



Escuela  
Superior  
de Informática

UNIVERSIDAD DE CASTILLA-LA MANCHA  
ESCUELA SUPERIOR DE INFORMÁTICA

**Planificación e Integración de Sistemas y Servicios.**

4º GRADO EN INGENIERÍA INFORMÁTICA.

---

## **Práctica 4. Gestión de prioridad de tráfico en IP.**

---

*Autor:*

**Alberto Salas Seguín y  
Marcos López Sobrino.**

*Fecha:*

15 de enero de 2019

## Índice

<b>1. Entorno de trabajo.</b>	<b>2</b>
1.1. Cliente y servidor. . . . .	2
1.2. Router. . . . .	3
1.3. Archivos de configuración e inicialización. . . . .	3
<b>2. Marcado de paquetes.</b>	<b>4</b>
2.1. Configuración de iptables. . . . .	4
2.2. Reglas iptables utilizadas. . . . .	5
<b>3. Análisis del rendimiento.</b>	<b>7</b>
3.1. Captura de paquetes mediante Wireshark. . . . .	9
<b>4. Incidencias surgidas en el transcurso de la práctica.</b>	<b>12</b>
<b>5. Repositorio.</b>	<b>12</b>
<b>6. Bibliografía</b>	<b>13</b>

## 1. Entorno de trabajo.

El entorno de trabajo utilizado han sido 3 máquinas virtuales gestionadas mediante Vagrant, con la imagen *ubuntu/trusty64*. Una de ellas actúa como cliente, otra como servidor y otra como router entre ambas. A continuación, explicamos la configuración de las redes, de las máquinas, y los paquetes instalados.

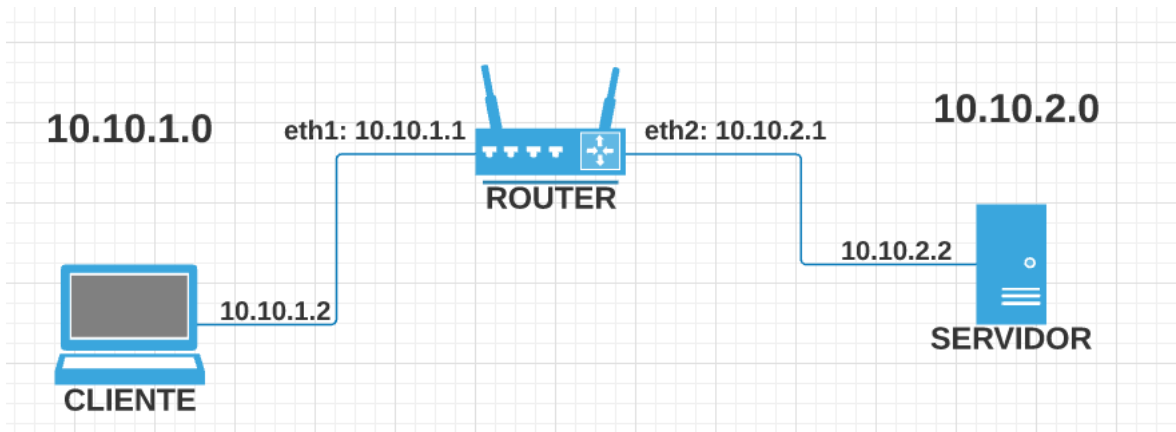


Figura 1: Topología de la red.

### 1.1. Cliente y servidor.

Como vemos en la figura anterior, la red del cliente, que corresponde a la interfaz **eth1** del router, tiene como IP **10.10.1.0**; mientras que la red del servidor, que corresponde a la interfaz **eth2** del router, tiene como IP **10.10.2.0**.

Para el correcto funcionamiento de las comunicaciones, hay que indicarle a la máquina cuál es la ruta por defecto, para ello, se ejecutan los siguientes comandos:

```
$ sudo ip route del default dev eth0
$ sudo ip route add default via <ip_router> dev eth1
```

donde *ip\_router* sería la dirección IP de la interfaz del router dentro de la red correspondiente.

Los paquetes necesarios para el funcionamiento de la práctica en el caso del cliente y el servidor son:

- **iptables-persistent**
- **iperf**

## 1.2. Router.

A la máquina del router se le ha instalado un entorno gráfico para poder usar cómodamente la herramienta **Wireshark**. Además, se le han asignado 2048 MB de RAM. Por otra parte, se ha configurado las interfaces de red con la dirección **.1** de la red correspondiente, tal y como vemos en la figura 1. Adicionalmente, para que la máquina actúe como router se debe activar el **IP forwarding**, para ello, en el playbook correspondiente del router se ha añadido una regla específica.

Por último, en el router se han instalado los siguientes paquetes:

- Para la interfaz gráfica:
  - **xorg**
  - **gnome-core**
  - **gnome-system-tools**
  - **gnome-app-install**
- **wireshark**
- **iptables-persistent**

## 1.3. Archivos de configuración e inicialización.

Todo lo anterior se recoge en el archivo **Vagrantfile** y los distintos **playbooks**, que se pueden encontrar junto a este documento. Para poner en funcionamiento el entorno de trabajo, ejecutamos:

```
vagrant up --provision
```

## 2. Marcado de paquetes.

### 2.1. Configuración de iptables.

Dado que tanto en la máquina del router como la del cliente, se van a introducir reglas de iptables para el correcto funcionamiento, para no perder estas reglas se va a utilizar *iptables-persistent*, el modo para recuperar todas las reglas cada vez que se inicie la máquina es mediante una *task* cuya función es cargar un archivo con todas las reglas necesarias en el directorio */etc/iptables/rules.v4*, este directorio se encuentra tanto en la máquina del router como en la máquina del cliente. La regla que hemos utilizado para copiar este archivo en el directorio mencionado es la siguiente:

```
- name: copy iptables
  copy:
    src: <archivo a copiar>
    dest: /etc/iptables/rules.v4
    owner: root
    group: root
    mode: 0644
  notify: "restore iptables"
```

Como vemos en el código anterior, en la última línea se hace mención a *notify*, lo cual hace necesario la presencia de un manejador, de manera que cada vez que se invoque a la tarea *copy tables*, se va a hacer el *notify*, cuya función es hacer un **restore** del archivo */etc/iptables/rules.v4* que hemos copiado. Esto lo podemos ver en el siguiente código:

```
handlers:
  - name: restore iptables
    shell: iptables-restore /etc/iptables/rules.v4
    listen: "restore iptables"
```

## 2.2. Reglas iptables utilizadas.

### ■ Router.

En el caso del router hemos utilizado las reglas iptables que vienen dadas en el enunciado de la práctica. Estas son:

```
iptables -t filter -A INPUT -m dscp --dscp 14 -m limit --  
    limit 5/s --limit-burst 5 -j ACCEPT  
  
iptables -t filter -A INPUT -m dscp --dscp 14 -j DROP
```

Donde los parámetros `-limit` y `-limit-burst` significan lo siguiente.

- **limit.** El módulo `limit` se utiliza para restringir la tasa de coincidencias, por ejemplo, para suprimir los mensajes de registro. Sólo coincidirá con un número determinado de veces por segundo (por defecto 3 coincidencias por hora, con una ráfaga de 5). Se necesitan dos argumentos opcionales:
  - **-limit.** Seguido de un número y una unidad, especifica el número medio máximo de coincidencias que puede permitirse por segundo. Nos permite configurar el tiempo que tiene que pasar para que esta regla se cumpla entre dos paquetes.
  - **-limit-burst.** Seguido de un número que indica la cantidad máxima de paquetes antes de que se active el límite anterior.
- **Cliente y servidor.** Dado que vamos a marcar paquetes, la tabla que vamos a usar es **mangle**. Por un lado, el tráfico RTP lo vamos a marcar con **Expedited Forwarding**, esta clase es la máxima prioridad para el marcado de paquetes RTP, debido a que no queremos una pérdida de paquetes, ya que estos paquetes contienen la voz de una comunicación, por lo que en caso de pérdida, la conversación no se desarrollaría de forma completa. Por otro lado, el tráfico SIP lo vamos a marcar como **Assured Forwarding 11**, este tráfico se utiliza para la señalización durante la llamada y establecer las comunicaciones, por lo que no es necesario marcarlo con una máxima prioridad como es el caso del tráfico RTP. Por último, el resto de paquetes, los vamos a marcar como **Best Effort**, es la categoría con menor prioridad en el marcado de paquetes.

Las reglas nos quedan de la siguiente manera:

- Resto de paquetes.

```
iptables -t mangle -A OUTPUT -j DSCP --set-dscp 0
```

- Tráfico SIP.

```
iptables -t mangle -A OUTPUT -p udp -m udp --dport 5060 -j  
DSCP --set-dscp-class af11  
  
iptables -t mangle -A OUTPUT -p tcp -m tcp --dport 5060 -j  
DSCP --set-dscp-class af11
```

Estas reglas, permiten el tráfico udp y tcp saliente que tiene como destino el puerto 5060 (dport), que es el puerto dedicado al protocolo SIP. Como hemos dicho antes, este tráfico se ha marcado con AF11.

- Tráfico RTP.

```
iptables -t mangle -A OUTPUT -p udp -m udp --dport  
6970:6999 -j DSCP --set-dscp-class ef  
  
iptables -t mangle -A OUTPUT -p tcp -m tcp --dport  
6970:6999 -j DSCP --set-dscp-class ef
```

Estas reglas, permiten el tráfico udp y tcp saliente que tiene como destino el rango de puertos recomendados por la IETF que son del puerto 6970 al 6999.

### 3. Análisis del rendimiento.

Para la realización de este apartado se va a utilizar la herramienta **iperf**. Lo primero que vamos a hacer es utilizar la herramienta para ver cual es el ancho de banda sin incluir el marcado de paquetes. Para establecer la comunicación:

```
$ iperf -s // Máquina del server.
$ iperf -c 10.10.2.2 // Máquina del router.
```

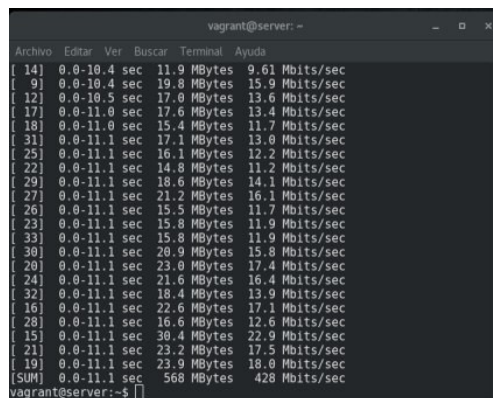
Tras realizar esta prueba, el ancho de banda obtenido:

```
vagrant@client:~$ iperf -c 10.10.2.2
-----
Client connecting to 10.10.2.2, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 10.10.1.2 port 39379 connected with 10.10.2.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  1.75 GBytes 1.50 Gbits/sec
```

Figura 2: Ancho de banda sin marcado.

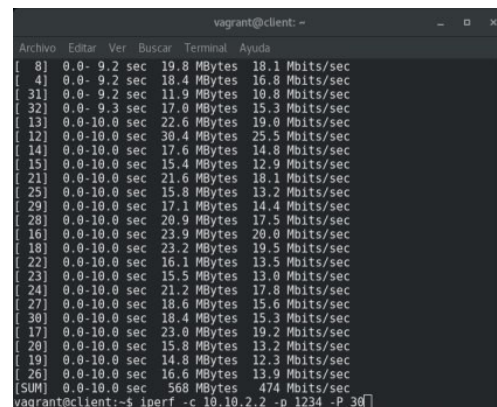
A continuación, se muestran los anchos de banda usando el marcado de paquetes. Estos anchos de banda se han medido ejecutando *iperf* de manera concurrente en tres terminales de la máquina *cliente* y en tres terminales de la máquina *servidor*. Hay que ejecutar iperf en tres puertos concretos:

- Puerto **1234** para el resto de tráfico.



```
vagrant@server:~$ iperf
[ 14] 0.0-10.4 sec 11.9 MBytes 9.61 Mbits/sec
[ 9] 0.0-10.4 sec 19.8 MBytes 15.9 Mbits/sec
[ 12] 0.0-10.5 sec 17.0 MBytes 13.6 Mbits/sec
[ 17] 0.0-11.0 sec 17.6 MBytes 13.4 Mbits/sec
[ 18] 0.0-11.0 sec 15.4 MBytes 11.7 Mbits/sec
[ 31] 0.0-11.1 sec 17.1 MBytes 13.0 Mbits/sec
[ 25] 0.0-11.1 sec 16.1 MBytes 12.2 Mbits/sec
[ 22] 0.0-11.1 sec 14.8 MBytes 11.2 Mbits/sec
[ 29] 0.0-11.1 sec 18.6 MBytes 14.1 Mbits/sec
[ 27] 0.0-11.1 sec 21.2 MBytes 16.1 Mbits/sec
[ 26] 0.0-11.1 sec 15.5 MBytes 11.7 Mbits/sec
[ 23] 0.0-11.1 sec 15.8 MBytes 11.9 Mbits/sec
[ 33] 0.0-11.1 sec 15.8 MBytes 11.9 Mbits/sec
[ 30] 0.0-11.1 sec 20.9 MBytes 15.8 Mbits/sec
[ 20] 0.0-11.1 sec 23.0 MBytes 17.4 Mbits/sec
[ 24] 0.0-11.1 sec 21.6 MBytes 16.4 Mbits/sec
[ 32] 0.0-11.1 sec 18.4 MBytes 13.9 Mbits/sec
[ 16] 0.0-11.1 sec 22.6 MBytes 17.1 Mbits/sec
[ 28] 0.0-11.1 sec 16.6 MBytes 12.6 Mbits/sec
[ 15] 0.0-11.1 sec 30.4 MBytes 22.9 Mbits/sec
[ 21] 0.0-11.1 sec 23.2 MBytes 17.5 Mbits/sec
[ 19] 0.0-11.1 sec 23.9 MBytes 18.0 Mbits/sec
[SUM] 0.0-11.1 sec 568 MBytes 428 Mbits/sec
vagrant@server:~$
```

Fig. 3: Ejecución de iperf en el servidor.

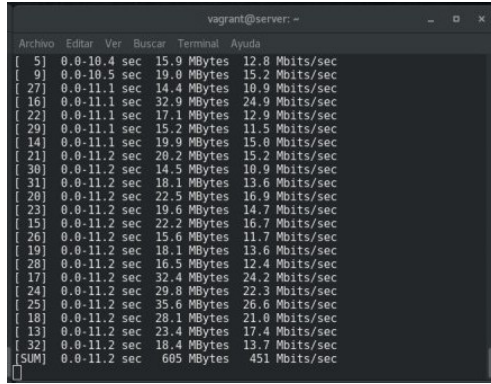


```
vagrant@client:~$ iperf
[ 8] 0.0- 9.2 sec 19.8 MBytes 18.1 Mbits/sec
[ 4] 0.0- 9.2 sec 18.4 MBytes 16.8 Mbits/sec
[ 31] 0.0- 9.2 sec 11.9 MBytes 10.8 Mbits/sec
[ 32] 0.0- 9.3 sec 17.0 MBytes 13.6 Mbits/sec
[ 13] 0.0-10.0 sec 22.6 MBytes 19.0 Mbits/sec
[ 12] 0.0-10.0 sec 30.4 MBytes 25.5 Mbits/sec
[ 14] 0.0-10.0 sec 17.6 MBytes 14.8 Mbits/sec
[ 15] 0.0-10.0 sec 15.4 MBytes 12.9 Mbits/sec
[ 21] 0.0-10.0 sec 21.6 MBytes 18.1 Mbits/sec
[ 25] 0.0-10.0 sec 15.8 MBytes 13.2 Mbits/sec
[ 29] 0.0-10.0 sec 17.1 MBytes 14.4 Mbits/sec
[ 28] 0.0-10.0 sec 20.9 MBytes 17.5 Mbits/sec
[ 16] 0.0-10.0 sec 23.9 MBytes 20.0 Mbits/sec
[ 18] 0.0-10.0 sec 23.2 MBytes 19.5 Mbits/sec
[ 22] 0.0-10.0 sec 16.1 MBytes 13.5 Mbits/sec
[ 23] 0.0-10.0 sec 15.5 MBytes 13.0 Mbits/sec
[ 24] 0.0-10.0 sec 21.2 MBytes 17.8 Mbits/sec
[ 27] 0.0-10.0 sec 18.6 MBytes 15.6 Mbits/sec
[ 30] 0.0-10.0 sec 19.4 MBytes 15.3 Mbits/sec
[ 17] 0.0-10.0 sec 23.0 MBytes 19.2 Mbits/sec
[ 20] 0.0-10.0 sec 15.8 MBytes 13.2 Mbits/sec
[ 19] 0.0-10.0 sec 14.8 MBytes 12.3 Mbits/sec
[ 26] 0.0-10.0 sec 16.6 MBytes 13.9 Mbits/sec
[SUM] 0.0-10.0 sec 568 MBytes 474 Mbits/sec
vagrant@client:~$ iperf -c 10.10.2.2 -p 1234 -P 30
```

Fig. 4: Ejecución de iperf en el cliente.



- Puerto **5060** para el tráfico *SIP*.

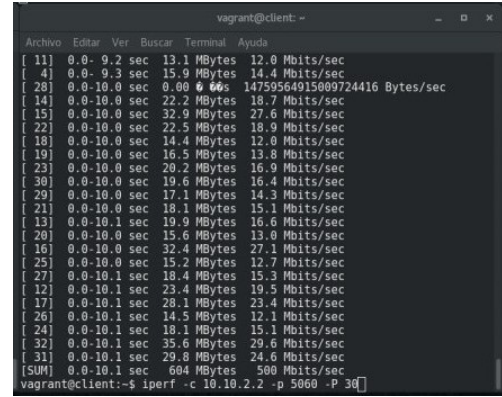


```

vagrant@server:~$ iperf -s
[  5] 0.0-10.4 sec 15.9 MBytes 12.8 Mbits/sec
[  9] 0.0-10.5 sec 19.0 MBytes 15.2 Mbits/sec
[ 27] 0.0-11.1 sec 14.4 MBytes 10.9 Mbits/sec
[ 16] 0.0-11.1 sec 32.9 MBytes 24.9 Mbits/sec
[ 22] 0.0-11.1 sec 17.1 MBytes 12.9 Mbits/sec
[ 29] 0.0-11.1 sec 15.2 MBytes 11.5 Mbits/sec
[ 14] 0.0-11.1 sec 19.9 MBytes 15.0 Mbits/sec
[ 21] 0.0-11.2 sec 20.2 MBytes 15.2 Mbits/sec
[ 30] 0.0-11.2 sec 14.5 MBytes 10.9 Mbits/sec
[ 31] 0.0-11.2 sec 18.1 MBytes 13.6 Mbits/sec
[ 20] 0.0-11.2 sec 22.5 MBytes 16.9 Mbits/sec
[ 23] 0.0-11.2 sec 19.6 MBytes 14.7 Mbits/sec
[ 15] 0.0-11.2 sec 22.2 MBytes 16.7 Mbits/sec
[ 26] 0.0-11.2 sec 15.6 MBytes 11.7 Mbits/sec
[ 19] 0.0-11.2 sec 18.1 MBytes 13.6 Mbits/sec
[ 28] 0.0-11.2 sec 16.5 MBytes 12.4 Mbits/sec
[ 17] 0.0-11.2 sec 32.4 MBytes 24.2 Mbits/sec
[ 24] 0.0-11.2 sec 29.8 MBytes 22.3 Mbits/sec
[ 25] 0.0-11.2 sec 35.6 MBytes 26.6 Mbits/sec
[ 18] 0.0-11.2 sec 28.1 MBytes 21.0 Mbits/sec
[ 13] 0.0-11.2 sec 23.4 MBytes 17.4 Mbits/sec
[ 32] 0.0-11.2 sec 18.4 MBytes 13.7 Mbits/sec
[SUM] 0.0-11.2 sec 685 MBytes 451 Mbits/sec

```

Fig. 5: Ejecución de iperf en el servidor.



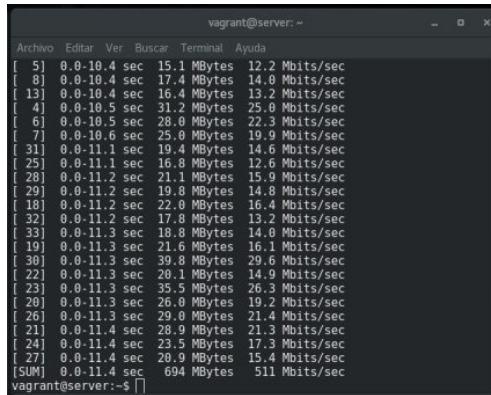
```

vagrant@client:~$ iperf -c 10.10.2.2 -p 5060 -P 30
[ 11] 0.0- 9.2 sec 13.1 MBytes 12.0 Mbits/sec
[  4] 0.0- 9.3 sec 15.9 MBytes 14.4 Mbits/sec
[ 28] 0.0-10.0 sec 0.00 0.00s 14759564915009724416 Bytes/sec
[ 14] 0.0-10.0 sec 22.2 MBytes 18.7 Mbits/sec
[ 15] 0.0-10.0 sec 32.9 MBytes 27.6 Mbits/sec
[ 22] 0.0-10.0 sec 22.5 MBytes 18.9 Mbits/sec
[ 18] 0.0-10.0 sec 14.4 MBytes 12.0 Mbits/sec
[ 19] 0.0-10.0 sec 16.5 MBytes 13.8 Mbits/sec
[ 23] 0.0-10.0 sec 20.2 MBytes 16.9 Mbits/sec
[ 30] 0.0-10.0 sec 19.6 MBytes 16.4 Mbits/sec
[ 29] 0.0-10.0 sec 17.1 MBytes 14.3 Mbits/sec
[ 21] 0.0-10.0 sec 18.1 MBytes 15.1 Mbits/sec
[ 13] 0.0-10.1 sec 19.9 MBytes 16.6 Mbits/sec
[ 20] 0.0-10.0 sec 15.6 MBytes 13.0 Mbits/sec
[ 16] 0.0-10.0 sec 32.4 MBytes 27.1 Mbits/sec
[ 25] 0.0-10.0 sec 15.2 MBytes 12.7 Mbits/sec
[ 27] 0.0-10.1 sec 18.4 MBytes 15.3 Mbits/sec
[ 12] 0.0-10.1 sec 23.4 MBytes 19.5 Mbits/sec
[ 17] 0.0-10.1 sec 28.1 MBytes 23.4 Mbits/sec
[ 26] 0.0-10.1 sec 14.5 MBytes 12.1 Mbits/sec
[ 24] 0.0-10.1 sec 18.1 MBytes 15.1 Mbits/sec
[ 32] 0.0-10.1 sec 35.6 MBytes 29.6 Mbits/sec
[ 31] 0.0-10.1 sec 29.8 MBytes 24.6 Mbits/sec
[SUM] 0.0-10.1 sec 604 MBytes 500 Mbits/sec
vagrant@client:~$ iperf -c 10.10.2.2 -p 5060 -P 30

```

Fig. 6: Ejecución de iperf en el cliente.

- Puerto **6980** para el tráfico *RTP*.

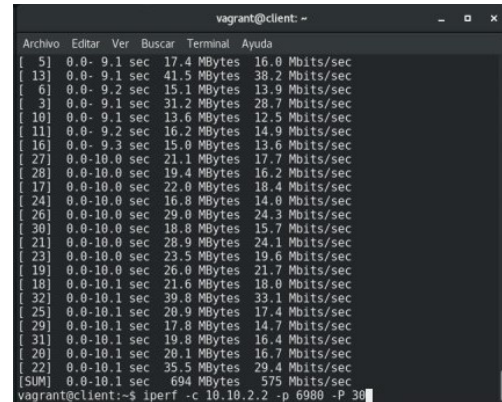


```

vagrant@server:~$ iperf -s
[  5] 0.0-10.4 sec 15.1 MBytes 12.2 Mbits/sec
[  8] 0.0-10.4 sec 17.4 MBytes 14.0 Mbits/sec
[ 13] 0.0-10.4 sec 16.4 MBytes 13.2 Mbits/sec
[  4] 0.0-10.5 sec 31.2 MBytes 25.0 Mbits/sec
[  6] 0.0-10.5 sec 28.0 MBytes 22.3 Mbits/sec
[  7] 0.0-10.6 sec 25.0 MBytes 19.9 Mbits/sec
[ 31] 0.0-11.1 sec 19.4 MBytes 14.6 Mbits/sec
[ 25] 0.0-11.1 sec 16.8 MBytes 12.6 Mbits/sec
[ 28] 0.0-11.2 sec 21.1 MBytes 15.9 Mbits/sec
[ 29] 0.0-11.2 sec 19.8 MBytes 14.8 Mbits/sec
[ 18] 0.0-11.2 sec 22.0 MBytes 16.4 Mbits/sec
[ 32] 0.0-11.2 sec 17.8 MBytes 13.2 Mbits/sec
[ 33] 0.0-11.3 sec 18.8 MBytes 14.0 Mbits/sec
[ 19] 0.0-11.3 sec 21.6 MBytes 16.1 Mbits/sec
[ 30] 0.0-11.3 sec 39.8 MBytes 29.6 Mbits/sec
[ 22] 0.0-11.3 sec 20.1 MBytes 14.9 Mbits/sec
[ 23] 0.0-11.3 sec 35.5 MBytes 26.3 Mbits/sec
[ 20] 0.0-11.3 sec 26.0 MBytes 19.2 Mbits/sec
[ 26] 0.0-11.3 sec 29.0 MBytes 21.4 Mbits/sec
[ 21] 0.0-11.4 sec 28.9 MBytes 21.3 Mbits/sec
[ 24] 0.0-11.4 sec 23.5 MBytes 17.3 Mbits/sec
[ 27] 0.0-11.4 sec 20.9 MBytes 15.4 Mbits/sec
[SUM] 0.0-11.4 sec 694 MBytes 511 Mbits/sec
vagrant@server:~$

```

Fig. 7: Ejecución de iperf en el servidor.



```

vagrant@client:~$ iperf -c 10.10.2.2 -p 6980 -P 30
[  5] 0.0- 9.1 sec 17.4 MBytes 16.0 Mbits/sec
[ 13] 0.0- 9.1 sec 41.5 MBytes 38.2 Mbits/sec
[  6] 0.0- 9.2 sec 15.1 MBytes 13.9 Mbits/sec
[  3] 0.0- 9.1 sec 31.2 MBytes 28.7 Mbits/sec
[ 10] 0.0- 9.1 sec 13.6 MBytes 12.5 Mbits/sec
[ 11] 0.0- 9.2 sec 16.2 MBytes 14.9 Mbits/sec
[ 16] 0.0- 9.3 sec 15.0 MBytes 13.6 Mbits/sec
[ 27] 0.0-10.0 sec 21.1 MBytes 17.7 Mbits/sec
[ 28] 0.0-10.0 sec 19.4 MBytes 16.2 Mbits/sec
[ 17] 0.0-10.0 sec 22.0 MBytes 18.4 Mbits/sec
[ 24] 0.0-10.0 sec 16.8 MBytes 14.0 Mbits/sec
[ 26] 0.0-10.0 sec 29.0 MBytes 24.3 Mbits/sec
[ 30] 0.0-10.0 sec 18.8 MBytes 15.7 Mbits/sec
[ 21] 0.0-10.0 sec 28.9 MBytes 24.1 Mbits/sec
[ 22] 0.0-10.0 sec 23.5 MBytes 19.6 Mbits/sec
[ 19] 0.0-10.0 sec 26.0 MBytes 21.7 Mbits/sec
[ 18] 0.0-10.1 sec 21.6 MBytes 18.0 Mbits/sec
[ 32] 0.0-10.1 sec 39.8 MBytes 33.1 Mbits/sec
[ 25] 0.0-10.1 sec 28.9 MBytes 17.4 Mbits/sec
[ 29] 0.0-10.1 sec 17.8 MBytes 14.7 Mbits/sec
[ 31] 0.0-10.1 sec 19.8 MBytes 16.4 Mbits/sec
[ 20] 0.0-10.1 sec 20.1 MBytes 16.7 Mbits/sec
[ 22] 0.0-10.1 sec 35.5 MBytes 29.4 Mbits/sec
[SUM] 0.0-10.1 sec 694 MBytes 575 Mbits/sec
vagrant@client:~$ iperf -c 10.10.2.2 -p 6980 -P 30

```

Fig. 8: Ejecución de iperf en el cliente.

Observando las imágenes anteriores, vemos que el reparto del ancho de banda de la Figura 2 se ha repartido de la manera esperada según el marcaje de los paquetes. En Fig. 7 y Fig. 8, que corresponden con el tráfico en uno de los puertos **RTP**, obtenemos el mayor ancho de banda (**575 Mbits/sec**) porque los paquetes estaban marcados como *Expedited Forwarding*. En Fig. 3. y Fig.4, se obtiene un ancho de banda de **474 Mbits/sec**, el menor de entre los tres casos, pues los paquetes están marcados como *Best Effort*. En Fig. 5. y Fig. 6, correspondientes con el tráfico **SIP**, es donde obtenemos un ancho de banda comprendido entre los dos anteriores (**500 Mbits/sec**); esto se justifica porque los paquetes están marcados como *Assured Forwarding*.

### 3.1. Captura de paquetes mediante Wireshark.

En esta sección, se muestran capturas de Wireshark de paquetes que simulan *RTP* y *SIP*, además de un paquete *TCP* perteneciente a un flujo de otro conjunto de puertos.

Para comprobar el marcaje del paquete, en la siguientes imágenes, observamos la línea “*Differentiated Services Field*”; en el campo DSCP, aparecerá **CS0** para el caso de los paquetes marcados como **Best Effort**, **AF11** para los paquetes marcados como **Assured Forwarding**, y **EF PHB** para los paquetes marcados como **Expedited Forwarding**.

#### ■ Expedited Forwarding.

No.	Time	Source	Destination	Protocol	Length	Info
7	0.003404677	10.10.2.2	10.10.1.2	TCP	66	6980 → 48488 [ACK] Seq=1 Ack=1
8	0.003469236	10.10.2.2	10.10.1.2	TCP	66	6980 → 48488 [ACK] Seq=1 Ack=1
9	0.003673530	10.10.1.2	10.10.2.2	TCP	14546	48488 → 6980 [PSH, ACK] Seq=13

▶	Frame 9: 14546 bytes on wire (116368 bits), 14546 bytes captured (116368 bits) on interface 0
▶	Ethernet II, Src: PcsCompu_ef:9f:c1 (08:00:27:ef:9f:c1), Dst: PcsCompu_2d:cd:ec (08:00:27:2d:cd:ec)
▼	Internet Protocol Version 4, Src: 10.10.1.2, Dst: 10.10.2.2
	0100 .... = Version: 4
	.... 0101 = Header Length: 20 bytes (5)
▶	Differentiated Services Field: 0xb8 (DSCP: EF PHB, ECN: Not-ECT)
	Total Length: 14532
	Identification: 0x61eb (25067)
▶	Flags: 0x4000, Don't fragment
	Time to live: 63
	Protocol: TCP (6)
	Header checksum: 0x8979 [validation disabled]
	[Header checksum status: Unverified]
	Source: 10.10.1.2
	Destination: 10.10.2.2
▶	Transmission Control Protocol, Src Port: 48488, Dst Port: 6980, Seq: 13057, Ack: 1, Len: 14480

0000	08 00 27 2d cd ec 08 00 27 ef 9f c1 08 00 45 b8	..'-....'.....E.
0010	38 c4 61 eb 40 00 3f 06 89 79 0a 0a 01 02 0a 0a	8:a@?..y.....
0020	02 02 bd 68 1b 44 80 62 81 00 83 6a 5f 4e 80 18	..hD b...j_N..
0030	01 c9 4f ce 00 00 01 01 08 0a 00 07 2c 21 00 06	..0.....,!..
0040	61 ab 32 33 34 35 36 37 38 39 30 31 32 33 34 35	a 234567 89012345
0050	36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31	67890123 45678901
0060	32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37	23456789 01234567
0070	38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33	89012345 67890123
0080	34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39	45678901 23456789

Figura 3: Paquetes con marcado Expedited Forwarding.

■ Assured Forwarding 11.

No.	Time	Source	Destination	Protocol	Length	Info
7	0.006373930	10.10.2.2	10.10.1.2	TCP	66	5060 → 41435 [ACK] Seq=1 A
8	0.007266991	10.10.1.2	10.10.2.2	SIP	14546	Continuation
9	0.007549905	10.10.1.2	10.10.2.2	SIP	14546	Continuation

▶ Frame 8: 14546 bytes on wire (116368 bits), 14546 bytes captured (116368 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_ef:9f:c1 (08:00:27:ef:9f:c1), Dst: PcsCompu_2d:cd:ec (08:00:27:2d:cd:ec) ▼ Internet Protocol Version 4, Src: 10.10.1.2, Dst: 10.10.2.2						
0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x28 (DSCP: AF11, ECN: Not-ECT) Total Length: 14532 Identification: 0xd1d7 (53719) ▶ Flags: 0x4000, Don't fragment Time to live: 63 Protocol: TCP (6) Header checksum: 0x1a1d [validation disabled] [Header checksum status: Unverified] Source: 10.10.1.2 Destination: 10.10.2.2						
▶ Transmission Control Protocol, Src Port: 41435, Dst Port: 5060, Seq: 13057, Ack: 1, Len: 14480						

0000	08 00 27 2d cd ec 08 00	27 ef 9f c1 08 00 45 28	..'-....'.....E(
0010	38 c4 d1 d7 40 00 3f 06	1a 1d 0a 0a 01 02 0a 0a	8...@?.....
0020	02 02 a1 db 13 c4 7a 27	d8 38 a2 29 4d cc 80 18	.....z'..8.)M...
0030	01 c9 4f ce 00 00 01 01	08 0a 00 06 74 fe 00 05	..0.....t...
0040	aa 5b 32 33 34 35 36 37	38 39 30 31 32 33 34 35	..[234567 89012345
0050	36 37 38 39 30 31 32 33	34 35 36 37 38 39 30 31	67890123 45678901
0060	32 33 34 35 36 37 38 39	30 31 32 33 34 35 36 37	23456789 01234567
0070	38 39 30 31 32 33 34 35	36 37 38 39 30 31 32 33	89012345 67890123
0080	34 35 36 37 38 39 30 31	32 33 34 35 36 37 38 39	45678901 23456789

Figura 4: Paquetes con marcado Assured Forwarding.

■ Best Effort.

No.	Time	Source	Destination	Protocol	Length	Info
11544	6.218832565	10.10.1.2	10.10.2.2	TCP	1514	44724 → 1234 [ACK] Seq=4199
11545	6.218846818	10.10.1.2	10.10.2.2	TCP	1514	44724 → 1234 [ACK] Seq=4199
11546	6.218859565	10.10.1.2	10.10.2.2	TCP	1514	44724 → 1234 [ACK] Seq=4199
▶ Frame 11545: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_ef:9f:c1 (08:00:27:ef:9f:c1), Dst: PcsCompu_2d:cd:ec (08:00:27:2d:cd:ec) ▼ Internet Protocol Version 4, Src: 10.10.1.2, Dst: 10.10.2.2 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 1500 Identification: 0x3f19 (16153) ▶ Flags: 0x4000, Don't fragment Time to live: 63 Protocol: TCP (6) Header checksum: 0xdfef [validation disabled] [Header checksum status: Unverified] Source: 10.10.1.2 Destination: 10.10.2.2 ▶ Transmission Control Protocol, Src Port: 44724, Dst Port: 1234, Seq: 419920001, Ack: 1, Len: 1448						
0000	08 00 27 2d cd ec 08 00	27 ef 9f c1 08 00 45 00	..'-....'.....E..			
0010	05 dc 3f 19 40 00 3f 06	df eb 0a 0a 01 02 0a 0a	..?@.?. ....			
0020	02 02 ae b4 04 d2 10 7c	b7 4d 1b 72 6f 2d 80 10	..... ..M-ro...			
0030	01 c9 1c e6 00 00 01 01	08 0a 00 03 60 38 00 02	.....`8...			
0040	94 d6 36 37 38 39 30 31	32 33 34 35 36 37 38 39	..678901 23456789			
0050	30 31 32 33 34 35 36 37	38 39 30 31 32 33 34 35	01234567 89012345			
0060	36 37 38 39 30 31 32 33	34 35 36 37 38 39 30 31	67890123 45678901			
0070	32 33 34 35 36 37 38 39	30 31 32 33 34 35 36 37	23456789 01234567			
0080	38 39 30 31 32 33 34 35	36 37 38 39 30 31 32 33	89012345 67890123			

Figura 5: Paquetes con marcado Best Effort.

## 4. Incidencias surgidas en el transcurso de la práctica.

Durante el desarrollo de la práctica, han surgido importantes incidencias. La más destacada, fue la imposibilidad de generar tráfico RTP. Al principio, usábamos *sipp* para simular llamadas, lo que debería generar tanto tráfico SIP como RTP, pero no era así. Buscamos durante bastante tiempo por todos los medios, pero no conseguimos encontrar una solución al problema.

Tras esto, acudimos a tutorías, y comenzamos a usar *sipp* y generar tráfico en los puertos de RTP. Para ello, mandamos paquetes por los puertos asignados a RTP usando, primero, scripts propios en python, y después, el módulo *SimpleHTTPServer* desde el servidor, y *wget* en el cliente, con varios procesos. Por diferentes motivos, tampoco conseguimos de esta manera el objetivo de la práctica.

El siguiente paso fue intentarlo estableciendo llamadas reales; para ello, instalamos una interfaz gráfica y *linphone* en el cliente y servidor. Tras configurarlo, conseguimos hacer la llamada, pero al descolgar el programa se cerraba por un error. Dedicamos tiempo a informarnos sobre el error e intentar solucionarlo, pero vimos que muchas personas reportaban esta incidencia para la cual no había una solución. Pensamos que podía ser por la máquina utilizada (*ubuntu/trusty64*), pero probamos con una máquina Debian y tras instalar *linphone* y probar de nuevo a realizar la llamada, seguía produciéndose el mismo error por lo que decidimos no perder más tiempo en ello y optar por la última opción.

Esta última opción, aportada por el profesor, es la que se presenta en los epígrafes anteriores.

También hubo algunos inconvenientes, lejos de la generación de tráfico RTP, relacionados con la puesta en marcha del escenario, o las reglas *iptables*, pero se pudieron solucionar en un periodo de tiempo reducido.

## 5. Repositorio.

A continuación, adjuntamos el enlace de nuestro repositorio en *GitHub*, donde se pueden ver los distintos pasos que hemos ido realizando para abordar la práctica.

También están incluidos todos los archivos referentes a *Vagrant*, *Ansible*, *iptables*, etc., además de la carpeta “Memoria”.

El enlace del repositorio es:

[https://github.com/mlopezs/P4-GestionPrioridadTraficoIP\\_PISS](https://github.com/mlopezs/P4-GestionPrioridadTraficoIP_PISS)

## 6. Bibliografía

- Configuración reglas Iptables.

<https://netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-7.html#ss7.3>

<https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-packet-marking-10103-dscpvalues.html>

<https://community.rti.com/kb/how-set-dscp-flag-your-system>

- Acerca de sipp.

<http://sipp.sourceforge.net/doc/reference.html>