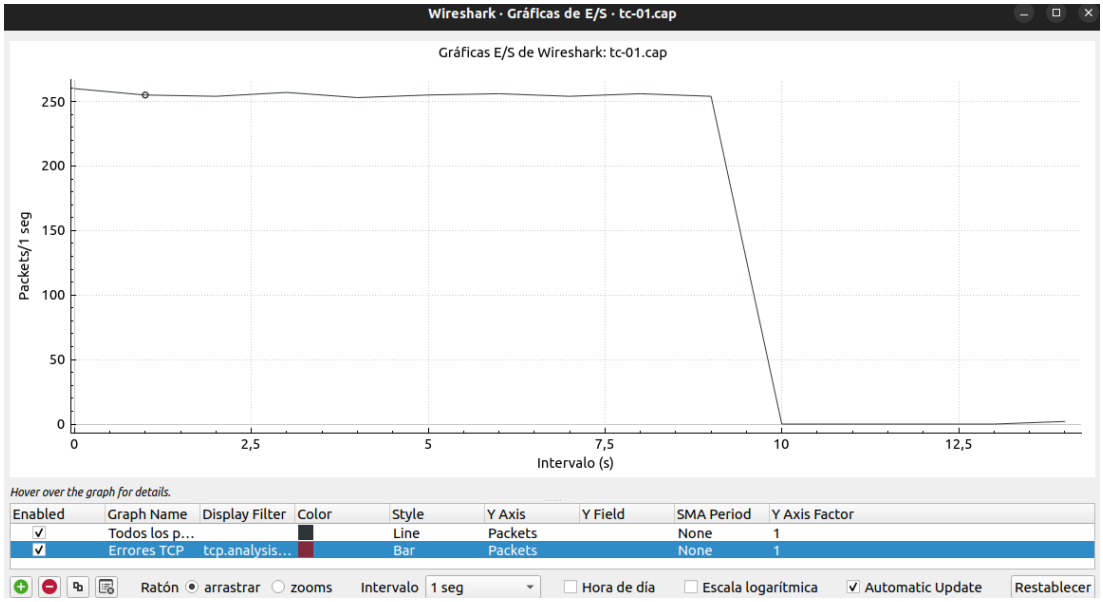
Práctica 4

1. Control de tráfico

1.1.1 Flujo de datos



1.1.2 Dos flujos de datos

pc3:~# iperf -u -s

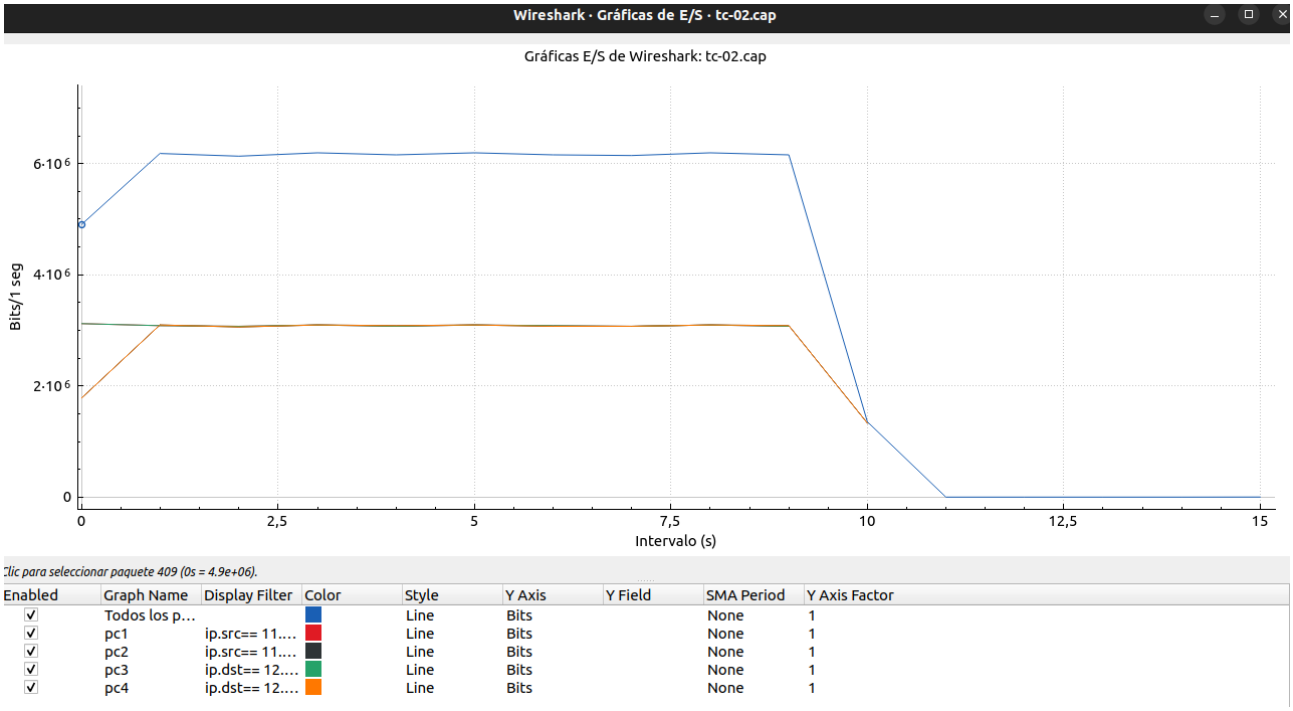
--
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.30 port 5001 connected with 11.188.0.10 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-10.0 sec 3.58 MBytes 3.00 Mbits/sec 0.068 ms 0/ 2550 (0%)
[3] 0.0-10.0 sec 1 datagrams received out-of-order

El resultado de la prueba de Iperf en pc3 indica que el servidor está escuchando en el puerto UDP 5001 y está recibiendo datagramas UDP de 1470 bytes de tamaño. Durante un intervalo de tiempo de 10 segundos, se transfirieron 3.58 MBytes de datos desde el cliente al servidor a un ancho de banda de 3.00 Mbits/sec. El jitter fue bajo (0.068 ms) y no se perdieron datagramas, excepto por un datagrama que se recibió fuera de orden.

pc4:~# iperf -u -s

--
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.40 port 5001 connected with 11.188.0.20 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-10.0 sec 3.57 MBytes 3.00 Mbits/sec 0.080 ms 1/ 2550 (0.039%)
[3] 0.0-10.0 sec 1 datagrams received out-of-order

El resultado de la prueba de Iperf en pc4 indica que el servidor está escuchando en el puerto UDP 5001 y está recibiendo datagramas UDP de 1470 bytes de tamaño. Durante un intervalo de tiempo de 10 segundos, se transfirieron 3.57 MBytes de datos desde el cliente al servidor a un ancho de banda de 3.00 Mbits/sec. El jitter fue bajo (0.080 ms), pero se perdieron 1 de los 2550 datagramas recibidos (0.039%). Además, se recibió un datagrama fuera de orden.



El ancho de banda medido en ambos flujos es de 3.00 Mbits/sec. Sin embargo, el flujo de pc1 → pc3, se mantiene muy próximo a 3 Mbits/sec, al contrario del flujo de pc2 → pc4, que sufre una subida y una bajada de Mbits/sec y sufre una pérdida de un paquete.

1.2 Control de admisión para el tráfico de entrada

Script empleado:

```
#!/bin/sh

INTERFACE="eth0"
PC1_IP="11.188.0.10/32"
PC2_IP="11.188.0.20/32"

# Verificar si ya existe una disciplina de cola en la interfaz
if tc qdisc show dev $INTERFACE | grep -q "ingress"; then
    # Si existe, eliminarla
    tc qdisc del dev $INTERFACE ingress
fi

tc qdisc add dev $INTERFACE ingress handle ffff:

# Agregar el filtro para el flujo 1. Restringir el flujo 1 a 1mbit/s y 10k de cubeta con TBF
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

# Agregar el filtro para el flujo 2. Restringir el flujo 2 a 2mbit/s y 10k de cubeta
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

1.2.1 Antes de ejecutar el script:
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

Después de ejecutar el script:
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 ----------

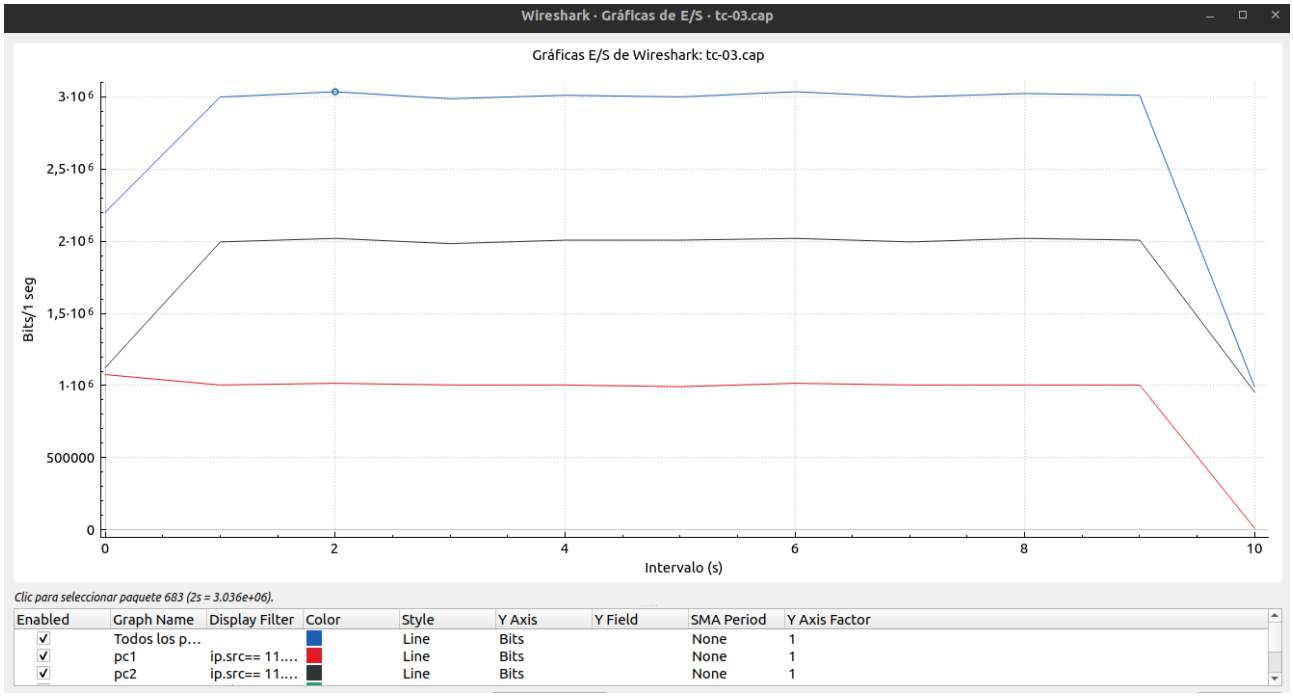
1.2.3 ***pc3:~# iperf -u -s***
--
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.30 port 5001 connected with 11.188.0.10 port 32768
[ID] Interval   Transfer   Bandwidth   Jitter   Lost/Total Datagrams
[3] 0.0-10.2 sec 1.17 MBytes   961 Kbits/sec 16.118 ms 1713/ 2550 (67%)
[3] 0.0-10.2 sec 1 datagrams received out-of-order

Durante la prueba, se enviaron *2550* datagramas y se perdieron *1713* (*67*%). El tamaño de cada datagrama fue de 1470 bytes y el tamaño del búfer UDP fue de 108 KBytes. Durante el intervalo de tiempo de 10.2 segundos, se transfirieron *1.17 MBytes* de datos a un ancho de banda de *961* Kbits/sec. El jitter (variación en el retardo de la transmisión) fue de *16.118* ms.

pc4:~# iperf -u -s
--
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.40 port 5001 connected with 11.188.0.20 port 32768
[ID] Interval   Transfer   Bandwidth   Jitter   Lost/Total Datagrams
[3] 0.0-10.0 sec 2.33 MBytes 1.96 Mbits/sec 0.087 ms 887/ 2551 (35%)
[3] 0.0-10.0 sec 1 datagrams received out-of-order

Durante un intervalo de 10.0 segundos, se transfirieron *2.33 MBytes* con un ancho de banda promedio de *1.96 Mbits/sec*. Además, se observó un jitter de *0.087* ms y se perdió el *35*% de los datagramas recibidos, con un total de *887* datagramas perdidos de un total de *2551*. También se informó que se recibió un datagrama fuera de orden durante la prueba.

1.2.4



En la gráfica podemos observar que el ancho de banda que se puede observar del flujo de pc1→pc3 es de 1Mbits/seg, a pesar de que se establece una velocidad de 3Mbits/seg para el envio de datos, al igual que el ancho de banda del flujo de pc2→pc4 el cual se puede observar en la gráfica que es de 2Mbits/seg. Esto se debe a la regla de filtrado creada para dichos flujos que limitan la velocidad de transmisión del flujo 1 y 2 a 1Mbits/seg y 2Mbits/seg, y el tamaño máximo de ráfaga a 10Kbits. Por ello se puede observar un ancho de banda mayor en el flujo de pc2, dado que tiene un limite superior al flujo 1.

1.2.5 En el caso del primer flujo, pc1→pc3, se perdieron 1713 paquetes, llegando únicamente 837 paquetes. En el caso del segundo flujo, pc2,→pc4, se perdieron 887 paquetes, llegando únicamente 1664 paquetes.

1.3 Disciplinas de colas para el tráfico de salida

1.3.1 Tocken Bucket Filter (TBF)

Script creado:
#!/bin/sh

INTERFACE="eth1"

tc qdisc add dev $INTERFACE root handle 1: tbf rate 1.5mbit burst 10k latency 10ms

1.3.1.1 pc3:~# iperf -u -s

--
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.30 port 5001 connected with 11.188.0.10 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-10.6 sec 446 KBytes 346 Kbits/sec 33.917 ms 2239/ 2550 (88%)
[3] 0.0-10.6 sec 1 datagrams received out-of-order

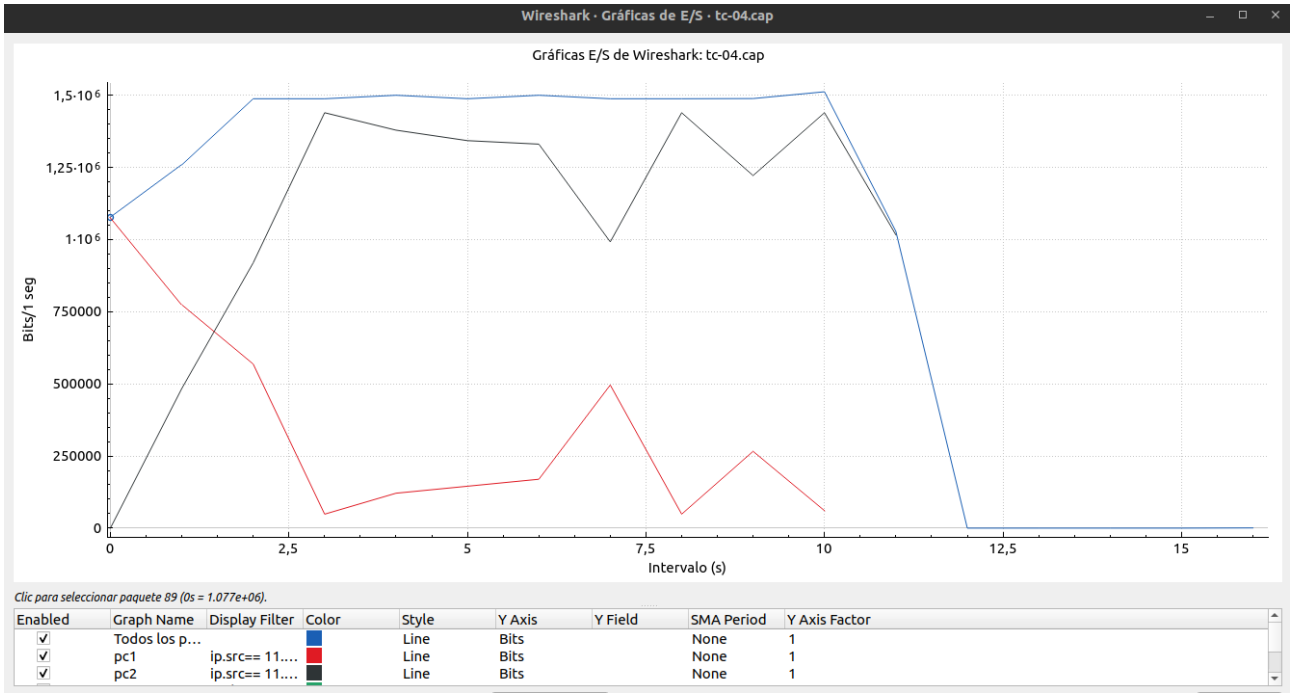
Durante la prueba, se enviaron *2550* datagramas y se perdieron *2239* (*88*%). El tamaño de cada datagrama fue de 1470 bytes y el tamaño del búfer UDP fue de 108 KBytes. Durante el intervalo de tiempo de 10.6 segundos, se transfirieron *446* KBytes de datos a un ancho de banda de *346* Kbits/sec. El jitter fue de *33.917* ms.

pc4:~# iperf -u -s

--
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.40 port 5001 connected with 11.188.0.20 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-10.2 sec 1.51 MBytes 1.24 Mbits/sec 16.490 ms 1476/ 2550 (58%)
[3] 0.0-10.2 sec 1 datagrams received out-of-order

Durante un intervalo de 10.2 segundos, se transfirieron *1.51 MBytes* con un ancho de banda promedio de *1.24 Mbits/sec*. Además, se observó un jitter de *16.490 ms* y se perdió el *58*% de los datagramas recibidos, con un total de *1476* datagramas perdidos de un total de *2550*. También se informó que se recibió un datagrama fuera de orden durante la prueba.

1.3.1.2



El ancho de banda del flujo de pc1→ pc3 comienza aproximadamente en 1Mbits/seg, la velocidad que le permite el filtro de entrada, y posteriormente disminuye hasta llegar por debajo de los 500Kbits/seg, sin embargo, la limitación de la velocidad del tráfico de salida impuesto de 1.5 Mbits/seg mediante el TBF no afecta a dicho flujo ya que retransmite a una velocidad menor del límite.
Por otro lado, el flujo de pc2→pc4 sí se ve afectado, dado que su filtro de entrada posee un límite de velocidad mayor que el del otro flujo, su ancho de banda final es mayor que el otro flujo. Su tráfico se sitúa a poco menos de 1.5 Mbits/seg debido a la limitación de la velocidad del tráfico de salida.

1.3.1.3 ***pc4:~# iperf -u -s***

--
Server listening on UDP port 5001
Receiving 1470 byte datagrams

UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.40 port 5001 connected with 11.188.0.20 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-16.2 sec 1.88 MBytes 971 Kbits/sec 92.593 ms 1208/ 2548 (47%)
[3] 0.0-16.2 sec 1 datagrams received out-of-order
read failed: Connection refused

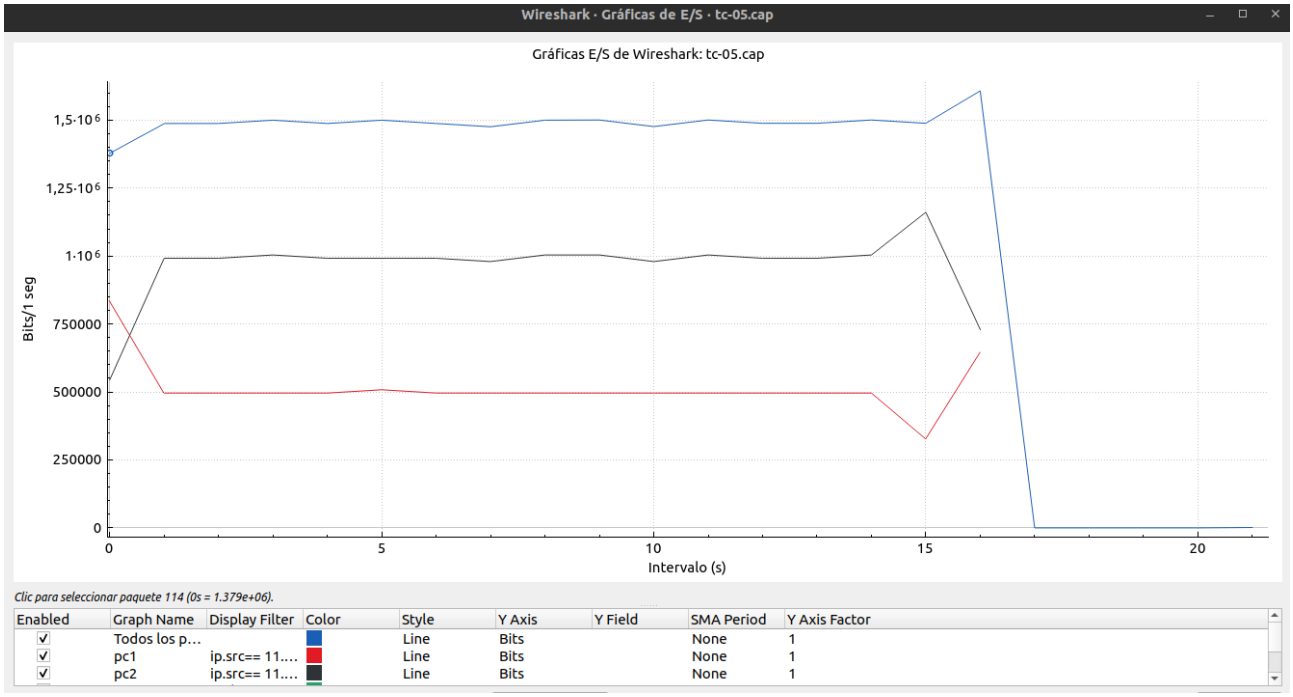
Durante la prueba, se enviaron *2550* datagramas y se perdieron *1208* (*47%*). Durante el intervalo de tiempo de 16.2 segundos, se transfirieron *1.88 MBytes* de datos a un ancho de banda de *971 Kbits/sec*. El jitter fue de *92.593* ms. También se informó que se recibió un datagrama fuera de orden durante la prueba.

pc3:~# iperf -u -s
--
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
--
[3] local 12.188.0.30 port 5001 connected with 11.188.0.10 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-16.7 sec 1024 KBytes 502 Kbits/sec 8.437 ms 1838/ 2551 (72%)
[3] 0.0-16.7 sec 1 datagrams received out-of-order
read failed: Connection refused

Durante un intervalo de *16.7* segundos, se transfirieron *1024 KBytes* con un ancho de banda promedio de *502 Kbits/sec*. Además, se observó un jitter de *8.437 ms* y se perdió el *72%* de los datagramas recibidos, con un total de *1838* datagramas perdidos de un total de *2551*. También se informó que se recibió un datagrama fuera de orden durante la prueba.

Además, en ambos resultados obtenemos un "*read failed*", lo que indica que se cerró la conexión antes de que se pudiera leer toda la información.

1.3.1.4



Tal como se puede observar en la gráfica, en ambos flujos continua el envío de paquetes después de haber finalizado el envío de datos de los 10 segundos hasta aproximadamente 16 segundos. El envío de datos de después de los 10 segundos se debe a la alta latencia de 20 segundos, dado que es la cantidad de tiempo que

un paquete de datos se puede retrasar en la cola antes de ser transmitido, por ello varios paquetes llegan después de los 10 segundos.

1.3.1.5 R1 tarda en reenviar todo el tráfico alrededor de 16 segundos aproximadamente. De la disciplina de cola asociada a eth0 de r1 se descartan alrededor de 22.5Mbps. De la disciplina de cola asociada a eth1 de r1 se descartan alrededor de 15Mbps.

1.3.2 TBF + PRIO

1.3.2.1 *pc3:~# iperf -u -s*

Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
---
[3] local 12.188.0.30 port 5001 connected with 11.188.0.10 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-11.5 sec 963 KBytes 687 Kbits/sec 96.602 ms 1880/ 2551 (74%)

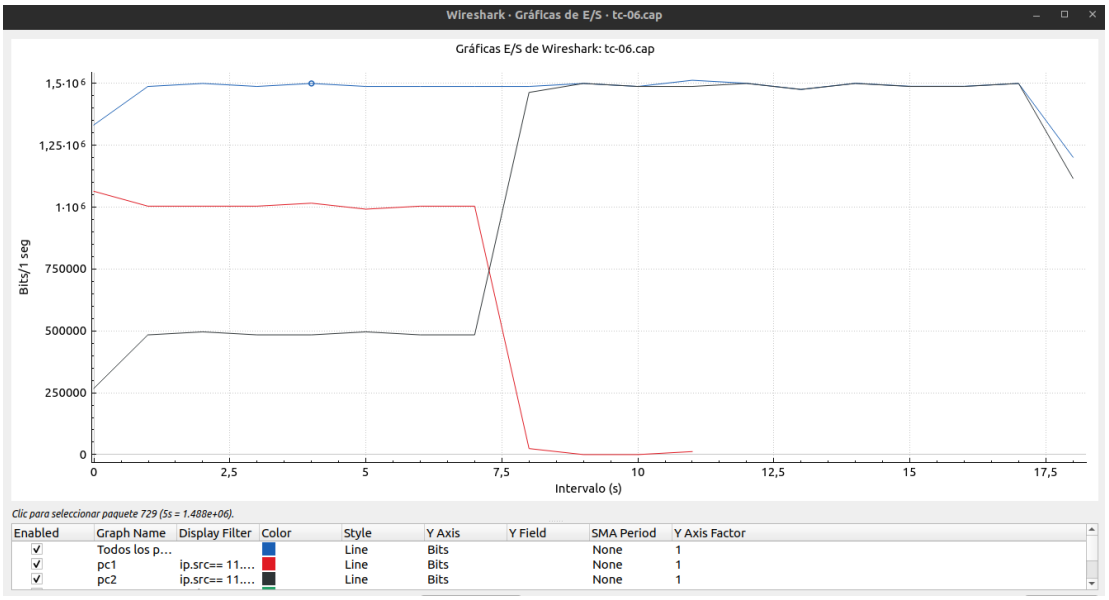
Durante un intervalo de *11.5* segundos, se transfirieron *963 KBytes* con un ancho de banda promedio de *687 Kbits/sec*. Además, se observó un jitter de *96.602 ms* y se perdió el *74*% de los datagramas recibidos, con un total de *1880* datagramas perdidos de un total de *2551*.

pc4:~# iperf -u -s

Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
---
[3] local 12.188.0.40 port 5001 connected with 11.188.0.20 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-18.1 sec 2.27 MBytes 1.05 Mbits/sec 4.046 ms 933/ 2549 (37%)
[3] 0.0-18.1 sec 1 datagrams received out-of-order
read failed: Connection refused

Durante la prueba, se enviaron *2549* datagramas y se perdieron *933* (*37*%). Durante el intervalo de tiempo de *18.1* segundos, se transfirieron *2.27 MBytes* de datos a un ancho de banda de *1.05 Mbits/sec*. El jitter fue de *4.046* ms. También se informó que se recibió un datagrama fuera de orden durante la prueba.

1.3.2.2



El ancho de banda del flujo pc1→pc3 se encuentra en 1Mbits/seg. Cerca de los 7 segundos comienza a descender, y después de los 10 segundos del envío de datos proceden a llegar paquetes también.
El ancho de banda del flujo de pc2→pc3 se encuentra en 500Kbits/seg los primeros 7 segundos, después aumenta hasta 1.5Mbits/seg llegando paquetes después de los 10 segundos del envío de datos hasta aproximadamente los 17.5 segundos.
Tal como podemos observar, se da prioridad principalmente al principio al flujo de pc1 y posteriormente la prioridad pasa al flujo de pc2.

1.3.3 Hierarchical token Bucket (HBT)

1.3.3.1 *pc3:~# iperf -u -s*

Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
---
[3] local 12.188.0.30 port 5001 connected with 11.188.0.10 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-14.5 sec 1.17 MBytes 680 Kbits/sec 19.800 ms 1714/ 2551 (67%)
read failed: Connection refused

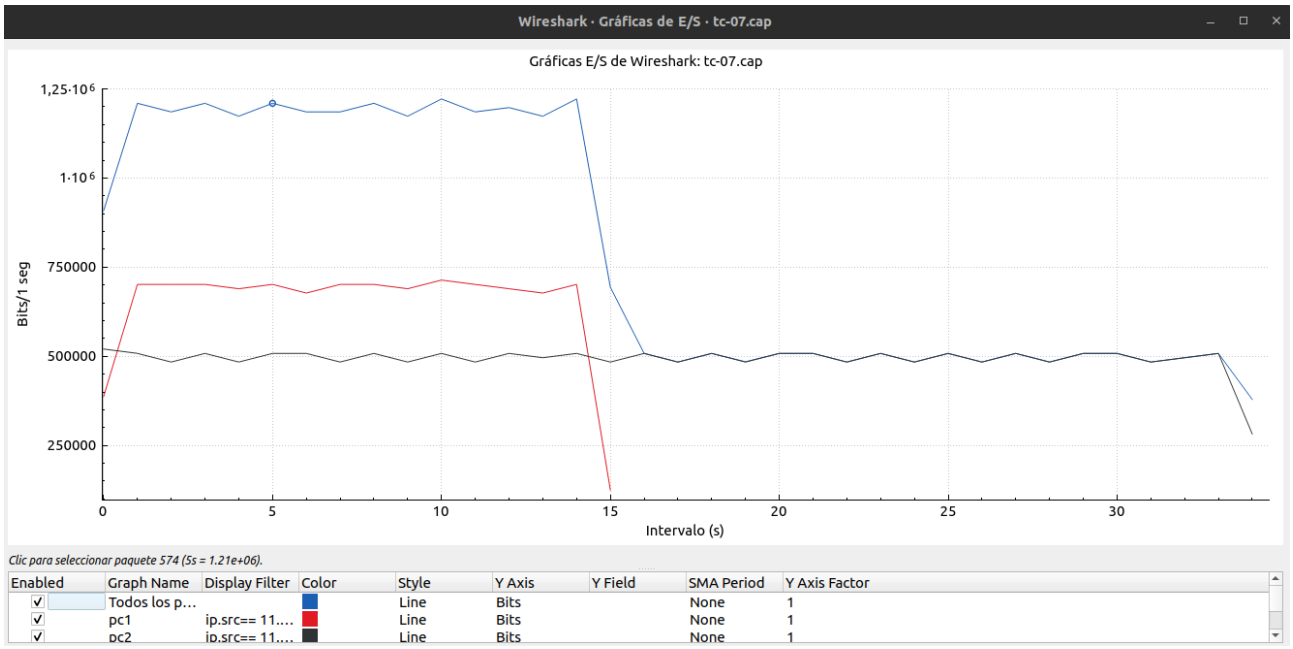
Durante un intervalo de *14.5* segundos, se transfirieron *1.17 MBytes* con un ancho de banda promedio de *680 Kbits/sec*. Además, se observó un jitter de *19.800 ms* y se perdió el *67*% de los datagramas recibidos, con un total de *1714* datagramas perdidos de un total de *2551.*

pc4:~# iperf -u -s

Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
---
[3] local 12.188.0.40 port 5001 connected with 11.188.0.20 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-34.3 sec 1.98 MBytes 484 Kbits/sec 11.026 ms 1138/ 2551 (45%)
read failed: Connection refused

Durante la prueba, se enviaron *2551* datagramas y se perdieron *1138* (*45*%). Durante el intervalo de tiempo de *18.1* segundos, se transfirieron *1.98 MBytes* de datos a un ancho de banda de *484 Kbits/sec*. El jitter fue de *11.026* ms. También se informó que se recibió un datagrama fuera de orden durante la prueba.

1.3.3.2



Según la gráfica, el ancho de banda del flujo de pc1 es de unos 700Kbits/seg, mientras que el flujo de pc2, es de unos 500Kbits/seg. En ambos flujos llegan paquetes fuera de los 10 segundos del envío de datos.
Cada flujo tiene esos anchos de banda debido a la configuración creada con HBT, empleada para el reparto del ancho de banda máximo entre diferentes redes.

1.3.3.3 *pc3:~# iperf -u -s*

Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)

---

[3] local 12.188.0.30 port 5001 connected with 11.188.0.10 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-14.5 sec 1.17 MBytes 678 Kbits/sec 20.802 ms 1712/ 2549 (67%)
[3] 0.0-14.5 sec 1 datagrams received out-of-order
read failed: Connection refused

Durante un intervalo de *14.5* segundos, se transfirieron *1.17 MBytes* con un ancho de banda promedio de *678 Kbits/sec*. Además, se observó un jitter de *20.802 ms* y se perdió el *67%* de los datagramas recibidos, con un total de *1712* datagramas perdidos de un total de *2549.* Además se recibió un datagrama fuera de orden.

pc4:~# iperf -u -s

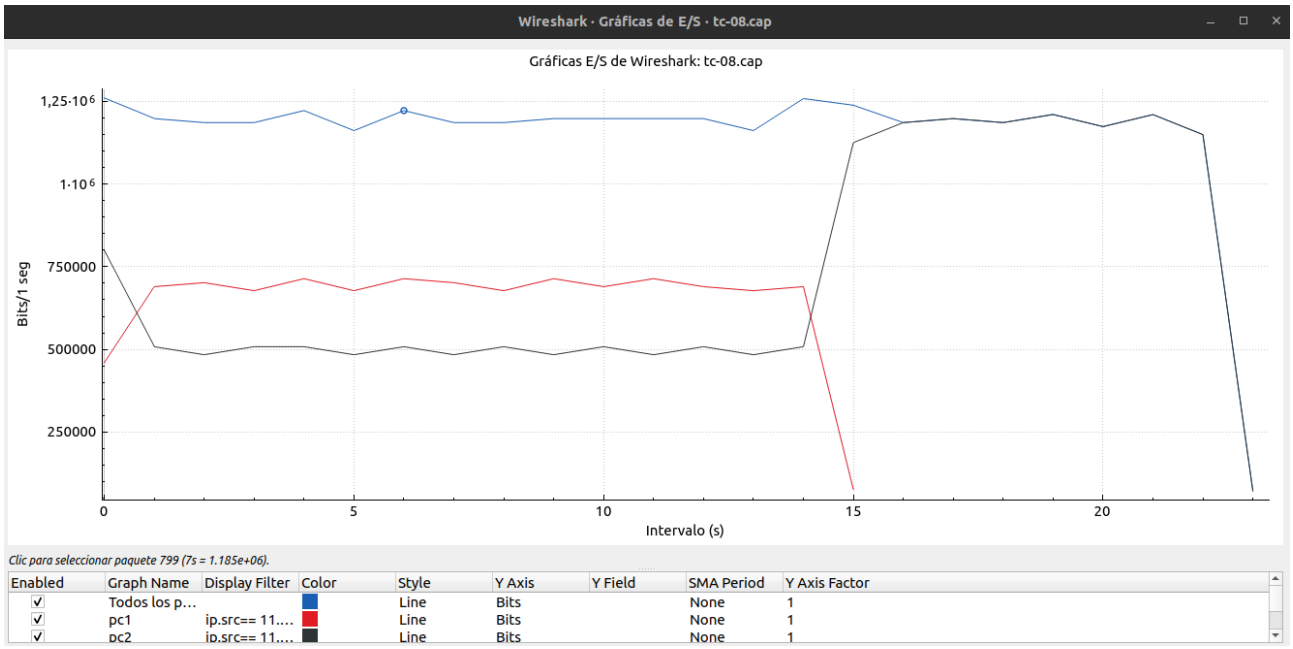
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)

---

[3] local 12.188.0.40 port 5001 connected with 11.188.0.20 port 32768
[ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[3] 0.0-23.1 sec 2.01 MBytes 730 Kbits/sec 24.904 ms 1115/ 2551 (44%)
read failed: Connection refused

Durante la prueba, se enviaron *2551* datagramas y se perdieron *1115* (*44%*). Durante el intervalo de tiempo de *23.1* segundos, se transfirieron *1.98 MBytes* de datos a un ancho de banda de *730 Kbits/sec*. El jitter fue de *24.904* ms.

1.3.3.4



Al igual que en la gráfica anterior, ambos flujos siguen el ancho de banda marcado por la configuración HBT, sin embargo, cuando el flujo de pc1 termina con el envío de datos fuera de tiempo, es cuando el flujo de pc2 aprovecha a usar los 1.2Mbit de ancho de banda máximo ya que posee un ceil mayor que en la anterior captura que le permite usar dicha magnitud.

2. Diffserv
2.1 Configuración de función policing y marcado de tráfico en DSCP

2.1.1 Script para la configuración del perfil de tráfico de r1:

```
#!/bin/sh

PC1_IP="11.188.0.10/32"
PC2_IP="11.188.0.20/32"

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, pasa al 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 600kbt y ráfaga 10k, marcado con calidad AF31. Si se supera este ancho de banda, el tráfico será descartao 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 calidad AF21. Si se supera este ancho de banda, pasa al 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 calidad AF11. Si se supera este ancho de banda, el tráfico será descartao 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

# Crear 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

Script para la configuración del perfil de tráfico de r2:
#!/bin/sh

PC3_IP="12.188.0.30/32"

tc qdisc add dev eth0 ingress handle ffff:

 # Flujo 5: máximo 400kbit con ráfaga 10k para el tráfico dirigido a pc6, marcado con calidad AF31. Si se supera este ancho de banda, pasa al 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 de 300kbit y ráfaga 10k para el tráfico dirigido a pc6, marcado con calidad AF21. Si se supera este ancho de banda, pasa al flujo 7.

*tc filter add dev eth0 parent ffff: *
*protocol ip prio 5 u32 *
*match ip src $PC3_IP *
police rate 600kbit burst 10k continue flowid :6

 # Flujo 7: máximo 100kbit con ráfaga 10k, marcado con calidad 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

 # Crear disciplina de cola de salida DSMARK

tc qdisc add dev eth1 root handle 1:0 dsmark indices 4

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

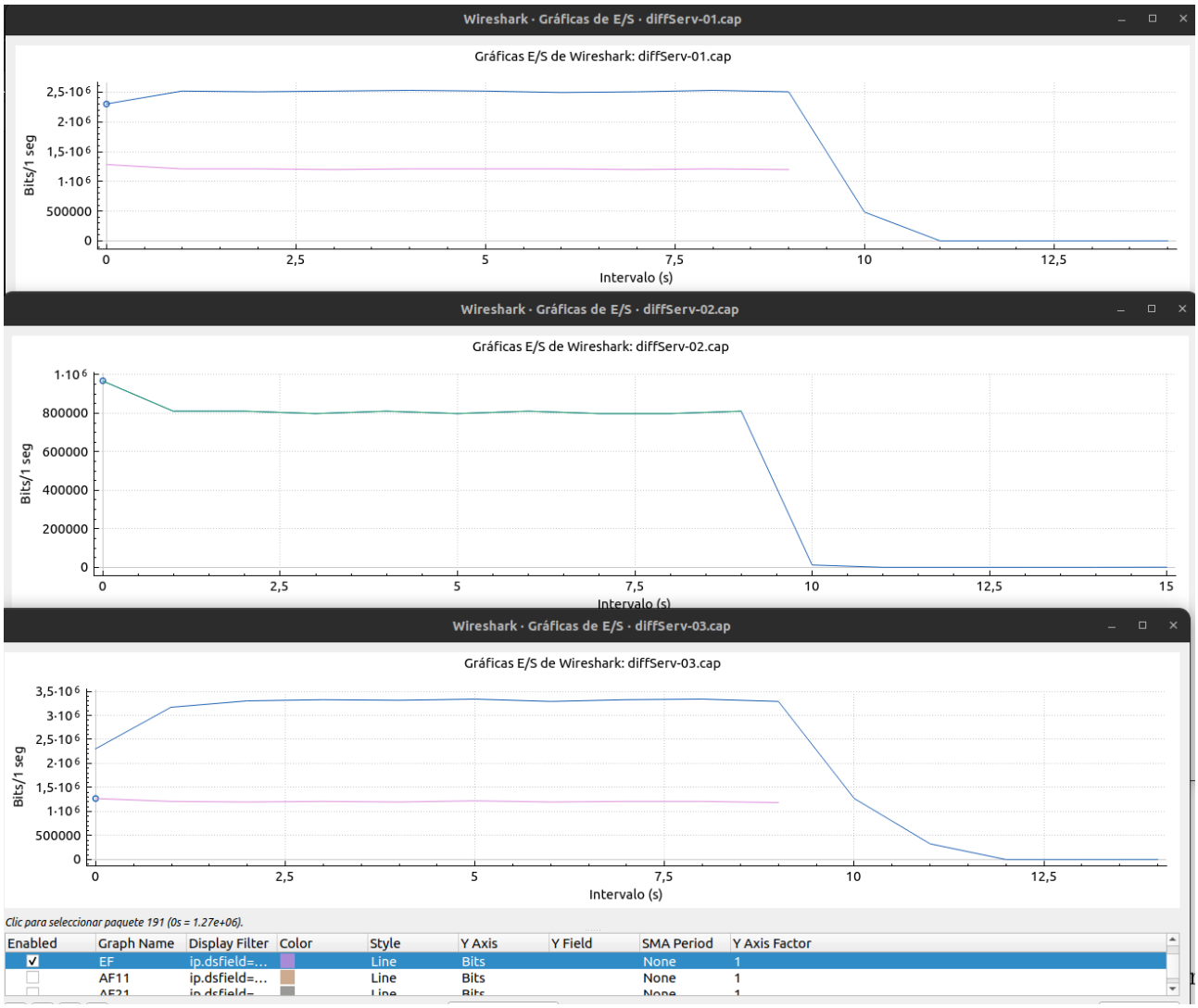
*tc filter add dev eth1 parent 1:0 protocol ip prio 1 *
handle 5 tcindex classid 1:5

*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

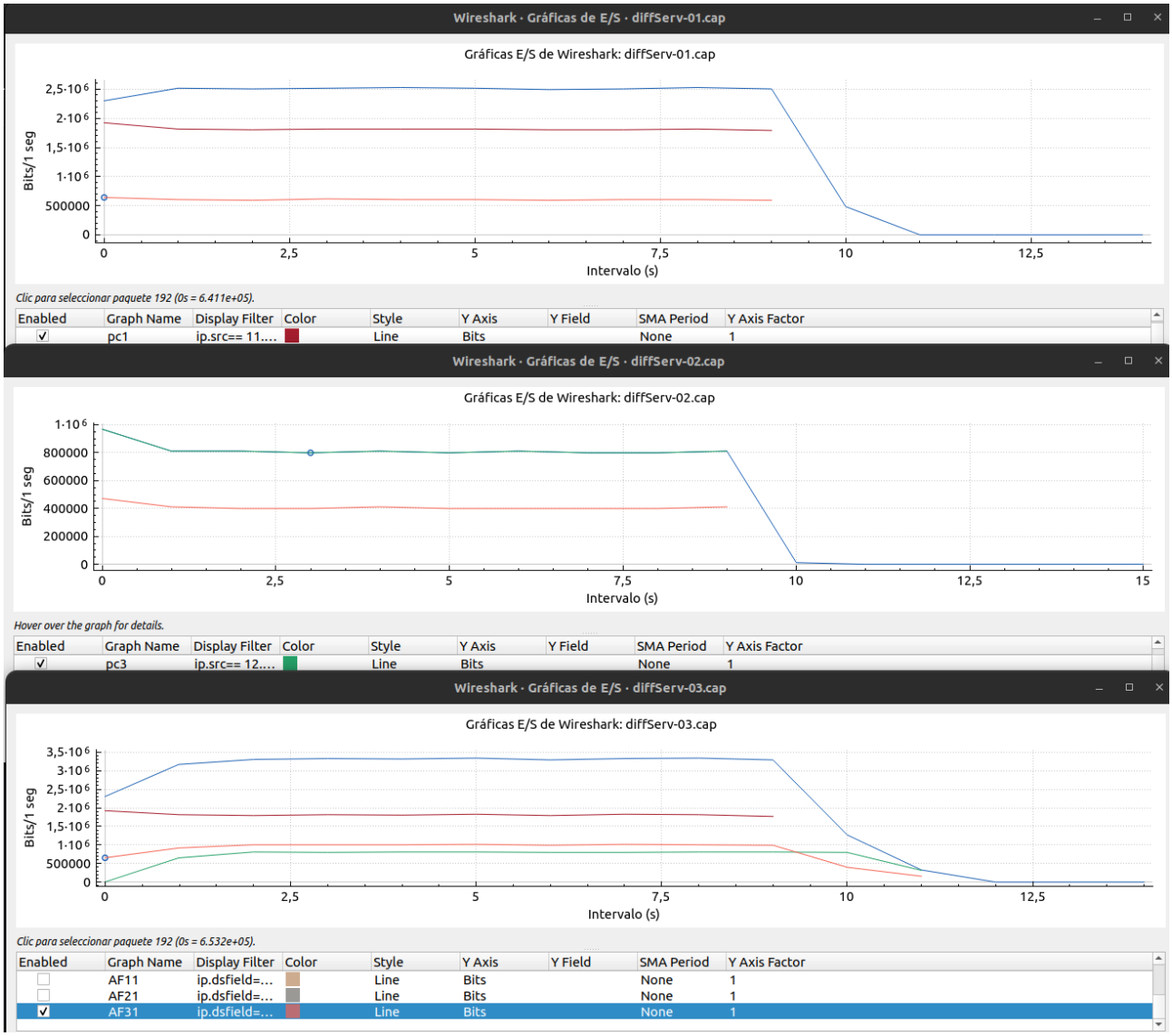
2.1.3

Tráfico de EF:



En este caso el tráfico de EF se puede observar en las capturas 1 y 3, ya que en la subred 12.188.0.0 no se marca el tráfico con la calidad de servicio EF. Dicho flujo sigue el límite de ancho de banda establecido de 1.2Mbits/seg, tal como especifica la red diffServ configurada.

Tráfico de AF31:

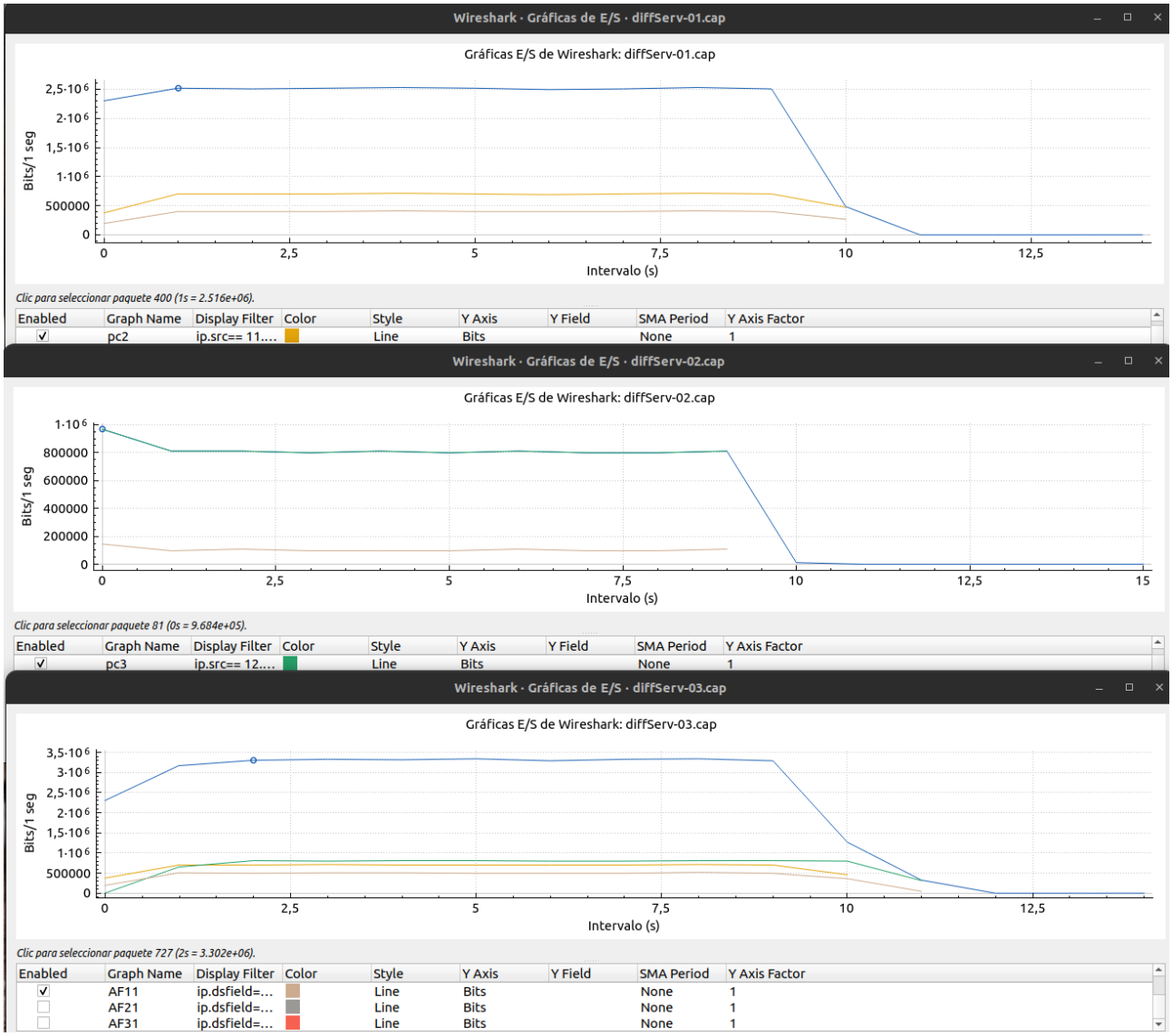


El flujo 2 y 5 se pueden observar en todas las capturas. En cada captura el flujo 2 y 5 siguen el ancho de banda establecido en la red diffServ. La suma de los anchos de banda máximo de cada flujo es el ancho de banda final del tráfico de AF31.

Tráfico de AF21:

El flujo 3 y 6 se pueden observar en todas las capturas. En cada captura el flujo 3 y 6 siguen el ancho de banda establecido en la red diffServ. La suma de los anchos de banda máximo de cada flujo es el ancho de banda final del tráfico de AF21.

Tráfico de AF11:



El flujo 4 y 7 se pueden observar en todas las capturas. En cada captura el flujo 4 y 7 siguen el ancho de banda establecido en la red diffServ. La suma de los anchos de banda máximo de cada flujo es el ancho de banda final del tráfico de AF11.

2.2. Tratamiento de tráfico en función del marcado DSCP

2.2.1 Script para la configuración del perfil de tráfico de r3:

```
#!/bin/sh

# Se crea la disciplina de cola ingress para rellenar el campo tc_index
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

tc class add dev eth2 parent 2:0 classid 2:1 htb rate 2.4Mbit
```

 # Interfaz ETH2
tc class add dev eth2 parent 2:1 classid 2:10 htb rate 1Mbit ceil 1Mbit # Flujo EF
tc class add dev eth2 parent 2:1 classid 2:11 htb rate 500kbit ceil 500kbit # Flujo AF31
tc class add dev eth2 parent 2:1 classid 2:12 htb rate 400kbit ceil 400kbit # Flujo AF21
tc class add dev eth2 parent 2:1 classid 2:13 htb rate 200kbit ceil 200kbit # Flujo AF11

 # ETH2 Trafico EF=> DS=0xb8 => DSCP=0x2e
*tc filter add dev eth2 parent 2:0 protocol ip prio 1 *
handle 0x2e tcindex classid 2:10

 # ETH2 Trafico AF31=> DS=0x68 => DSCP=0x1a
*tc filter add dev eth2 parent 2:0 protocol ip prio 1 *
handle 0x1a tcindex classid 2:11

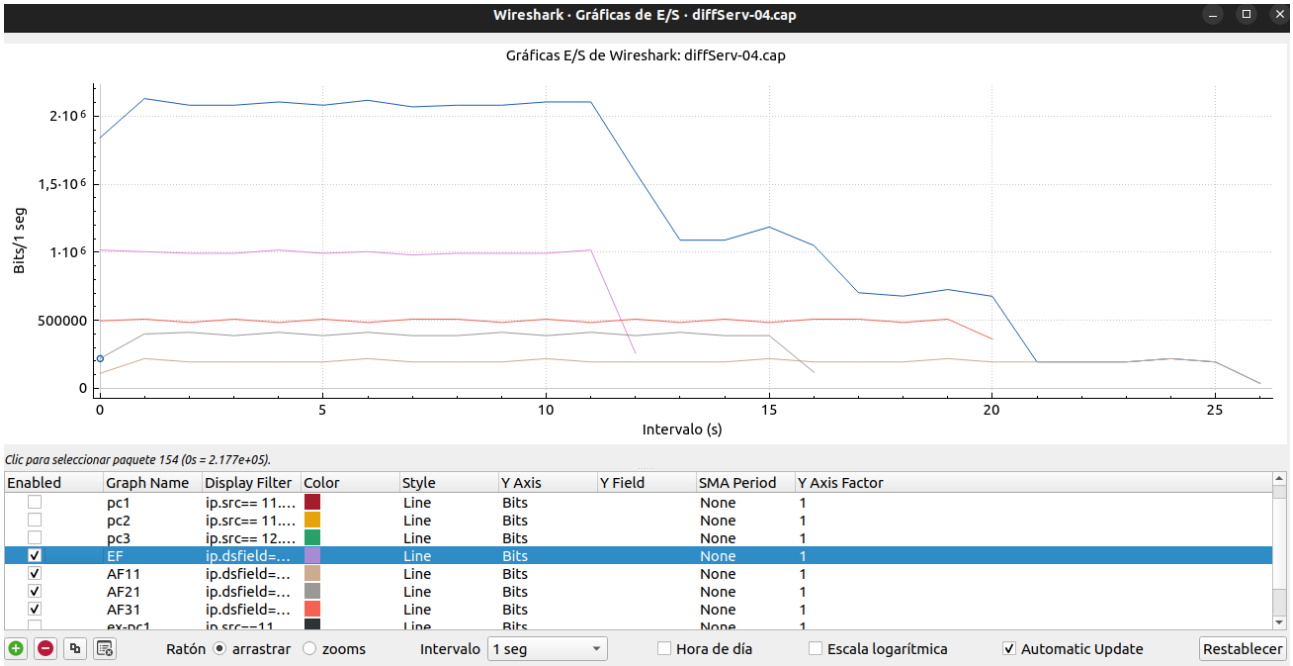
 # ETH2 Trafico AF21=> DS=0x48 => DSCP=0x12
*tc filter add dev eth2 parent 2:0 protocol ip prio 1 *
handle 0x12 tcindex classid 2:12

 # ETH2 Trafico AF11=> DS=0x28 =00101000 => DSCP=001010=0x0a
*tc filter add dev eth2 parent 2:0 protocol ip prio 1 *
handle 0x0a tcindex classid 2:13

Para lograr esto con Hierarchical Token Bucket (HTB), podemos hacer uso de la opción "quantum" en la configuración de cada clase.

Por defecto, el quantum se calcula como la tasa de datos de la clase dividido por el valor de "r2q". Esto significa que cada clase obtendrá su parte justa del ancho de banda definido. Sin embargo, podemos aumentar el valor del quantum para permitir que una clase use más ancho de banda si no se está utilizando por otras clases.

2.2.3



Tal como se puede observar en la gráfica, el resultado obtenido corresponde con la configuración realizada en r3 para las distintas calidades de servicio.
Por otro lado, todos los flujos han terminado encolando tráfico después de los 10 segundos. El flujo de EF es el que menos tráfico encola, mientras que el flujo que más tráfico ha encolado es el flujo de AF11.

2.2.4 Script con las modificaciones para la configuración del perfil de tráfico de r3:

```
#!/bin/sh

# Se crea la disciplina de cola ingress para rellenar el campo tc_index
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

tc class add dev eth2 parent 2:0 classid 2:1 htb rate 2.4Mbit

# Interfaz ETH2
tc class add dev eth2 parent 2:1 classid 2:10 htb rate 1Mbit ceil 2.4Mbit # Flujo EF
tc class add dev eth2 parent 2:1 classid 2:11 htb rate 500kbit ceil 2.4Mbit # Flujo AF31
tc class add dev eth2 parent 2:1 classid 2:12 htb rate 400kbit ceil 2.4Mbit # Flujo AF21
tc class add dev eth2 parent 2:1 classid 2:13 htb rate 200kbit ceil 2.4Mbit # Flujo AF11

# ETH2 Trafico EF=> DS=0xb8 => DSCP=0x2e
tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
handle 0x2e tcindex classid 2:10

# ETH2 Trafico AF31=> DS=0x68 => DSCP=0x1a
tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
handle 0x1a tcindex classid 2:11

# ETH2 Trafico AF21=> DS=0x48 => DSCP=0x12
tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
handle 0x12 tcindex classid 2:12

# ETH2 Trafico AF11=> DS=0x28 =00101000 => DSCP=001010=0x0a
tc filter add dev eth2 parent 2:0 protocol ip prio 1 \
handle 0x0a tcindex classid 2:13
```

La modificación realizada en el script es el aumento del ceil en cada una de las clases a 2.4Mbit, de esta manera, si alguna de los flujos no emplea al máximo su ancho de banda disponible, otro flujo puede ser capaz de aumentar su ancho de banda ya que su ceil se lo permite.

2.2.5 Tal como se puede apreciar en la gráfica, observamos que el tráfico que llega después de los 10 segundos es mucho menor que en la gráfica de la captura anterior. Esto se debe a que una vez los flujos dejen de usar el ancho de banda que se les ha asignado, otro flujo puede emplear ese ancho de banda y aumentar el suyo, consiguiendo que el tiempo de llegada del tráfico de después de los 10 segundos sea menor.