ALBERTO LEÓN LUENGO

MEMORIA PRÁCTICA 4: CALIDAD DE SERVICIO (CONTROL DE TRÁFICO EN LINUX)

ÍNDICE DE CONTENIDOS

[] 1. CONTROL DE TRÁFICO

} 1.1 SIN CONTROL DE TRÁFICO NI A LA ENTRADA NI A LA SALIDA

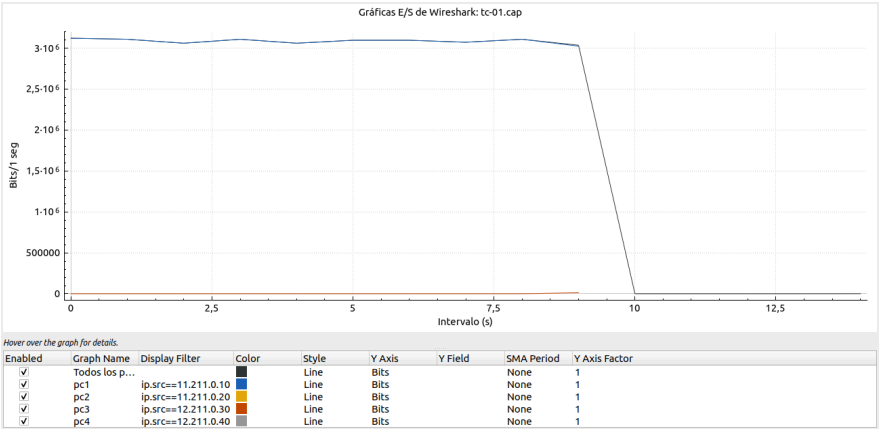
El router r1 no tiene activado el control de tráfico en ninguna de sus interfaces.

} 1.1.1 UN FLUJO DE DATOS

COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-01.cap (arrancar una captura de tráfico)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)
Una vez pasados 10 segundos, esto es lo que se observa en el servidor:

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec  3.57 MBytes  3.00 Mbits/sec  0.130 ms    1/ 2550 (0.039%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

COMANDO: user@f-IX-pcX:~$ wireshark tc-01.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Como se puede ver en la gráfica, se están enviando datagramas con un ancho de banda de 3 MBits/sec, y una vez transcurridos 10 segundos desde que se inició la comunicación, se dejan de enviar datagramas.

} 1.1.2 DOS FLUJOS DE DATOS

COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-02.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 12.211.0.40 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 1

Una vez pasados 10 segundos, esto es lo que se observa en los servidores pc4 y pc3, respectivamente:

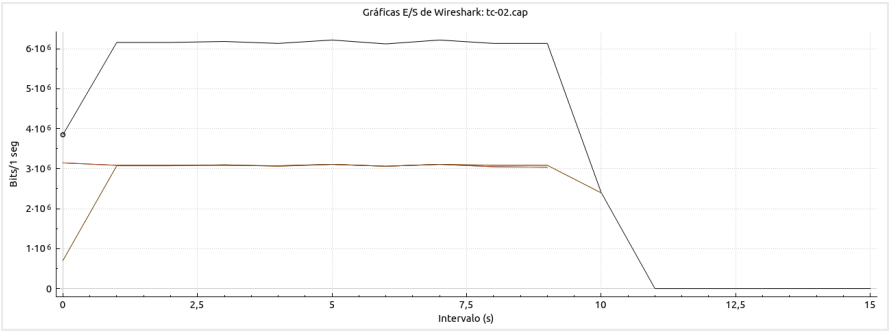
```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.40 port 5001 connected with 11.211.0.20 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec  3.58 MBytes  3.00 Mbits/sec  0.119 ms    1/ 2552 (0.039%)
```

```
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec  3.57 MBytes  3.00 Mbits/sec  0.154 ms    1/ 2550 (0.039%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

Como se puede ver, tanto el servidor de pc4 como en el de pc3 se está escuchando en el puerto UDP 5001 y se están recibiendo datagramas UDP (2552 en pc4 y 2550 en pc3) de 1470 bytes de tamaño durante un intervalo de 10 segundos. Además, se transfieren 3.58 MBytes mediante un ancho de banda de 3.00 MBits/sec de pc4 a pc2, mientras que en el caso de pc3 a pc1, se transfieren 3.57 MBytes mediante un ancho de banda de 3.00 MBits/sec.
Y por último, los jitters de ambos son bajos (0.119 ms en pc4 y 0.154 ms en pc3) y sólo se ha perdido un datagrama tanto en pc4 como en pc3, el cual entró fuera de orden.

PREGUNTA 2

COMANDO: user@f-IX-pcX:~$ wireshark tc-02.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Como se ha mencionado anteriormente, ambos flujos poseen un ancho de banda de 3.00 MBits/sec, con la diferencia que el flujo que va de pc1 a pc3 se mantiene muy próximo a este valor, mientras que el flujo de pc2 a pc4 sufre una subida y una bajada brusca de MBits/sec en comparación con la anterior.

} 1.2 CONTROL DE ADMISIÓN PARA EL TRÁFICO DE ENTRADA

Este es el script tc-ingress.sh que cumple con los requisitos solicitados:
COMANDO: r1:~# nano /hosthome/tc-ingress.sh (mostrar contenido del fichero tc-ingress.sh)

```
#!/bin/sh

INTERFACE="eth0"
PC1_IP="11.211.0.10/32"
PC2_IP="11.211.0.20/32"

# Borra la disciplina de cola si está definida, para que no dé un error.
if tc qdisc show dev $INTERFACE | grep -q "ingress"; then
    # Si devuelve algo por la salida, borra la disciplina de cola.
    tc qdisc del dev $INTERFACE ingress
fi

# Si no devuelve nada, crea la disciplina de cola.
tc qdisc add dev $INTERFACE ingress handle ffff:

# FLUJO 1
# Con la dirección IP origen de pc1 se quiere restringir con TBF a una velocidad de 1mbit y una cubeta de 10k.
tc filter add dev $INTERFACE parent ffff: protocol ip prio 4 u32 match ip src $PC1_IP police rate 1mbit burst 10k drop flowid :1

# FLUJO 2
# Con la dirección IP origen de pc2 se va a restringir a una velocidad de 2mbit y una cubeta de 10k.
tc filter add dev $INTERFACE parent ffff: protocol ip prio 5 u32 match ip src $PC2_IP police rate 2mbit burst 10k drop flowid :2
```

PREGUNTA 1
Ésta es la configuración de las disciplinas de cola configuradas inicialmente en r1:
COMANDO: r1:~# tc qdisc show dev eth0

```
qdisc pfifo_fast 0: root bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
```

Y ésta es la configuración de las disciplinas de cola una vez ejecutado el script tc-ingress.sh:
COMANDO: r1:~# /hosthome/tc-ingress.sh (ejecutar script tc-ingress.sh)
COMANDO: r1:~# tc qdisc show dev eth0

```
qdisc pfifo_fast 0: root bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc ingress ffff: parent ffff:fff1 ----------------
```

PREGUNTA 2
COMANDO: r1:~# /hosthome/tc-ingress.sh (ejecutar script tc-ingress.sh)
COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-03.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 12.211.0.40 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 3

Una vez pasados 10 segundos, esto es lo que se observa en los servidores pc4 y pc3, respectivamente:

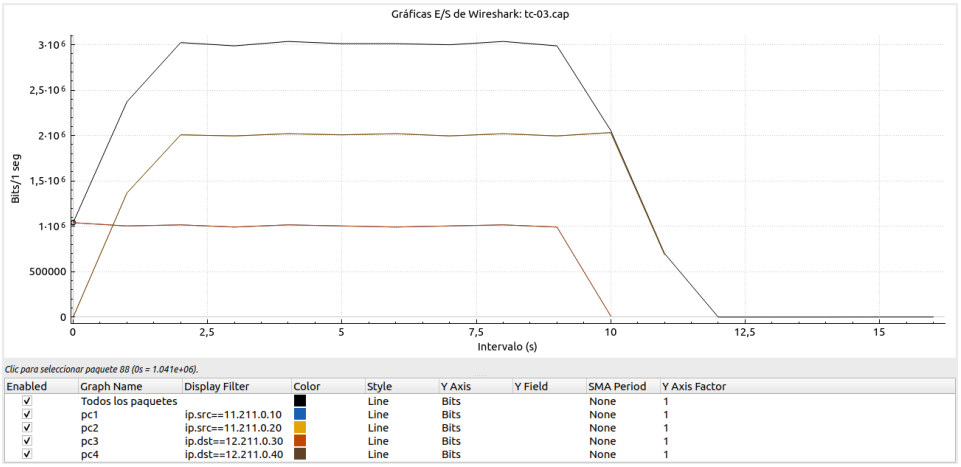
```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.40 port 5001 connected with 11.211.0.20 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec  2.33 MBytes  1.96 Mbits/sec  0.143 ms  887/ 2552 (35%)
```

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.2 sec  1.17 MBytes    958 Kbits/sec  15.916 ms 1718/ 2551 (67%)
[  3]  0.0-10.2 sec  1 datagrams received out-of-order
```

Como se puede ver, tanto el servidor de pc4 como en el de pc3 se está escuchando en el puerto UDP 5001 y se están recibiendo datagramas UDP (2552 en pc4 y 2551 en pc3) de 1470 bytes de tamaño durante un intervalo de 10 segundos en el caso de pc4 y 10.2 segundos en el caso de pc3.
Además, se transfieren 2.33 MBytes mediante un ancho de banda de 1.96 MBits/sec de pc4 a pc2, mientras que en el caso de pc3 a pc1, se transfieren 1.17 MBytes mediante un ancho de banda de 958 kbits/sec.
Y por último, el jitter de pc4 es muy bajo (0.143 ms), mientras que el de pc3 es bastante más alto con respecto al anterior (15.916 ms), y en cuanto a pérdidas de datagramas, pc4 ha perdido el 35% de sus datagramas (887/2552), mientras que pc3 ha perdido el 67% de los suyos (1718/2551), además de que pc3 ha recibido un datagrama fuera de orden.

PREGUNTA 4

COMANDO: user@f-IX-pcX:~$ wireshark tc-03.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Como se ha mencionado anteriormente, el flujo correspondiente a pc1 y pc3 posee un ancho de banda de 1 MBit/sec aproximadamente (958 kbits/sec), mientras que el de pc2 y pc4 es de 2 MBits/sec aproximadamente (1.96 MBits/sec).
Sin embargo, también podemos observar cómo los datos se envían a una velocidad de 3 MBits/sec debido a las reglas de filtrado definidas anteriormente en el script tc-ingress.sh, donde se ha limitado la velocidad de transmisión de ambos flujos a 1 MBit/sec y 2 MBits/sec respectivamente, además del tamaño máximo de la cubeta a 10 kbits.

PREGUNTA 5

Como se ha mencionado anteriormente, pc4 ha perdido el 35% de sus paquetes (887/2552), mientras que pc3 ha perdido el 67% de los suyos (1718/2551).

} 1.3 DISCIPLINAS DE COLA PARA EL TRÁFICO DE SALIDA

Este es el script tc-egress-tbf.sh que cumple con los requisitos solicitados:
COMANDO: r1:~# nano /hosthome/tc-egress-tbf.sh (mostrar contenido del fichero tc-egress-tbf.sh)

```
#!/bin/sh

INTERFACE="eth1"

# Disciplina TBF de salida con tasa de envío de 1.5mbit, tamaño de cubeta 10k y latencia 10ms.
tc qdisc add dev $INTERFACE root handle 1: tbf rate 1.5mbit burst 10k latency 10ms
```

} 1.3.1 TOKEN BUCKET FILTER (TBF)

COMANDO: r1:~# /hosthome/tc-egress-tbf.sh (ejecutar script tc-egress-tbf.sh)
COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-04.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 12.211.0.40 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 1
Una vez pasados 10 segundos, esto es lo que se observa en los servidores pc4 y pc3, respectivamente:

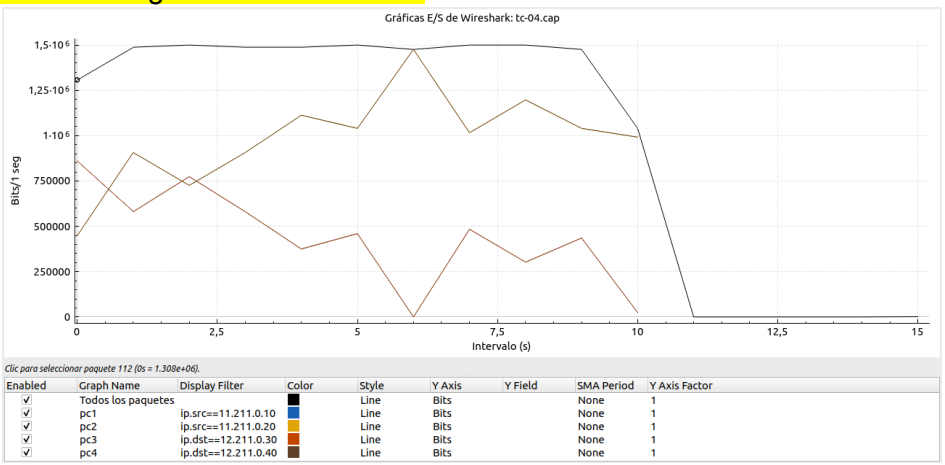
```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.40 port 5001 connected with 11.211.0.20 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.1 sec  1.26 MBytes  1.05 Mbits/sec  2.156 ms 1655/ 2552 (65%)
```

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.5 sec    577 KBytes    448 Kbits/sec  32.647 ms 2149/ 2551 (84%)
```

Como se puede ver, tanto el servidor de pc4 como en el de pc3 se está escuchando en el puerto UDP 5001 y se están recibiendo datagramas UDP (2552 en pc4 y 2551 en pc3) de 1470 bytes de tamaño durante un intervalo de 10.1 segundos en el caso de pc4 y 10.5 segundos en el caso de pc3.
Además, se transfieren 1.26 MBytes mediante un ancho de banda de 1.05 MBits/sec de pc4 a pc2, mientras que en el caso de pc3 a pc1, se transfieren 577 KBytes mediante un ancho de banda de 448 kbits/sec.
Y por último, el jitter de pc4 es bajo (2.156 ms), mientras que el de pc3 es bastante más alto con respecto al anterior (32.647 ms), y en cuanto a pérdidas de datagramas, pc4 ha perdido el 65% de sus datagramas (1655/2552), mientras que pc3 ha perdido el 84% de los suyos (2149/2551).

PREGUNTA 2

COMANDO: user@f-IX-pcX:~$ wireshark tc-04.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Clic para seleccionar paquete 112 (0s = 1.308e+06).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	Todos los paquetes			Line	Bits		None	1
☑	pc1	ip.src==11.211.0.10		Line	Bits		None	1
☑	pc2	ip.src==11.211.0.20		Line	Bits		None	1
☑	pc3	ip.dst==12.211.0.30		Line	Bits		None	1
☑	pc4	ip.dst==12.211.0.40		Line	Bits		None	1

El ancho de banda del flujo de pc1 y pc3 comienza en 1 MBit/sec aproximadamente (configurado en el script tc-ingress.sh), para después disminuir y quedarse por debajo de los 500 kbits/sec.
Sin embargo, la limitación de la velocidad del tráfico de salida establecida en el script tc-egress-tbf.sh mediante TBF (1.5 MBits/sec) no afecta a dicho flujo ya que éste retransmite a una velocidad menor.
Por otro lado, el flujo que va de pc2 a pc4 sí se ve afectado, ya que su filtro de entrada posee un límite de velocidad y un ancho de banda mayores que el del otro flujo, por lo que su tráfico se sitúa a poco menos de 1.5 MBits/seg debido a la limitación de la velocidad del tráfico de salida establecida en tc-egress-tbf.sh.

Estas son las modificaciones añadidas al script tc-egress-tbf.sh que cumple con los requisitos solicitados:
COMANDO: r1:~# nano /hosthome/tc-egress-tbf.sh (mostrar contenido del fichero tc-egress-tbf.sh)

```
#!/bin/sh

INTERFACE="eth1"

# Disciplina TBF de salida con tasa de envio de 1.5mbit, tamaño de cubeta 10k y latencia 20s.
tc qdisc add dev $INTERFACE root handle 1: tbf rate 1.5mbit burst 10k latency 20s
```

Para que éstas modificaciones hagan efecto, se apaga y enciende r1 una vez modificado el fichero, y ya dentro de r1, se establece la configuración que se nos pide, en este caso:
COMANDO: r1:~# /hosthome/tc-ingress.sh (ejecutar script tc-ingress.sh)
COMANDO: r1:~# /hosthome/tc-egress-tbf.sh (ejecutar script tc-egress-tbf.sh)

PREGUNTA 3

COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-05.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 12.211.0.40 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)
Una vez pasados 20 segundos, esto es lo que se observa en los servidores pc4 y pc3, respectivamente:

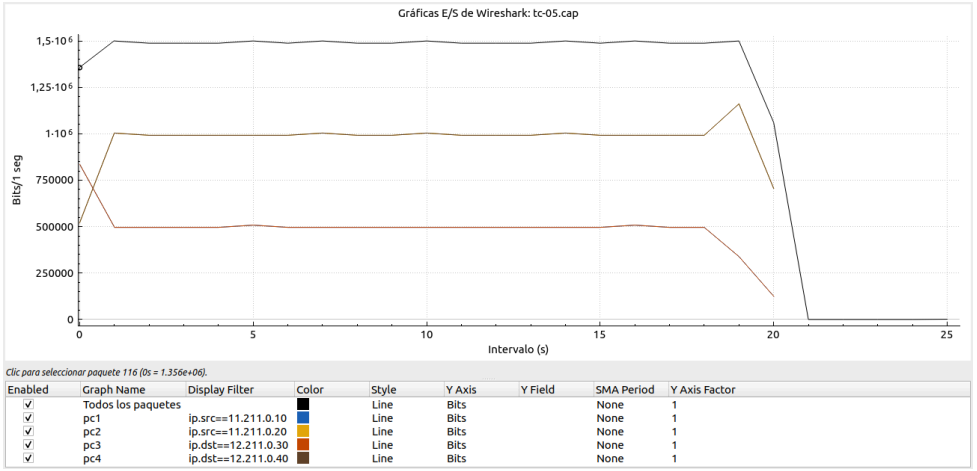
```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.40 port 5001 connected with 11.211.0.20 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-19.8 sec  2.33 MBytes    987 Kbits/sec  6.312 ms  885/ 2550 (35%)
[  3]  0.0-19.8 sec  1 datagrams received out-of-order
read failed: Connection refused
```

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-20.0 sec  1.17 MBytes    492 Kbits/sec  34.350 ms 1715/ 2551 (67%)
[  3]  0.0-20.0 sec  1 datagrams received out-of-order
read failed: Connection refused
```

Como se puede ver, tanto el servidor de pc4 como en el de pc3 se está escuchando en el puerto UDP 5001 y se están recibiendo datagramas UDP (2550 en pc4 y 2551 en pc3) de 1470 bytes de tamaño durante un intervalo de 19.8 segundos en el caso de pc4 y 20.0 segundos en el caso de pc3.
Además, se transfieren 2.33 MBytes mediante un ancho de banda de 987 kbits/sec de pc4 a pc2, mientras que en el caso de pc3 a pc1, se transfieren 1.17 MBytes mediante un ancho de banda de 492 kbits/sec.
Y por último, el jitter de pc4 es de 6.312 ms, mientras que el de pc3 es bastante más alto con respecto al anterior (34.350 ms), y en cuanto a pérdidas de datagramas, pc4 ha perdido el 35% de sus datagramas (885/2550), mientras que pc3 ha perdido el 67% de los suyos (1715/2551), además de que tanto pc4 como pc3 han recibido un datagrama fuera de orden.
La gran diferencia que se puede ver en estos servidores con respecto a casos anteriores, es el mensaje “read failed: connection refused”, indicando que la conexión se cerró antes de que se pudieran leer todos los datagramas.

PREGUNTA 4

COMANDO: user@f-IX-pcX:~$ wireshark tc-05.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Como se puede observar, tanto el flujo de pc1 y pc3 como el de pc2 y pc4 siguen enviando paquetes habiendo sobrepasado 10 segundos, hasta que ambos llegan a los 20 segundos aproximadamente, ya que es el tiempo máximo establecido en el script tc-egress-tbf.sh que tienen los datagramas para ser retrasados por la cola antes de enviarse.

PREGUNTA 5

r1 ha tardado aproximadamente 20 segundos en mandar todo el tráfico. Tanto pc1 como pc2 envían datos a 3 MBits/sec, de los cuales filtra 1 MBit/sec para pc1, y 2 MBits/sec para pc2 (script tc-ingress.sh).

Inicialmente, tanto pc1 como pc2 envían la misma cantidad de MBits:

Tráfico Total = pc1 + pc2 = 3 MBits/sec x 10 s + 3 MBits/sec x 10 s = 60 Mbits

Sin embargo, en la interfaz eth0 de r1, se descartan la mitad de MBits respecto al tráfico inicial:

Tráfico r1 (eth0) = pc1 + pc2 = 1 MBits/sec x 10 s + 2 MBits/sec x 10 s = 30 Mbits

Y por último, en la interfaz eth1 de r1, se mantiene la misma cantidad de MBits (script tc-egress-tbf.sh):

Tráfico r1 (eth1) = pc1 + pc2 = 1.5 MBits/sec x 20 s = 30 Mbits

} 1.3.2 TBF + PRIO

Este es el script tc-egress-tbf-prio.sh que cumple con los requisitos solicitados:
COMANDO: r1:~# nano /hosthome/tc-egress-tbf-prio.sh (mostrar contenido del fichero tc-egress-tbf-prio.sh)

```
#!/bin/sh

INTERFACE="eth1"

# Configuración de TBF con un ancho de banda de 1.5mbit, cubeta de 10k y latencia 20s.
tc qdisc add dev $INTERFACE root handle 1: tbf rate 1.5mbit burst 10k latency 20s

# Crea la disciplina de cola.
tc qdisc add dev $INTERFACE parent 1:0 handle 10:0 prio

# PRIORIDAD 1 (MÁS PRIORITARIO)
# Tráfico de la dirección IP origen de pc1.
tc filter add dev $INTERFACE parent 10:0 prio 1 protocol ip u32 \
        match ip src 11.211.0.10/32 flowid 10:1

# PRIORIDAD 2 (PRIORIDAD INTERMEDIA)
# Tráfico de la dirección IP origen de pc2.
tc filter add dev $INTERFACE parent 10:0 prio 2 protocol ip u32 \
        match ip src 11.211.0.20/32 flowid 10:2

# PRIORIDAD 3 (MENOS PRIORITARIO)
# Sin definir, ya que no se necesita.
```

COMANDO: r1:~# /hosthome/tc-egress-tbf-prio.sh (ejecutar script tc-egress-tbf.sh)
COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-06.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 12.211.0.40 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 1
Una vez pasados 20 segundos, esto es lo que se observa en los servidores pc4 y pc3, respectivamente:

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.40 port 5001 connected with 11.211.0.20 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-18.1 sec  2.04 MBytes    944 Kbits/sec  5.460 ms 1100/ 2552 (43%)
read failed: Connection refused
```

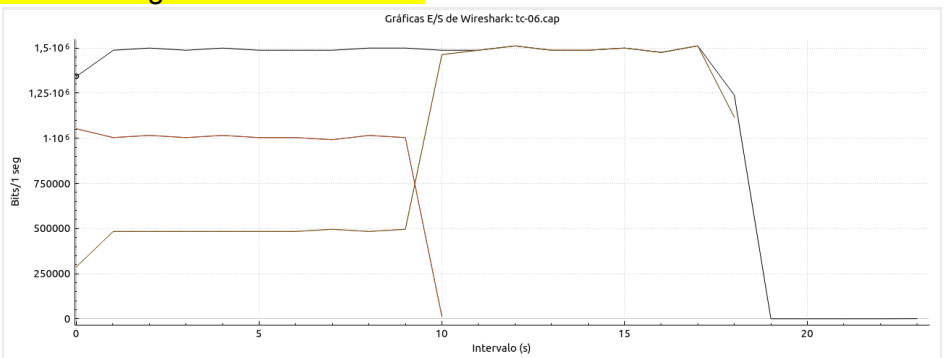
```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.2 sec  1.17 MBytes    960 Kbits/sec  20.936 ms 1714/ 2550 (67%)
[  3]  0.0-10.2 sec  1 datagrams received out-of-order
```

Como se puede ver, tanto el servidor de pc4 como en el de pc3 se está escuchando en el puerto UDP 5001 y se están recibiendo datagramas UDP (2552 en pc4 y 2550 en pc3) de 1470 bytes de tamaño durante un intervalo de 18.1 segundos en el caso de pc4 y 10.2 segundos en el caso de pc3.
Además, se transfieren 2.04 MBytes mediante un ancho de banda de 944 kbits/sec de pc4 a pc2, mientras que en el caso de pc3 a pc1, se transfieren 1.17 MBytes mediante un ancho de banda de 960 kbits/sec.
Y por último, el jitter de pc4 es de 5.460 ms, mientras que el de pc3 es bastante más alto con respecto al anterior (20.936 ms), y en cuanto a pérdidas de datagramas, pc4 ha perdido el 43% de sus datagramas (1100/2550), mientras que pc3 ha perdido el 67% de los suyos (1714/2550), además de que tpc3 ha recibido un datagrama fuera de orden.
Nuevamente aparece el mensaje “read failed: connection refused”, indicando que la conexión se cerró antes de que se pudieran leer todos los datagramas en pc4.

PREGUNTA 2

COMANDO: user@f-IX-pcX:~$ wireshark tc-06.cap (abrir una captura de tráfico en wireshark)

Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Clic para seleccionar paquete 115 (0s = 1.344e+06).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	Todos los paquetes			Line	Bits		None	1
☑	pc1	ip.src==11.211.0.10		Line	Bits		None	1
☑	pc2	ip.src==11.211.0.20		Line	Bits		None	1
☑	pc3	ip.dst==12.211.0.30		Line	Bits		None	1
☑	pc4	ip.dst==12.211.0.40		Line	Bits		None	1

Como se puede ver en la gráfica, el flujo que va de pc1 a pc3 se encuentra uniforme en 1 MBit/sec hasta que transcurren 8-9 segundos desde que se empezaron a mandar datagramas, donde dicho valor desciende y llega a 0 exactamente en el segundo 10.

Por otro lado, el flujo que va de pc2 a pc4 se encuentra uniforme en 500 kbits/sec hasta que transcurren 8-9 segundos desde que se empezaron a mandar datagramas al igual que en el caso anterior, con la diferencia que éste flujo aumenta hasta llegar a los 1.5 MBits/sec, y una vez pasados 10 segundos del inicio, se siguen enviando datagramas hasta llegar a 15 segundos, donde dicho flujo deja de enviar datagramas.

Con esto podemos concluir que, en este caso, se da prioridad al flujo que va de pc1 a pc3, y una vez éste termina, pasa al flujo que va de pc2 a pc4.

} 1.3.3 HIERARCHICAL TOKEN BUCKET (HTB)

Este es el script tc-egress-htb.sh que cumple con los requisitos solicitados:
COMANDO: r1:~# nano /hosthome/tc-egress-htb.sh (mostrar contenido del fichero tc-egress-htb.sh)

```
#!/bin/sh

INTERFACE="eth1"
PC1_IP="11.211.0.10/32"
PC2_IP="11.211.0.20/32"

# Crea la disciplina de cola.
tc qdisc add dev $INTERFACE root handle 1:0 htb

# Disciplina HTB de salida con ancho de banda 1.2mbit.
tc class add dev $INTERFACE parent 1:0 classid 1:1 htb rate 1.2mbit

# 700kbit para el tráfico con origen en pc1 (ceil 700kbit).
tc class add dev $INTERFACE parent 1:1 classid 1:2 htb rate 700kbit ceil 700kbit

# 500kbit para el tráfico con origen en pc2 (ceil 500kbit).
tc class add dev $INTERFACE parent 1:1 classid 1:3 htb rate 500kbit ceil 500kbit

# FLUJO 1
tc filter add dev $INTERFACE parent 1:0 protocol ip prio 1 u32 \
        match ip src $PC1_IP \
        flowid 1:2

# FLUJO 2
tc filter add dev $INTERFACE parent 1:0 protocol ip prio 1 u32 \
        match ip src $PC2_IP \
        flowid 1:3
```

COMANDO: r1:~# /hosthome/tc-egress-tbf-prio.sh (ejecutar script tc-egress-tbf.sh)
COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-07.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 12.211.0.40 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 1
Una vez pasados 20 segundos, esto es lo que se observa en los servidores pc4 y pc3, respectivamente:

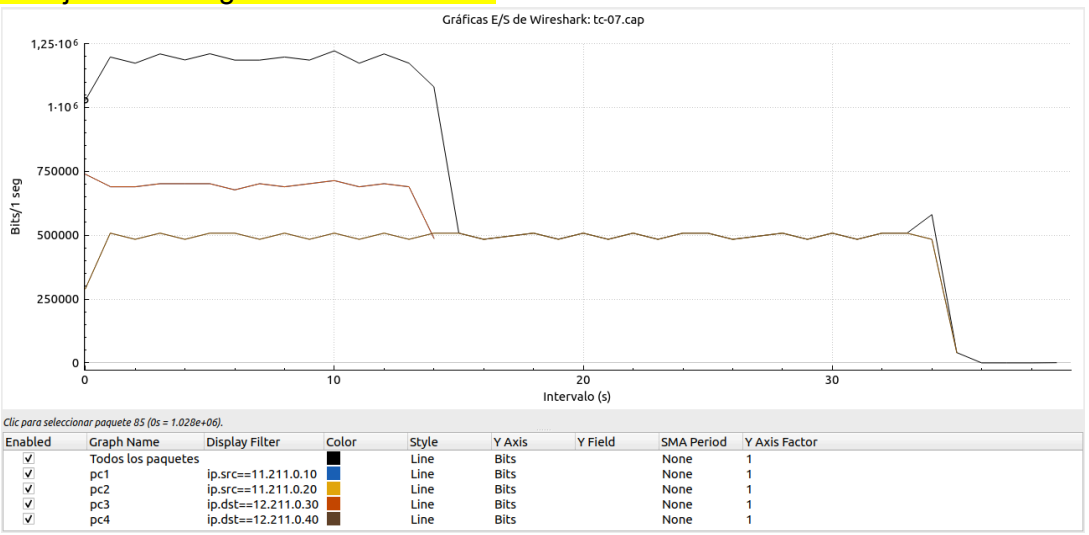
```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.40 port 5001 connected with 11.211.0.20 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-34.4 sec  1.98 MBytes    484 Kbits/sec  7.073 ms 1137/ 2551 (45%)
read failed: Connection refused
```

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-14.5 sec  1.17 MBytes    678 Kbits/sec  20.136 ms 1713/ 2551 (67%)
[  3]  0.0-14.5 sec  1 datagrams received out-of-order
read failed: Connection refused
```

Como se puede ver, tanto el servidor de pc4 como en el de pc3 se está escuchando en el puerto UDP 5001 y se están recibiendo datagramas UDP (2551 tanto en pc4 como en pc3) de 1470 bytes de tamaño durante un intervalo de 34.4 segundos en el caso de pc4 y 14.5 segundos en el caso de pc3.
Además, se transfieren 1.98 MBytes mediante un ancho de banda de 484 kbits/sec de pc4 a pc2, mientras que en el caso de pc3 a pc1, se transfieren 1.17 MBytes mediante un ancho de banda de 678 kbits/sec.
Y por último, el jitter de pc4 es de 7.073 ms, mientras que el de pc3 es bastante más alto con respecto al anterior (20.136 ms), y en cuanto a pérdidas de datagramas, pc4 ha perdido el 45% de sus datagramas (1137/2551), mientras que pc3 ha perdido el 67% de los suyos (1713/2551), además de que pc3 ha recibido un datagrama fuera de orden.
Nuevamente aparece tanto en pc4 como en pc3 el mensaje “read failed: connection refused”, indicando que la conexión se cerró antes de que se pudieran leer todos los datagramas.

PREGUNTA 2

COMANDO: user@f-IX-pcX:~$ wireshark tc-07.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	Todos los paquetes			Line	Bits		None	1
☑	pc1	ip.src==11.211.0.10		Line	Bits		None	1
☑	pc2	ip.src==11.211.0.20		Line	Bits		None	1
☑	pc3	ip.dst==12.211.0.30		Line	Bits		None	1
☑	pc4	ip.dst==12.211.0.40		Line	Bits		None	1

Como se puede ver en la gráfica, el flujo que va de pc1 a pc3 posee un ancho de banda de 700 kbits/sec aproximadamente, mientras que el del flujo que va de pc2 a pc4 es de 500 kbits/sec aproximadamente. Además, en ambos flujos siguen llegando datagramas pasados 10 segundos debido a la configuración establecida en el script tc-egress-hbt.sh.

Estas son las modificaciones añadidas al script tc-egress-htb.sh que cumple con los requisitos solicitados:
COMANDO: r1:~# nano /hosthome/tc-egress-htb.sh (mostrar contenido del fichero tc-egress-htb.sh)

```
#!/bin/sh

INTERFACE="eth1"
PC1_IP="11.211.0.10/32"
PC2_IP="11.211.0.20/32"

# Crea la disciplina de cola.
tc qdisc add dev $INTERFACE root handle 1:0 htb

# Disciplina HTB de salida con ancho de banda 1.2mbit.
tc class add dev $INTERFACE parent 1:0 classid 1:1 htb rate 1.2mbit

# 700kbit para el tráfico con origen en pc1 (ceil 1.2mbit).
tc class add dev $INTERFACE parent 1:1 classid 1:2 htb rate 700kbit ceil 1.2mbit

# 500kbit para el tráfico con origen en pc2 (ceil 1.2mbit).
tc class add dev $INTERFACE parent 1:1 classid 1:3 htb rate 500kbit ceil 1.2mbit

# FLUJO 1
tc filter add dev $INTERFACE parent 1:0 protocol ip prio 1 u32 \
        match ip src $PC1_IP \
        flowid 1:2

# FLUJO 2
tc filter add dev $INTERFACE parent 1:0 protocol ip prio 1 u32 \
        match ip src $PC2_IP \
        flowid 1:3
```

Para que éstas modificaciones hagan efecto, se apaga y enciende r1 una vez modificado el fichero, y ya dentro de r1, se establece la configuración que se nos pide, en este caso:
COMANDO: r1:~# /hosthome/tc-ingress.sh (ejecutar script tc-ingress.sh)
COMANDO: r1:~# /hosthome/tc-egress-htb.sh (ejecutar script tc-egress-htb.sh)

COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc3:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/tc-08.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 12.211.0.30 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 12.211.0.40 -b 3M (arrancar iperf en modo cliente UDP)
COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 3

Una vez pasados 20 segundos, esto es lo que se observa en los servidores pc4 y pc3, respectivamente:

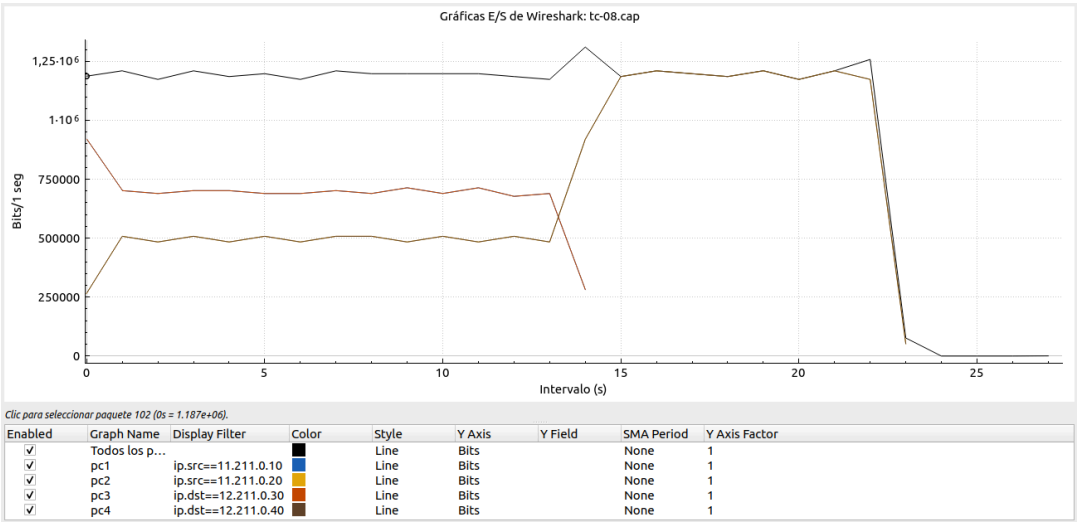
```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.40 port 5001 connected with 11.211.0.20 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-22.4 sec  1.98 MBytes    743 Kbits/sec  22.083 ms 1137/ 2551 (45%)
read failed: Connection refused
```

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   108 KByte (default)
------------------------------------------------------------
[  3] local 12.211.0.30 port 5001 connected with 11.211.0.10 port 32768
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-14.2 sec  1.17 MBytes    692 Kbits/sec  19.400 ms 1711/ 2547 (67%)
[  3]  0.0-14.2 sec  1 datagrams received out-of-order
read failed: Connection refused
```

Como se puede ver, tanto el servidor de pc4 como en el de pc3 se está escuchando en el puerto UDP 5001 y se están recibiendo datagramas UDP (2551 en pc4 y 2547 en pc3) de 1470 bytes de tamaño durante un intervalo de 22.4 segundos en el caso de pc4 y 14.2 segundos en el caso de pc3.
Además, se transfieren 1.98 MBytes mediante un ancho de banda de 743 kbits/sec de pc4 a pc2, mientras que en el caso de pc3 a pc1, se transfieren 1.17 MBytes mediante un ancho de banda de 692 kbits/sec.
Y por último, el jitter de pc4 es de 22.083 ms, mientras que el de pc3 es de 19.400 ms, y en cuanto a pérdidas de datagramas, pc4 ha perdido el 45% de sus datagramas (1137/2551), mientras que pc3 ha perdido el 67% de los suyos (1711/2551), además de que pc3 ha recibido un datagrama fuera de orden.
Nuevamente aparece tanto en pc4 como en pc3 el mensaje “read failed: connection refused”, indicando que la conexión se cerró antes de que se pudieran leer todos los datagramas.

PREGUNTA 4

COMANDO: user@f-lX-pcX:~$ wireshark tc-08.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Como se puede ver en la gráfica, ambos flujos siguen el mismo ancho de banda que en la gráfica anterior, con la diferencia de que, cuando el flujo que va de pc1 a pc3 termina de enviar datos fuera de tiempo, hace que el flujo que va de pc2 a pc4 use los 1.2 MBits/sec de ancho de banda máximos establecidos en la nueva configuración de su campo ceil en el script tc-egress-htb.sh.

[] 2. DIFFSERV

} 2.1 CONFIGURACIÓN DE FUNCIÓN POLICING Y MARCADO DE TRÁFICO EN DSCP

PREGUNTA 1

Este es el script diffServ-r1.sh que cumple con los requisitos solicitados:

COMANDO: r1:~# nano /hosthome/diffServ-r1.sh (mostrar contenido del fichero diffServ-r1.sh)

```
#!/bin/sh

PC1_IP="11.211.0.10/32"
PC2_IP="11.211.0.20/32"

tc qdisc add dev eth0 ingress handle ffff:

# FLUJO 1
# Máximo 1.2mbit con ráfaga 10k para el tráfico dirigido a pc4, marcado con calidad EF.
# Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 2.
tc filter add dev eth0 parent ffff: \
        protocol ip prio 4 u32 \
        match ip src $PC1_IP \
        police rate 1.2mbit burst 10k continue flowid :1

# FLUJO 2
# Máximo de 600kbit y ráfaga 10k, marcado con calidad AF31.
# Si se supera este ancho de banda, el tráfico será descartado definitivamente en r1.
tc filter add dev eth0 parent ffff: \
        protocol ip prio 5 u32 \
        match ip src $PC1_IP \
        police rate 600kbit burst 10k drop flowid :2

# FLUJO 3
# Máximo 300kbit con ráfaga 10k para el tráfico dirigido a pc5, marcado con AF21.
# Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 4.
tc filter add dev eth0 parent ffff: \
        protocol ip prio 4 u32 \
        match ip src $PC2_IP \
        police rate 300kbit burst 10k continue flowid :3

# FLUJO 4
# Máximo de 400kbit y ráfaga 10k, marcado con AF11.
# Si se supera este ancho de banda, el tráfico será descartado definitivamente en r1.
tc filter add dev eth0 parent ffff: \
        protocol ip prio 5 u32 \
        match ip src $PC2_IP \
        police rate 400kbit burst 10k drop flowid :4

# Crea la disciplina de cola de salida DSMARK.
tc qdisc add dev eth1 root handle 1:0 dsmark indices 8

tc class change dev eth1 classid 1:1 dsmark mask 0x3 value 0xb8
tc class change dev eth1 classid 1:2 dsmark mask 0x3 value 0x68
tc class change dev eth1 classid 1:3 dsmark mask 0x3 value 0x48
tc class change dev eth1 classid 1:4 dsmark mask 0x3 value 0x28

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
        handle 1 tcindex classid 1:1

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
        handle 2 tcindex classid 1:2

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
        handle 3 tcindex classid 1:3

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
        handle 4 tcindex classid 1:4
```

Este es el script diffServ-r2.sh que cumple con los requisitos solicitados:

COMANDO: r2:~# nano /hosthome/diffServ-r2.sh (mostrar contenido del fichero diffServ-r2.sh)

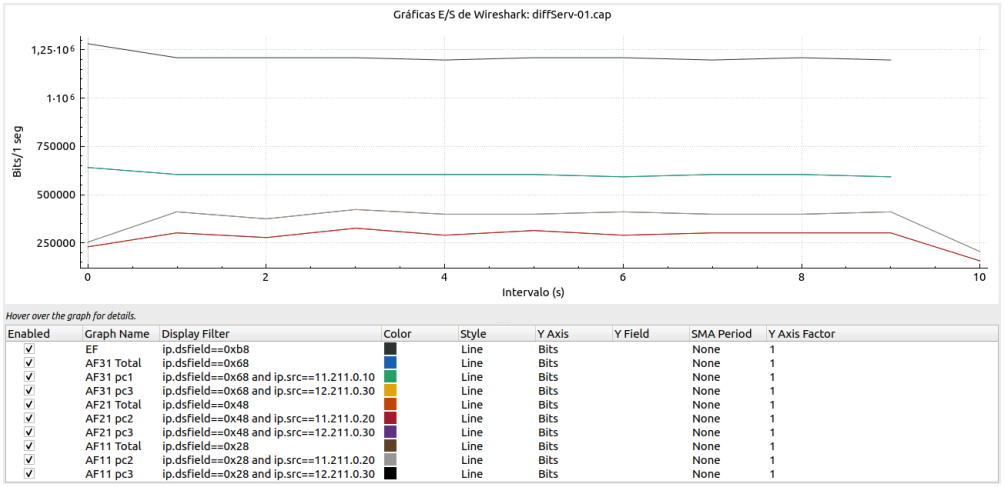
```
#!/bin/sh

PC3_IP="12.211.0.30/32"

tc qdisc add dev eth0 ingress handle ffff:

# FLUJO 5
# Máximo 400kbit con ráfaga 10k dirigido a pc6, marcado con AF31.
# Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 6.
tc filter add dev eth0 parent ffff: \
        protocol ip prio 4 u32 \
        match ip src $PC3_IP \
        police rate 400kbit burst 10k continue flowid :5

# FLUJO 6
# Máximo 300kbit con ráfaga 10k dirigido a pc6, marcado con AF21.
# Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 7.
tc filter add dev eth0 parent ffff: \
        protocol ip prio 5 u32 \
        match ip src $PC3_IP \
        police rate 300kbit burst 10k continue flowid :6

# FLUJO 7
# Máximo 100kbit con ráfaga 10k, marcado con AF11.
# Si se supera este ancho de banda, el tráfico será descartado definitivamente en r2.
tc filter add dev eth0 parent ffff: \
        protocol ip prio 6 u32 \
        match ip src $PC3_IP \
        police rate 100kbit burst 10k drop flowid :7

# Crea la disciplina de cola de salida DSMARK.
tc qdisc add dev eth1 root handle 1:0 dsmark indices 8

tc class change dev eth1 classid 1:5 dsmark mask 0x3 value 0x68
tc class change dev eth1 classid 1:6 dsmark mask 0x3 value 0x48
tc class change dev eth1 classid 1:7 dsmark mask 0x3 value 0x28

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
        handle 5 tcindex classid 1:5

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
        handle 6 tcindex classid 1:6

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
        handle 7 tcindex classid 1:7
```

PREGUNTA 2

COMANDO: r1:~# /hosthome/diffServ-r1.sh (ejecutar script diffServ-r1.sh)
COMANDO: r2:~# /hosthome/diffServ-r2.sh (ejecutar script diffServ-r2.sh)
COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc5:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc6:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r1:~# tcpdump -i eth1 -s 0 -w /hosthome/diffServ-01.cap (arrancar una captura de tráfico)
COMANDO: r2:~# tcpdump -i eth1 -s 0 -w /hosthome/diffServ-02.cap (arrancar una captura de tráfico)
COMANDO: r3:~# tcpdump -i eth2 -s 0 -w /hosthome/diffServ-03.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 16.211.0.40 -b 2M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 16.211.0.50 -b 1.5M (arrancar iperf en modo cliente UDP)
COMANDO: pc3:~# iperf -u -c 16.211.0.60 -b 1M (arrancar iperf en modo cliente UDP)

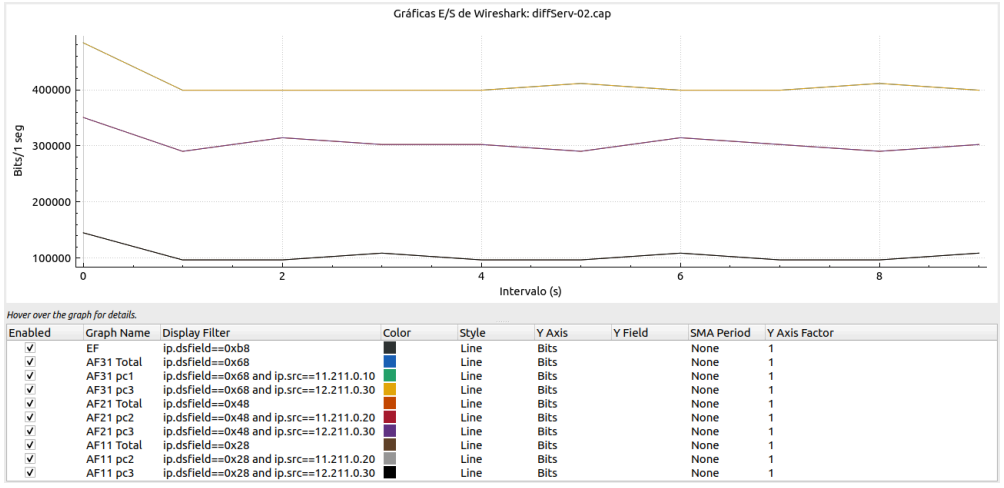
PREGUNTA 3

COMANDO: r1:~# ctrl+c (interrumpir una captura de tráfico)
COMANDO: r2:~# ctrl+c (interrumpir una captura de tráfico)
COMANDO: r3:~# ctrl+c (interrumpir una captura de tráfico)
COMANDO: user@f-lX-pcX:~$ wireshark diffServ-01.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



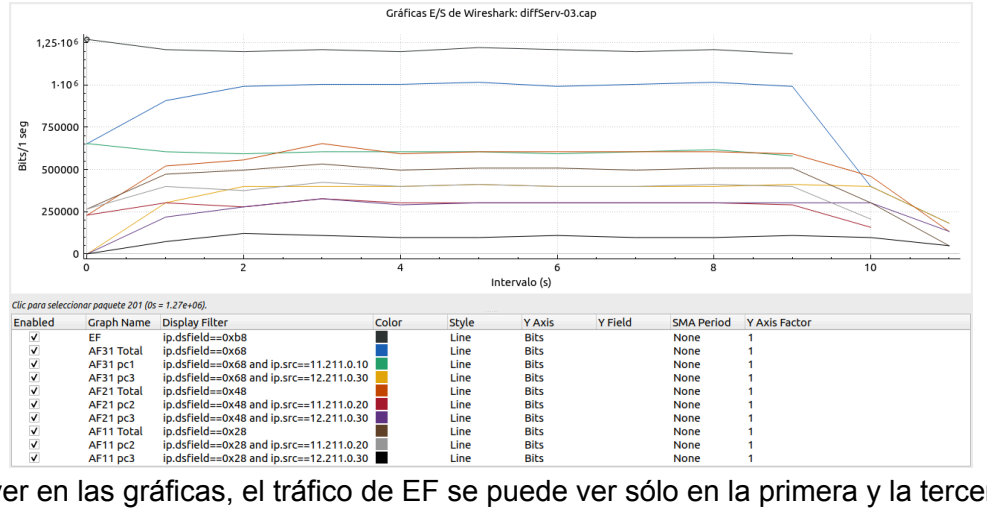
Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	EF	ip.dsfield==0xb8		Line	Bits		None	1
☑	AF31 Total	ip.dsfield==0x68		Line	Bits		None	1
☑	AF31 pc1	ip.dsfield==0x68 and ip.src==11.211.0.10		Line	Bits		None	1
☑	AF31 pc3	ip.dsfield==0x68 and ip.src==12.211.0.30		Line	Bits		None	1
☑	AF21 Total	ip.dsfield==0x48		Line	Bits		None	1
☑	AF21 pc2	ip.dsfield==0x48 and ip.src==11.211.0.20		Line	Bits		None	1
☑	AF21 pc3	ip.dsfield==0x48 and ip.src==12.211.0.30		Line	Bits		None	1
☑	AF11 Total	ip.dsfield==0x28		Line	Bits		None	1
☑	AF11 pc2	ip.dsfield==0x28 and ip.src==11.211.0.20		Line	Bits		None	1
☑	AF11 pc3	ip.dsfield==0x28 and ip.src==12.211.0.30		Line	Bits		None	1

COMANDO: user@f-lX-pcX:~$ wireshark diffServ-02.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	EF	ip.dsfield==0xb8		Line	Bits		None	1
☑	AF31 Total	ip.dsfield==0x68		Line	Bits		None	1
☑	AF31 pc1	ip.dsfield==0x68 and ip.src==11.211.0.10		Line	Bits		None	1
☑	AF31 pc3	ip.dsfield==0x68 and ip.src==12.211.0.30		Line	Bits		None	1
☑	AF21 Total	ip.dsfield==0x48		Line	Bits		None	1
☑	AF21 pc2	ip.dsfield==0x48 and ip.src==11.211.0.20		Line	Bits		None	1
☑	AF21 pc3	ip.dsfield==0x48 and ip.src==12.211.0.30		Line	Bits		None	1
☑	AF11 Total	ip.dsfield==0x28		Line	Bits		None	1
☑	AF11 pc2	ip.dsfield==0x28 and ip.src==11.211.0.20		Line	Bits		None	1
☑	AF11 pc3	ip.dsfield==0x28 and ip.src==12.211.0.30		Line	Bits		None	1

COMANDO: user@f-lX-pcX:~$ wireshark diffServ-03.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	EF	ip.dsfield==0xb8		Line	Bits		None	1
☑	AF31 Total	ip.dsfield==0x68		Line	Bits		None	1
☑	AF31 pc1	ip.dsfield==0x68 and ip.src==11.211.0.10		Line	Bits		None	1
☑	AF31 pc3	ip.dsfield==0x68 and ip.src==12.211.0.30		Line	Bits		None	1
☑	AF21 Total	ip.dsfield==0x48		Line	Bits		None	1
☑	AF21 pc2	ip.dsfield==0x48 and ip.src==11.211.0.20		Line	Bits		None	1
☑	AF21 pc3	ip.dsfield==0x48 and ip.src==12.211.0.30		Line	Bits		None	1
☑	AF11 Total	ip.dsfield==0x28		Line	Bits		None	1
☑	AF11 pc2	ip.dsfield==0x28 and ip.src==11.211.0.20		Line	Bits		None	1
☑	AF11 pc3	ip.dsfield==0x28 and ip.src==12.211.0.30		Line	Bits		None	1

Como se puede ver en las gráficas, el tráfico de EF se puede ver sólo en la primera y la tercera captura con un ancho de banda de 1.2 Mbits/sec, aunque la subred 12.211.0.0 (segunda captura) no marca ese tráfico. El resto de flujos (AF31, AF21 y AF11) sí pueden verse en todas las capturas, siguiendo el ancho de banda establecido en cada una de sus respectivas redes diffServ, donde el ancho de banda máximo de cada flujo equivale al ancho de banda final de los mismos.

} 2.2 TRATAMIENTO DE TRÁFICO EN FUNCIÓN DEL MARCADO DSCP

PREGUNTA 1

Este es el script diffServ-r3.sh que cumple con los requisitos solicitados:
COMANDO: r3:~# nano /hosthome/diffServ-r3.sh (mostrar contenido del fichero diffServ-r3.sh)

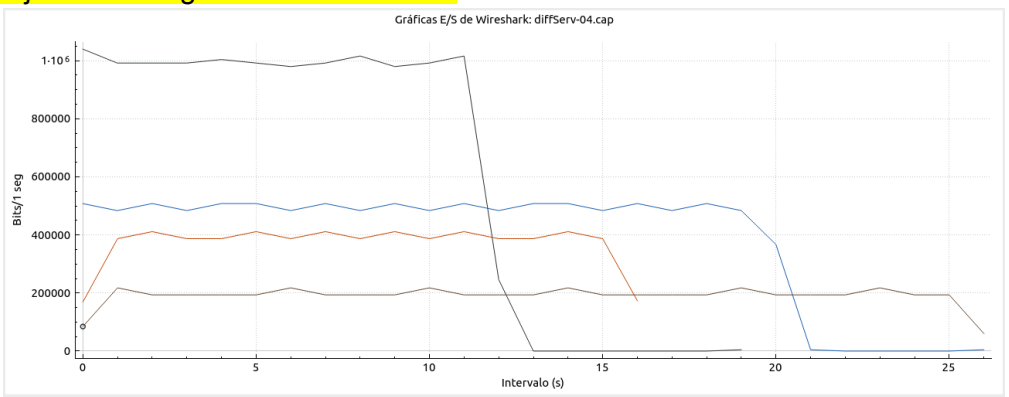
```
#!/bin/sh

tc qdisc add dev eth2 root handle 1:0 dsmark indices 8 set_tc_index

tc filter add dev eth2 parent 1:0 protocol ip prio 1 \
        tcindex mask 0xfc shift 2

tc qdisc add dev eth2 parent 1:0 handle 2:0 htb

# Configuración de HTB con ancho de banda 2.4Mbit para compartir entre todos los flujos.
tc class add dev eth2 parent 2:0 classid 2:1 htb rate 2.4Mbit

# EF: HTB 1Mbit como mínimo y 1Mbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:10 htb rate 1Mbit ceil 1Mbit

# AF31: HTB 500kbit como mínimo y 500kbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:11 htb rate 500kbit ceil 500kbit

# AF21: HTB 400kbit como mínimo y 400kbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:12 htb rate 400kbit ceil 400kbit

# AF11: HTB 200kbit como mínimo y 200kbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:13 htb rate 200kbit ceil 200kbit


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x2e tcindex classid 2:10


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x1a tcindex classid 2:11


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x12 tcindex classid 2:12


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x0a tcindex classid 2:13
```

PREGUNTA 2

COMANDO: r3:~# /hosthome/diffServ-r3.sh (ejecutar script diffServ-r3.sh)
COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc5:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc6:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r3:~# tcpdump -i eth2 -s 0 -w /hosthome/diffServ-04.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 16.211.0.40 -b 2M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 16.211.0.50 -b 1.5M (arrancar iperf en modo cliente UDP)
COMANDO: pc3:~# iperf -u -c 16.211.0.60 -b 1M (arrancar iperf en modo cliente UDP)
COMANDO: r3:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 3

COMANDO: r3:~# ctrl+c (interrumpir una captura de tráfico)
COMANDO: user@f-lX-pcX:~$ wireshark diffServ-04.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Como se puede ver en la gráfica, todos los flujos han terminado encolando tráfico una vez pasados los 10 segundos, siendo el flujo de EF el que menos tráfico encola, mientras que el flujo de AF11 ha sido el que más tráfico ha encolado.

PREGUNTA 4

Estas son las modificaciones añadidas al script diffServ-r3.sh que cumple con los requisitos solicitados:

COMANDO: r1:~# nano /hosthome/diffServ-r3.sh (mostrar contenido del fichero diffServ-r3.sh)

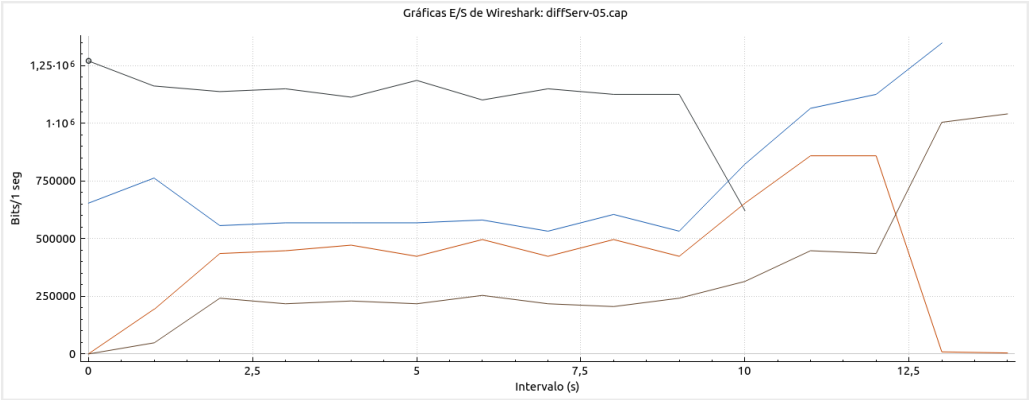
```
#!/bin/sh

tc qdisc add dev eth2 root handle 1:0 dsmark indices 8 set_tc_index

tc filter add dev eth2 parent 1:0 protocol ip prio 1 \
        tcindex mask 0xfc shift 2

tc qdisc add dev eth2 parent 1:0 handle 2:0 htb

# Configuración de HTB con ancho de banda 2.4Mbit para compartir entre todos los flujos.
tc class add dev eth2 parent 2:0 classid 2:1 htb rate 2.4Mbit

# EF: HTB 1Mbit como mínimo y 2.4Mbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:10 htb rate 1Mbit ceil 2.4Mbit

# AF31: HTB 500kbit como mínimo y 2.4Mbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:11 htb rate 500kbit ceil 2.4Mbit

# AF21: HTB 400kbit como mínimo y 2.4Mbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:12 htb rate 400kbit ceil 2.4Mbit

# AF11: HTB 200kbit como mínimo y 2.4Mbit como máximo.
tc class add dev eth2 parent 2:1 classid 2:13 htb rate 200kbit ceil 2.4Mbit


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x2e tcindex classid 2:10


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x1a tcindex classid 2:11


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x12 tcindex classid 2:12


tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
        handle 0x0a tcindex classid 2:13
```

Para que éstas modificaciones hagan efecto, se apaga y enciende r3 una vez modificado el fichero, y ya dentro de r3, se establece la configuración que se nos pide, en este caso:

COMANDO: r3:~# /hosthome/diffServ-r3.sh (ejecutar script diffServ-r3.sh)
COMANDO: pc4:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc5:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: pc6:~# iperf -u -s (arrancar iperf en modo servidor UDP)
COMANDO: r3:~# tcpdump -i eth2 -s 0 -w /hosthome/diffServ-05.cap (arrancar una captura de tráfico)
COMANDO: pc1:~# iperf -u -c 16.211.0.40 -b 2M (arrancar iperf en modo cliente UDP)
COMANDO: pc2:~# iperf -u -c 16.211.0.50 -b 1.5M (arrancar iperf en modo cliente UDP)
COMANDO: pc3:~# iperf -u -c 16.211.0.60 -b 1M (arrancar iperf en modo cliente UDP)
COMANDO: r3:~# ctrl+c (interrumpir una captura de tráfico)

PREGUNTA 5

COMANDO: r3:~# ctrl+c (interrumpir una captura de tráfico)
COMANDO: user@f-lX-pcX:~$ wireshark diffServ-05.cap (abrir una captura de tráfico en wireshark)
Para mostrar el flujo de forma gráfica en Wireshark: Estadísticas ⇒ Gráfica de E/S



Clic para seleccionar paquete 161 (0s = 1.27e+06).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	EF	ip.dsfield==0xb8		Line	Bits		None	1
☑	AF31	ip.dsfield==0x68		Line	Bits		None	1
☑	AF21	ip.dsfield==0x48		Line	Bits		None	1
☑	AF11	ip.dsfield==0x28		Line	Bits		None	1

Como se puede ver en la gráfica, el tráfico que llega pasados 10 segundos es considerablemente menor con respecto a la gráfica anterior. Esto ocurre debido a que, una vez los flujos dejan de usar el ancho de banda que se les había asignado, otro flujo puede emplear ese ancho de banda y aumentar el suyo, logrando así que el tiempo de llegada del tráfico pasados 10 segundos sea menor.