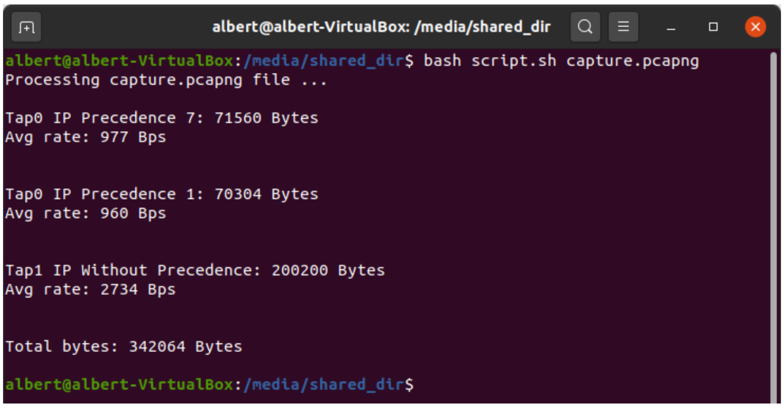


```
albert@albert-VirtualBox: /media/shared_dir
albert@albert-VirtualBox:/media/shared_dir$ bash script.sh -h

usage: script.sh [options] <infile>

Shell script that computes from the capture frame infile the total Bytes transmitt
ed and the average rate in Bytes per second (Bps) for the following traffic flows:
   * From 11.0.0.1 with precedence 7
   * From 11.0.0.1 with precedence 1
   * From 12.0.0.1

required arguments:
   infile:         Must be a .pcap or .pcapng file

optional arguments:
   -h, --help      Shows usage message to provide help
albert@albert-VirtualBox:/media/shared_dir$
```

Figure 11: Problem 1 shell script usage

The following table helps to better visualize the results:

Interface	IP precedence	Bytes transmitted (B)	Avg rate (Bps)	BW occupation (%)
tap0	7	71560	977	20.92
tap0	1	70304	960	20.55
tap1	Nan	200200	2734	58.52
	Total Bytes	342064		



```
albert@albert-VirtualBox: /media/shared_dir
albert@albert-VirtualBox:/media/shared_dir$ bash script.sh capture.pcapng
Processing capture.pcapng file ...

Tap0 IP Precedence 7: 71560 Bytes
Avg rate: 977 Bps


Tap0 IP Precedence 1: 70304 Bytes
Avg rate: 960 Bps


Tap1 IP Without Precedence: 200200 Bytes
Avg rate: 2734 Bps


Total bytes: 342064 Bytes

albert@albert-VirtualBox:/media/shared_dir$
```

Figure 12: Results problem 1 script execution

As we can see, approximately about half of the traffic in *tap0* was marked as surplus traffic. As we know from *tap1*, it is not under CAR's policy and it makes sense that it is sending traffic through R1 with a higher speed rate.

In addition, the same results can be obtained without the need to create a script just by using the powerful wireshark desktop tool.

Firstly must be disabled the "Decode IPv4 TOS field" preferences option to see the statistics and packets by filtering with the variable **ip.tos.precedence**.

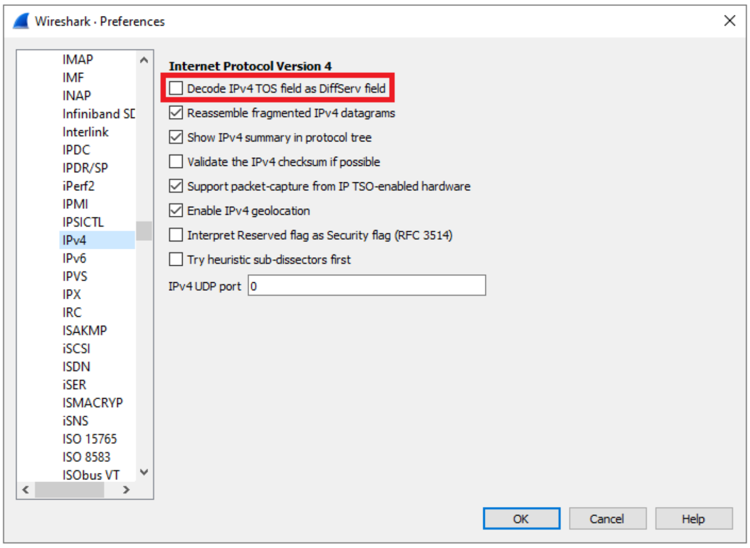


Figure 13: Wireshark preferences

And finally, in order to obtain the frame results, as in our case we are interested in knowing the destination ip, source ip and precedence, the following filters may be used:

1. **tap0 (prec 7):** ip.dst == 13.0.0.2 && ip.src == 11.0.0.1 && ip.tos.precedence == 7
2. **tap0 (prec 1):** ip.dst == 13.0.0.2 && ip.src == 11.0.0.1 && ip.tos.precedence == 1
3. **tap1:** ip.dst == 13.0.0.2 && ip.src == 12.0.0.1

4 Problem 2

Based on topology from slide page 39 (QoS, CBWFQ), write a shell script that computes the percentage of bandwidth occupation at serial link R1-R2 for each of the streams coming from C1-tap0 and C2-tap1. Use *tshark* application.

4.1 Setup

This problem aims to show the behaviour of the Class-Based Weighted Fair Queuing (CBWFQ).

The topology scenario is the same as in problem 1 which is represented in figure 2 on page 3. There are two *tap* interfaces that generate traffic in the same direction. Both traffic must pass through an intermediate router (R1) that classifies the traffic according to its origin and sends the traffic to the destination (R2) through a serial link.

As described on page 39 of the pdf, *tap0* traffic belongs to the first class, which is assigned a queue that uses 80% of the total bandwidth. On the other hand, the *tap1* traffic belongs to the second class and it uses the remaining 20% of the bandwidth.

When the queue is full, the "tail-drop" is applied to discard the incoming IP packages. To demonstrate this behavior, interfaces generate traffic at a higher speed than they are able to send from an intermediate router, due to the limitations of the serial link.

4.1.1 Host Computer

Tap interfaces configuration

The configuration of the *tap* interfaces (clouds) is the same as in problem 1. You can consult the commands in the table 1 on page 4, as well as the terminal execution example in the figure 3.

Network configuration

	Commands
Add Ip Route Gateways	sudo route add -net 13.0.0.0/24 gw 11.0.0.2 sudo route add -net 13.0.0.0/24 gw 12.0.0.2

Table 5: Network configuration commands

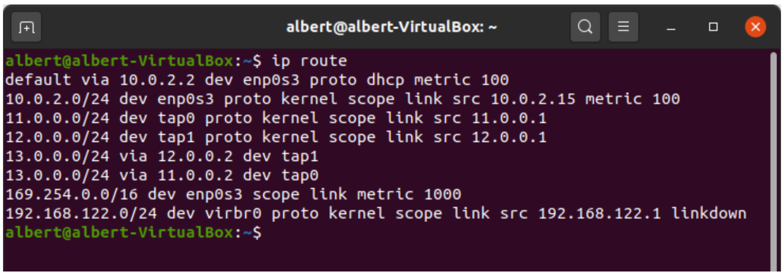


Figure 14: Host ip route table

4.1.2 Routers

Router 1

	Commands
Interface FastEthernet 0/0	ip address 11.0.0.2 255.255.255.0 duplex auto no shutdown
Interface FastEthernet 1/0	ip address 12.0.0.2 255.255.255.0 duplex auto no shutdown
Interface Serial 1/0	ip address 13.0.0.1 255.255.255.0 serial restart-delay 0 no shutdown
Acces Lists	access-list 101 permit 11.0.0.0 0.0.0.22 any access-list permit 102 12.0.0.0 0.0.0.255 any
Class Bandwidth	policy-map policy1 class class1 bandwidth percent 80 class class2 bandwidth percent 20
Class Access Group	class-map match-all class1 match access-group 101 class-map match-all class2 match access-group 102

Table 6: R1 router configuration commands

The creation of different policy classes (class1, class2) and the manual allocation of bandwidth can be seen in the applied configuration. As well as the mapping of these with the two access groups (101, 102).

Router 2

It has the same configuration as router 2 in problem 1. You can check its commands at page 4 in the table 1.

4.1.3 Traffic generation

Packet generator

To generate traffic through the network in order to fill bandwidth, we have used the CLI version of the *packETH* packet generator tool software.

Allows you to create and send any possible packet or sequence of packets on the ehternet link. It is very simple to install and use, as it is free and open source just getting from its repository on github, and supports many adjustments of parameters while sending.

- $ ./packETHcli -i tap0 -d 1000 -m 2 -f ping-tap0-100.pcap -n 0
- $ ./packETHcli -i tap1 -d 1000 -m 2 -f ping-tap1-300.pcap -n 0

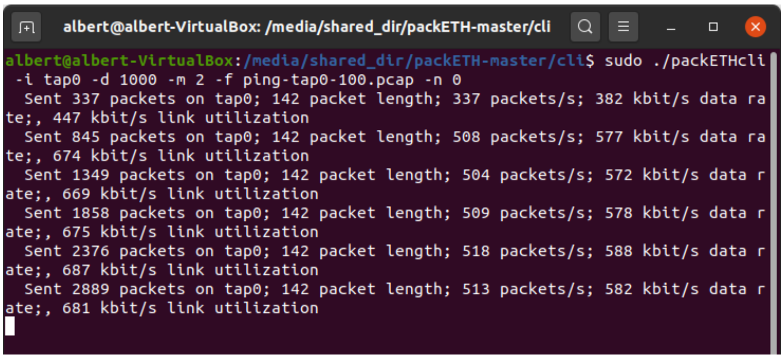


Figure 15: packETHcli capture tap0 execution

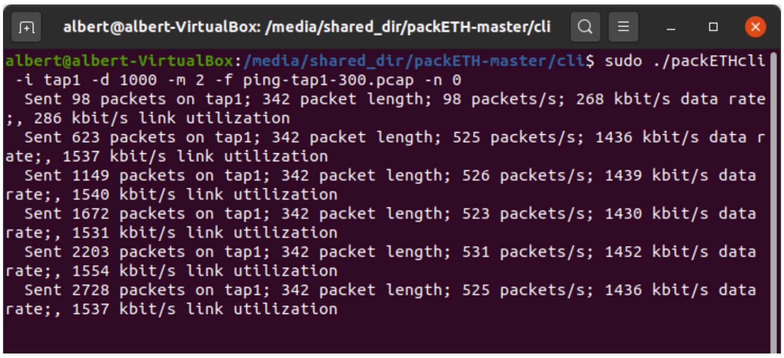


Figure 16: packETHcli capture tap1 execution

Set up pings

- $ ping -I tap0 13.0.0.2
- $ ping -I tap1 13.0.0.2

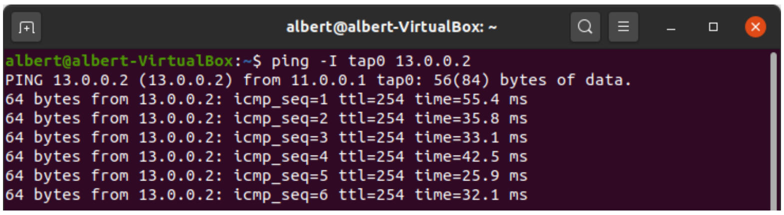


Figure 17: Ping through tap0 interface

Figure 18: Ping through tap1 interface

As we can see in figure 18, the ICMP[1] ping replies are not received, because of the 20% of the bandwidth allocated.

[1]ICMP: Internet Control Message Protocol

4.2 Results

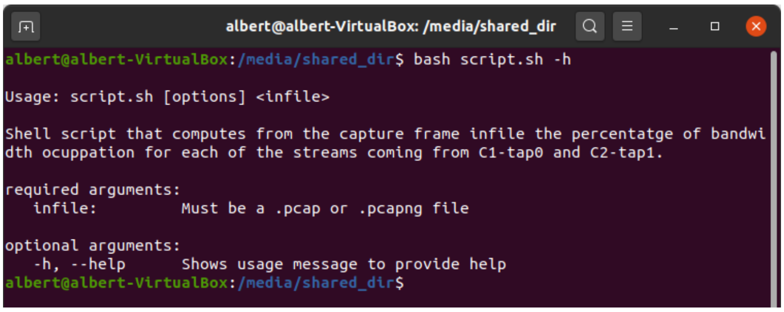


Figure 19: Problem 2 shell script usage

The following results have been obtained when the traffic has been captured for a enough period of time:

Interface	Bytes transmitted (B)	BW occupation (%)
tap0	460724	80%
tap1	112880	19%
Total Bytes	573604	

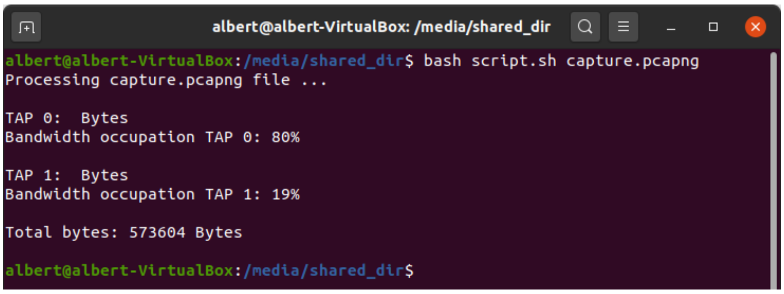


Figure 20: Results problem 2 script execution

As we can see the results are consistent since approximately 80% of the total traffic has its origin in *tap0* and the remaining 20% has its origin in *tap1*. The bandwidth obviously has the same proportion.

$$\textbf{Ratio} = \frac{Bytes(tap0)}{Bytes(tap1)} = \frac{460724}{112880} \approx 4.08 \approx 4 = \frac{80}{20} \tag{1}$$

5 Problem 3

Answer several questions, considering the following topology:

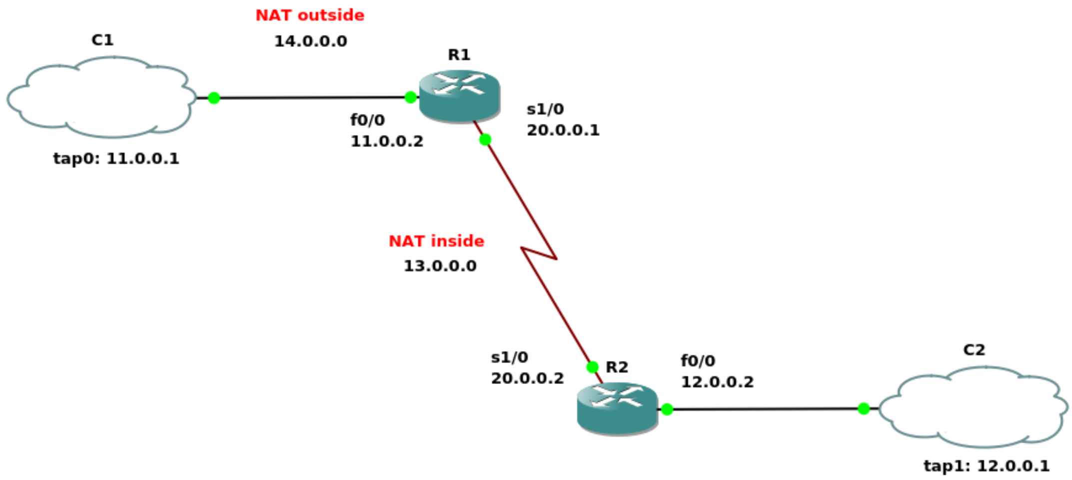


Figure 21: Problem 3 simulation topology scenario

5.1 Setup

This problem aims to demonstrate how Network Address Translation (NAT) works which is a way to map multiple local private addresses to a public one before transferring the information.

As can be seen, there are two *tap* interfaces, each one connected to a router which in turn are interconnected through a serial link.

But, in addition there is a NAT configuration in the R1 router which characterizise the behaviour of the whole topology and the internal 20.0.0.0 network.

5.1.1 Host computer

Tap interfaces configuration

Following the same configuration as in problem 1 and 2. You can consult the commands in the table 1 on page 4, as well as the terminal execution example in the figure 3.

Network configuration

	Commands
Add Ip Route Gateways	sudo route add -net 13.0.0.0/24 gw 11.0.0.2 dev tap0 sudo route add -net 14.0.0.0/24 gw 12.0.0.2 dev tap1

Table 7: Network routing configuration

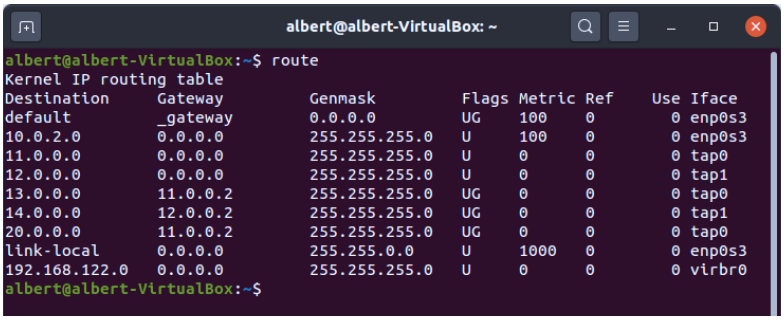


Figure 22: Host ip/kernel routing table

Might be also used the *"ip route"* command to check the ip table configuration as done in the previous problems. But in this case, for a better configuration view, was bet for using the first.

5.1.2 Default routers configuration

Router 1

	Commands
Interface FastEthernet 0/0	ip address 11.0.0.2 255.255.255.0 ip nat outside ip virtual-reassembly no shutdown duplex half
Interface Serial 1/0	ip address 20.0.0.1 255.255.255.0 ip nat inside ip virtual-reassembly serial restart-delay 0 no shutdown

Table 8: R1 router configuration commands

Router 2

	Commands
Interface Fast Ethernet 0/0	ip address 12.0.0.2 255.255.255.0 no shutdown duplex half
Interface Serial 1/0	ip address 20.0.0.2 255.255.255.0 serial restart-delay 0 no shutdown
Route configuration	ip route 11.0.0.0 255.255.255.0 20.0.0.1 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

Table 9: R2 router configuration commands

5.1.3 Traffic generation

Video transmission

- $ ffplay -protocol_whitelist file ,udp,rtp -i s.sdp 2> results.txt
- $ ffmpeg -re -i bbb_360p_c.ts -vcodec copy -an -sdp_file s.sdp -f rtp rtp://13.0.0.1:5004

Set up ping

- **$** sudo ping 13.0.0.1 -s 7025 -i 0.1

5.2 Questions

(Q1) 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

In first instance, what needs to be done is to define on our host computer IP table configuration the routes that represent the two NAT (inside/outside):

	Commands
R1 route configuration	ip route 12.0.0.0 255.255.255.0 20.0.0.2
	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

Table 10: R1 router ip route configuration

Then for R1 router NAT, must be defined the external network as the one connected to the *tap1* interface, and the external one as the one connected to the *tap0* interface.

	Commands
R1 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

Table 11: R1 router NAT configuration

```
!
ip route 12.0.0.0 255.255.255.0 20.0.0.2
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
!
no ip http server
no ip http secure-server
!
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
!
```

Figure 23: R1 router NAT and Route configuration

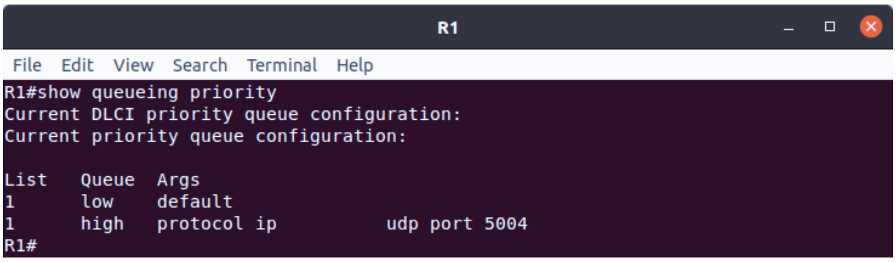
(Q2) Detail the required priority queuing (PQ) related R1 configuration in order to provide high priority to video streaming and low priority to default traffic. Which is the maximum allowed speed transmission rate along the path ?

The first thing to do is to define the priority group on the serial link interface of the R1 router:

- priority-group 1

To set the priority of the video (for the group), it has been established that IP traffic using User Datagram Protocol (UDP) through port 5004 has a high priority 2. Otherwise, the rest of the traffic has low priority.

- priority-list protocol ip high udp 5004
- priority-list 1 default low

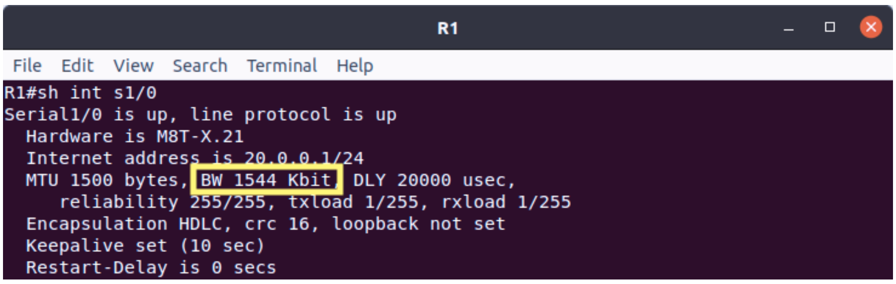


```
R1#show queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      high   protocol ip          udp port 5004
R1#
```

Figure 24: R1 router queueing priority configuration

With respect to the maximum transmission speed along the rotation is 1544 kbps. It has been obtained by executing the following command in R1 for the serial link interface:

- sh int s1/0



```
R1#sh int s1/0
Serial1/0 is up, line protocol is up
  Hardware is M8T-X.21
  Internet address is 20.0.0.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
```

Figure 25: R1 router serial link 1/0 bandwidth

(Q3) 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.

The video that has been downloaded from the indicated link is *Big Buck Bunny* with resolution of 640x360 (562 kpbs, Video bit rate).

In order to create a ping with a similar bit rate, the following command was used on the host computer terminal:

- $ sudo ping 13.0.0.1 -s 7025 -i 0.1

With this we assure sending 7025 bytes every 0.1 second, which is equivalent to sending 562kb every second.

(Q4) 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.

(Q5) Repeat the last item adding a ping transmission at the same rate that the video streaming.

References

[1] GNS3 — The software that empowers network professionals

[2] GNS3 — Community

[3] GNS3-0.5-tutorial.pdf

[4] Wireshark · Go Deep

[5] *tshark* - The Wireshark Network Analyzer 3.4.4

[6] Why is ip.tos.precedence empty?

[7] PackETH Repository by nickrobinson

[8] PackETH home web page