Control de Tráfico en Linux

Redes de Ordenadores para Robots y Máquinas Inteligentes

GSyC
Departamento de Teoría de la Señal y
Comunicaciones y Sistemas Telemáticos y Computación

Marzo de 2024

Contenidos

Contenidos

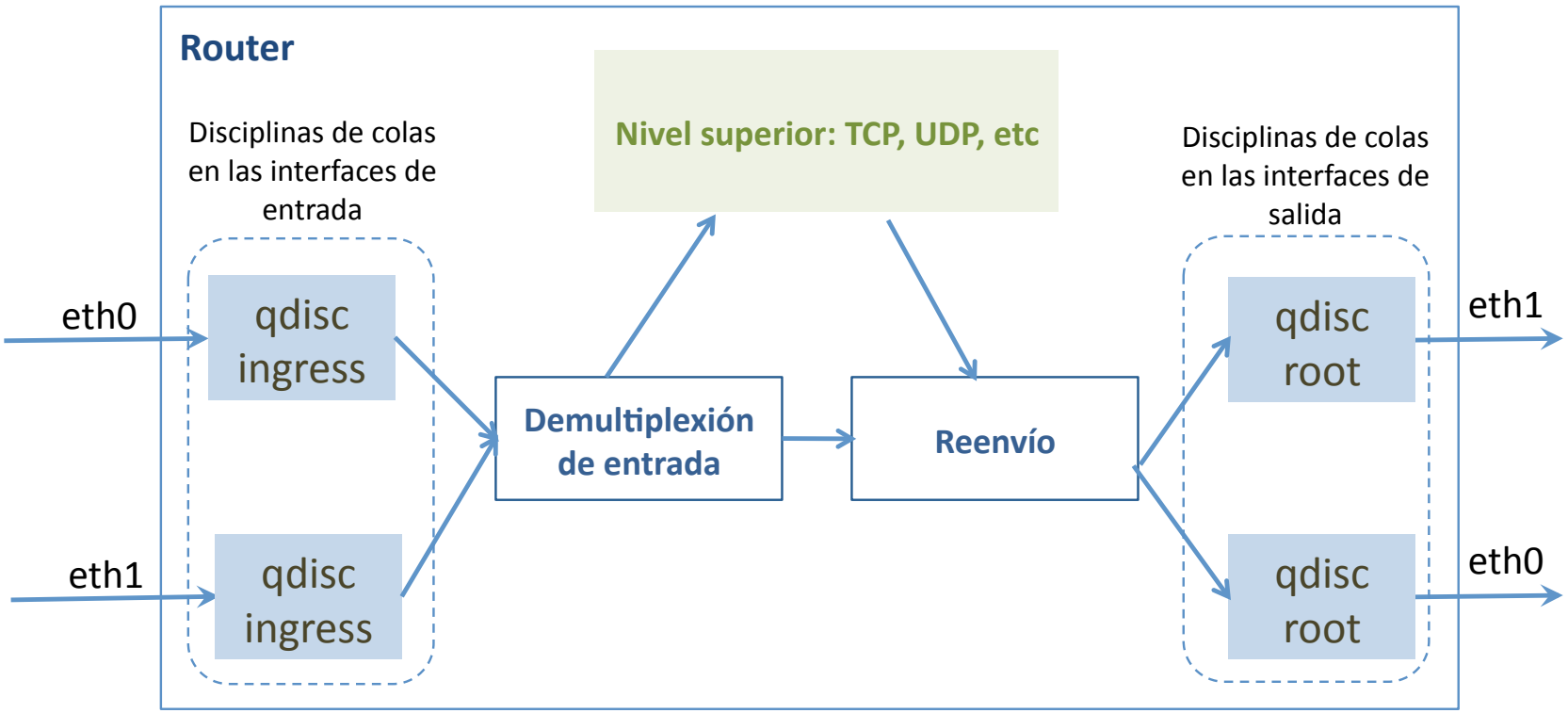
Control de tráfico en Linux

- tc (*traffic control*) es una herramienta que permite realizar el control de tráfico en Linux.
- tc utiliza disciplinas de colas, denominadas `qdisc` (*queueing discipline*).
- Cada vez que el kernel tiene que enviar un paquete a una interfaz, se encola en la `qdisc` configurada en dicha interfaz. El kernel trata de sacar el máximo número de paquetes de dicha `qdisc` para entregárselos al driver de la tarjeta de red.
- El tipo de `qdisc` más simple es FIFO (First In First Out). Si el driver de la tarjeta de red está ocupado, el paquete se quedará guardado en la cola.

Contenidos

Interfaces de entrada/salida de un router

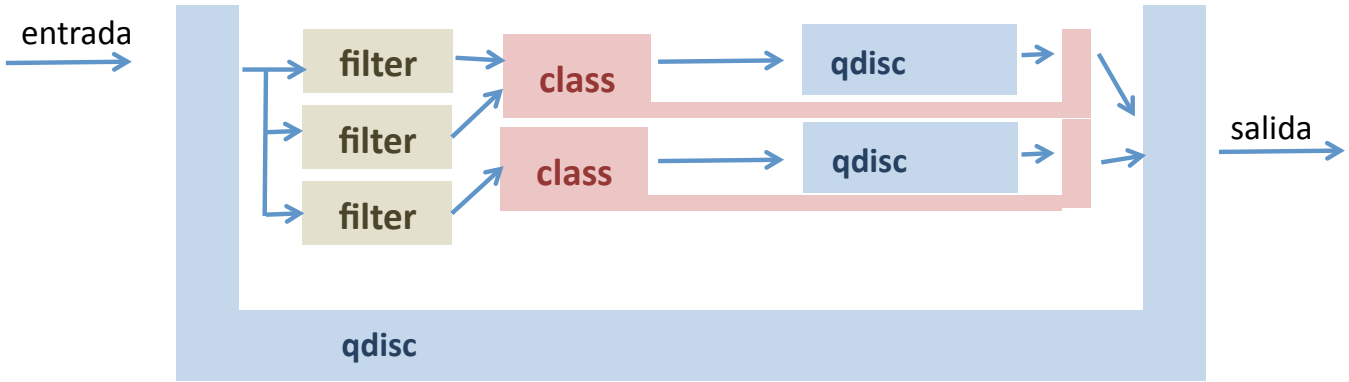
- El control de tráfico se puede aplicar en la interfaz de entrada de un router o en la interfaz de salida del router.
- Cada interfaz de red de un router puede actuar como entrada de paquetes, para los paquetes que se reciben en dicha interfaz, y como interfaz de salida, para los paquetes que se envían a través de dicha interfaz.



Elementos del control de tráfico: Qdisc/Class/Filter

- Qdisc (Disciplina de cola): determina qué paquetes se reenviarán antes que otros. Las hay de 2 tipos:
  - Disciplina de colas **sin clases de tráfico**: ejemplo FIFO.

  - Disciplina de colas **con clases de tráfico**: ejemplo PRIO
    - Class (Clase de tráfico): especifica las diferentes categorías de tráfico. Una clase de tráfico está asociada a una disciplina de cola (qdisc).
    - Filter (Filtro de tráfico): identifica el tráfico que se corresponde con cada clase (class).



Configuración de una disciplina de cola a la entrada/salida de un router

- En una determinada interfaz se pueden definir:
  - **Disciplinas de cola de entrada**: para los paquetes que entran a través de dicha interfaz.

```
tc qdisc add dev <interfaz> ingress ...
```

  - **Disciplinas de cola de salida**: para los paquetes que salen a través de dicha interfaz.

```
tc qdisc add dev <interfaz> root ...
```

- Por defecto, si no se modifican las disciplinas de cola a la entrada o a la salida de un router, el router aplicará una variante de FIFO (`pfifo_fast`, FIFO mejorado).

Descriptor (*handle*) de una disciplina de cola

- Cuando se define una disciplina de cola se le asocia un descriptor (*handle*) para poder referenciarla.
- El *handle* está formado por 2 números separados por el caracter ":", es decir, `X:Y`
  - `X` : es el número mayor.
  - `Y` : es el número menor.

  Donde `X` identifica la disciplina de cola e `Y` hace referencia a un elemento que depende de esa disciplina de cola (flujos, clases).
- El *handle* de la disciplina de la cola de entrada (*ingress*) es `ffff:` lo que equivale a `ffff:0`.

```
tc qdisc add dev <interfaz> ingress handle ffff:
```

- Típicamente el *handle* de la disciplina de la cola de salida (*root*) es `1:` lo que equivale a `1:0`.

```
tc qdisc add dev <interfaz> root handle 1: ...
```

Borrado de una disciplina de cola

- El borrado de disciplinas de cola en la entrada y en la salida de un router se hace de la siguiente forma:
- Borrado de la disciplina de la cola de entrada:

```
tc qdisc del dev <interfaz> ingress
```

- Borrado de la disciplina de la cola de salida:

```
tc qdisc del dev <interfaz> root
```

Contenidos

Control de admisión para el tráfico de entrada (*policing*)

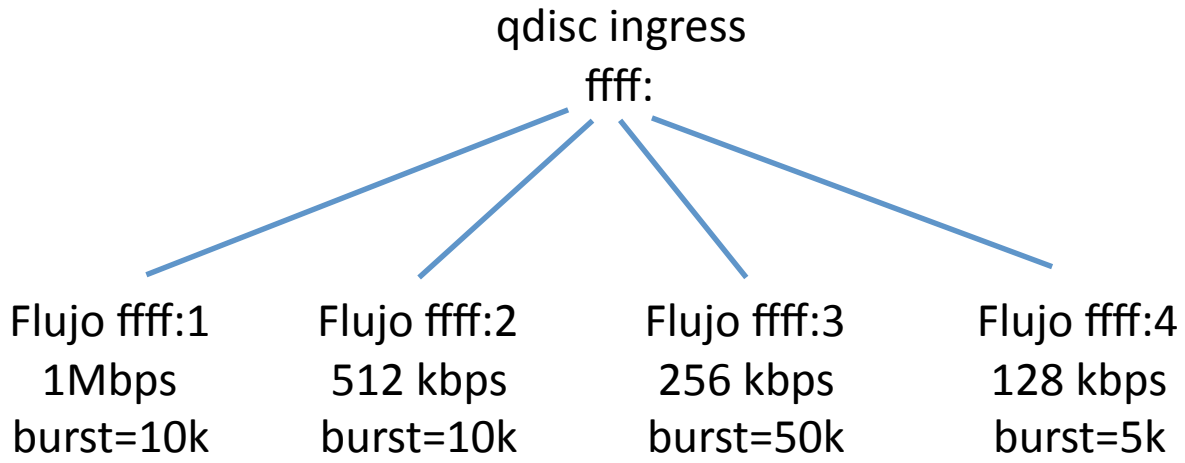
- El control de admisión para el tráfico de entrada se utiliza para garantizar que el router sólo procesa para un determinado flujo de paquetes de entrada un máximo de ancho de banda.
- Para garantizar que se cumple el control de admisión el router debe definir una disciplina de colas en la interfaz de red asociada a la entrada de paquetes. Ejemplo, si los paquetes se reciben en la interfaz `eth0`:

```
tc qdisc add dev eth0 ingress handle ffff:
```

- Se definen filtros para cada uno de los flujos de entrada que van a pertenecer a esa cola de entrada en la interfaz `eth0` (*handle* ffff:) indicando el ancho de banda máximo y el tamaño de la cubeta. El control de admisión se basa en el modelo TBF. Si el tráfico de entrada supera esa especificación se descarta o se le pasa al siguiente filtro.

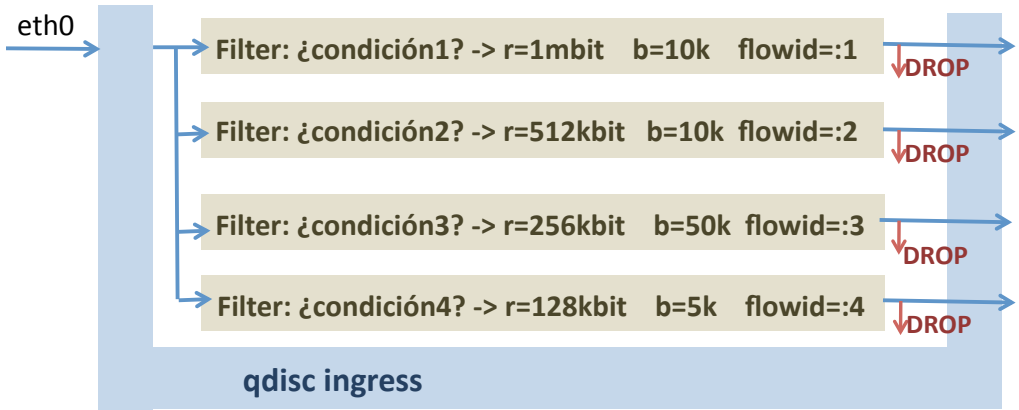
Ejemplo: control de admisión en el tráfico de entrada (I)

- Supongamos que se quiere separar el tráfico que se ha recibido en una determinada interfaz, en varios flujos.
- En la disciplina de cola de entrada (ffff:) queda almacenado todo el tráfico recibido en dicha interfaz.
- Ahora es necesario aplicar filtros a ese tráfico para separarlo en *i* flujos (cada flujo queda identificado por `ffff:i`) a los que asignaremos el ancho de banda que deseemos.



Ejemplo: control de admisión en el tráfico de entrada (II)

- Se define qdisc para la entrada: ingress
- Se definen filtros para clasificar el tráfico: los paquetes que cumplen una condición se les aplica un TBF y quedan clasificados dentro de un determinado flujo (*flowid*):
  - **Condición**: utilizaremos el filtro u32 que comprueba campos de la cabecera IP de los paquetes.
  - **Perfil de tráfico**: se configura con *rate* y *burst*.
  - **Flujo**: el tráfico que esté dentro de la especificación queda clasificado en un `flowid`.
  - **Acción**: el tráfico que supere la especificación puede reclasificarse pasando al siguiente filtro (*continue*) o descartarse (*drop*).
- Los filtros se aplican siguiendo el orden dado por el parámetro prioridad: primero los filtros cuyo valor de prioridad sea menor (números de prioridad bajos =>mayor prioridad).
- Si un paquete cumple la condición del filtro, se le aplica su clasificación o su acción asociada. Si no, se pasa al siguiente filtro por orden de prioridad.



Ejemplo: control de admisión en el tráfico de entrada (III)

- Ejemplo:

```
tc qdisc add dev eth0 ingress handle ffff:

tc filter add dev eth0 parent ffff: \
    protocol ip prio 4 u32 \
    match ip src 11.0.0.100/32 \
    police rate 1mbit burst 10k continue flowid :1

tc filter add dev eth0 parent ffff: \
    protocol ip prio 5 u32 \
    match ip src 11.0.0.100/32 \
    police rate 512kbit burst 10k continue flowid :2

tc filter add dev eth0 parent ffff: \
    protocol ip prio 6 u32 \
    match ip src 11.0.0.100/32 \
    police rate 256kbit burst 50k drop flowid :3

tc filter add dev eth0 parent ffff: \
    protocol ip prio 6 u32 \
    match ip src 0.0.0.0/0 \
    police rate 128kbit burst 5k  drop flowid :4
```

Contenidos

Contenidos

PFIFO para el tráfico de salida

- FIFO con tamaño de cola dado por el valor `limit` en número de paquetes:

```
tc qdisc add dev eth0 root pfifo limit 5
```

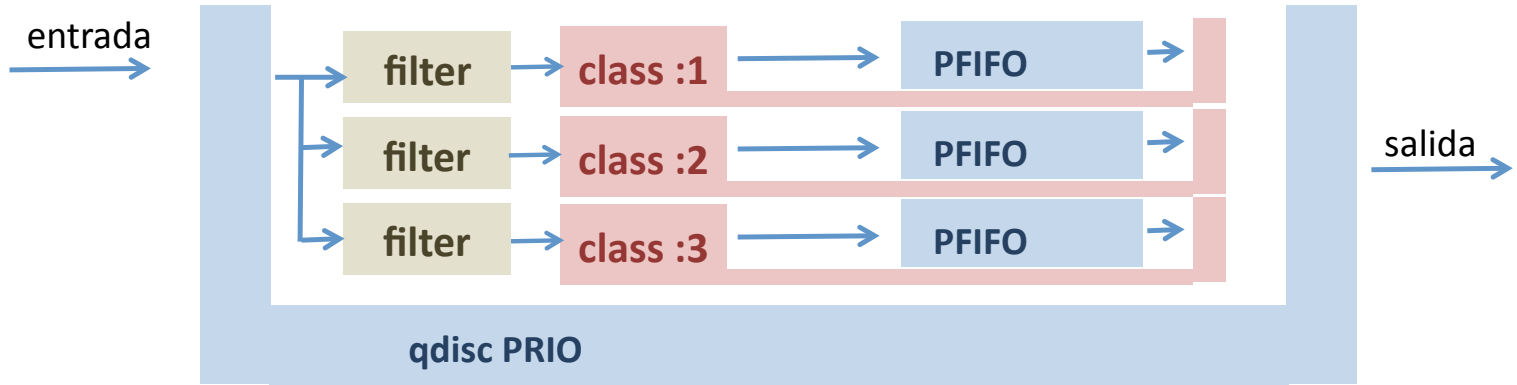
- Es una disciplina de colas muy sencilla que requiere poco tiempo de computación.
- Si hay congestión, FIFO perjudica a TCP debido a que una pérdida de un paquete en TCP activa los mecanismos de control de congestión que provocan una disminución de la tasa de envío para la conexión TCP. Una pérdida de un paquete UDP no tiene ningún impacto en la tasa de envío para esta comunicación.
- Una ráfaga de uno de los flujos que atraviese una cola FIFO puede llenar la cola y perjudicar al resto de los flujos.

Contenidos

PRIO para el tráfico de salida (I)

- PRIO es una disciplina de cola de prioridades que realiza algunas acciones automáticamente. El siguiente comando crea la disciplina de cola 1:0 y automáticamente crea 3 clases 1:1, 1:2 y 1:3 de tipo PFIFO.

```
tc qdisc add dev eth0 root handle 1: prio
```



- Si se desea crear un número diferente de prioridades se puede usar el parámetro: `bands <núm_prio>`.

PRIO para el tráfico de salida (II)

- Es necesario especificar los filtros para clasificar el tráfico dentro de las clases 1:1, 1:2 y 1:3 del ejemplo. Esta clasificación puede hacerse a través del filtro u32 utilizando la dirección IP origen:

```
tc filter add dev eth0 parent 1:0 prio 1 protocol ip u32 \
          match ip src 11.0.0.1/32 flowid 1:1
tc filter add dev eth0 parent 1:0 prio 2 protocol ip u32 \
          match ip src 11.0.0.2/32 flowid 1:2
tc filter add dev eth0 parent 1:0 prio 3 protocol ip u32 \
          match ip src 11.0.0.3/32 flowid 1:3
```

- La disciplina de colas de prioridad puede provocar retrasos y pérdidas de paquetes en la clase menos prioritaria 1:3 si hay mucho tráfico en 1:1 y 1:2.
- Para solventar que TCP se vea perjudicado con PFIFO, se podría clasificar tráfico TCP en 1:1 y el UDP en 1:2. Sin embargo, si hubiera muchos datos para TCP, los mecanismos de control de congestión de TCP provocarían que la tasa de envío fuera aumentando cada vez más mientras no se perdieran paquetes, viéndose en este caso UDP perjudicado ya que mientras hubiera tráfico en 1:1 no se cursaría el resto de tráfico.

Contenidos

Token Bucket Filter (TBF) para el tráfico de salida

- TBF limita la velocidad del tráfico de salida de una determinada interfaz de un router a través de los parámetros:
  - ancho de banda: velocidad de generación tokens
  - tamaño de la cubeta: para almacenar tokens que permitirán enviar ráfagas que superen el ancho de banda.
  - latencia: tiempo máximo de espera de un paquete por un token.
- Ejemplo: Aplicar TBF en la interfaz de salida eth1 de un router. Ancho de banda 7Mbps , tamaño de cubeta 10k (en bytes), latencia 50 ms:

```
tc qdisc add dev eth1 root handle 1: tbf  rate 7Mbit \
            burst 10k latency 50ms
```

Contenidos

TBF y PRIO

- TBF limita la velocidad del tráfico de salida de una determinada interfaz para todos los flujos que lo atraviesan.
- Se puede definir una disciplina de cola PRIO hija de TBF para que el tráfico además de estar limitado por TBF los paquetes se cursen atendiendo a las prioridades definidas en PRIO:

```
tc qdisc add dev eth1 root handle 1: tbf rate 7Mbit \
            burst 10k latency 50 ms

tc qdisc add dev eth1 parent 1:0 handle 10:0 prio

tc filter add dev eth1 parent 10:0 prio 1 protocol ip u32 \
     match ip src 101.0.0.10/32 flowid 10:1
tc filter add dev eth1 parent 10:0 prio 2 protocol ip u32 \
     match ip src 102.0.0.20/32 flowid 10:2
tc filter add dev eth1 parent 10:0 prio 3 protocol ip u32 \
     match ip src 103.0.0.30/32 flowid 10:3
```

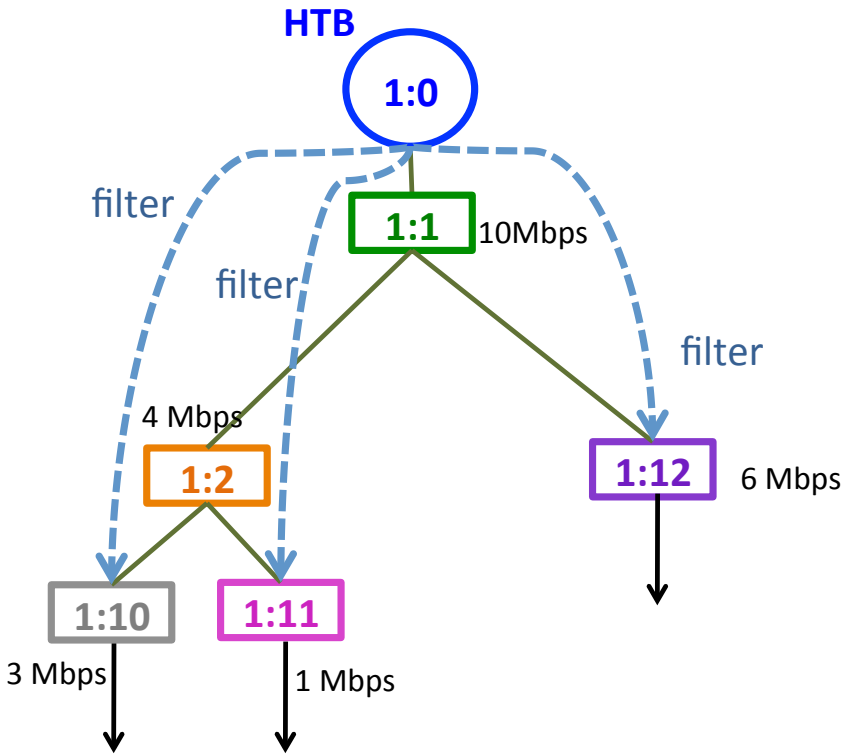
Contenidos

Hierarchical Token Bucket (HTB) para el tráfico de salida

- TBF se utiliza cuando se desea que todo el tráfico que sale por una determinada interfaz tenga unas características determinadas en cuanto a ancho de banda, tamaño de cubeta y latencia.
- HTB es necesario cuando se quiere clasificar el tráfico que sale por una determinada interfaz en varias clases, cada uno de ellas con unas características diferentes de ancho de banda.
- Si alguna clase no utiliza todo el ancho de banda que se le ha definido, dependiendo de la configuración, se podría utilizar dicho ancho de banda para algún otra clase que lo necesite.

Ejemplo de Hierarchical Token Bucket (HTB) para el tráfico de salida

- Definición de HTB con ancho de banda máximo de salida de 10Mbps de la interfaz eth0. Se quiere repartir este ancho de banda:
  - usuarioA (4Mbps): lo usará para tráfico HTTP (3Mbps) y tráfico de correo (1Mbps)
  - el usuarioB (6Mbps)



Ejemplo de Hierarchical Token Bucket (HTB) para el tráfico de salida

```
tc qdisc add dev eth0 root handle 1:0 htb

tc class add dev eth0 parent 1:0 classid 1:1 htb rate 10Mbit

tc class add dev eth0 parent 1:1 classid 1:2 htb rate 4Mbit ceil 10Mbit
tc class add dev eth0 parent 1:2 classid 1:10 htb rate 3Mbit ceil 10Mbit
tc class add dev eth0 parent 1:2 classid 1:11 htb rate 1Mbit ceil 10Mbit
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 6Mbit ceil 10Mbit

tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
  match ip src 1.1.1.1 \
  match ip dport 80 0xffff \
  flowid 1:10

tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
  match ip src 1.1.1.1 \
  match ip dport 25 0xffff \
  flowid 1:11

tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
  match ip src 2.2.2.2 \
  flowid 1:12
```

Unidades para tc

- Ancho de banda

kbps	Kilobytes per second
mbps	Megabytes per second
kbit	Kilobits per second
mbit	Megabits per second
bps	Bytes per second

- Cantidad de datos

kb o k	Kilobytes
mb o m	Megabytes
kbit	Kilobits
mbit	Megabits
b	Bytes

- Tiempo

s, sec o secs	Segundos
ms, msec o msecs	Milisegundos
us, usec, usecs	Microsegundos

Contenidos

Iperf: Generador de tráfico

- Iperf es una aplicación que permite generar tráfico TCP o UDP y medir el ancho de banda que se está utilizando entre dos máquinas finales.
- Iperf funciona en modo cliente/servidor.
- Utilizaremos Iperf para generar tráfico UDP desde el cliente al servidor durante 10 segundos a una determinada velocidad y medir el ancho de banda que en realidad se ha utilizado.
- Arrancaremos Iperf de la siguiente forma:
  - Servidor

```
iperf -u -s
```

  - Cliente

```
iperf -u -c <dirIPServidor> -b <anchoDeBanda>
```

    Donde `<anchoDeBanda>` es un número `n` en bps. Se puede usar `nK` (para enviar n kilobit por segundo) o `nM` (para enviar n megabit por segundo).

Resultado de la ejecución de Iperf

- Resultado de ejecución de Iperf en 12.0.0.30, en modo servidor:

```
iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
------------------------------------------------------------
[3] local 12.0.0.30 port 5001 connected with 11.0.0.10 port 32768
[3] 0.0- 9.7 sec 25.2 MBytes 21.8 Mbits/sec 0.289 ms 4940/22916 (22%)
[3] 0.0- 9.7 sec 1 datagrams received out-of-order
```

- Resultado de ejecución de Iperf en 11.0.0.10, en modo cliente:

```
iperf -u -c 12.0.0.30 -b 100M
------------------------------------------------------------
Client connecting to 12.0.0.30, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 108 KByte (default)
------------------------------------------------------------
[3] local 11.0.0.10 port 32768 connected with 12.0.0.30 port 5001
[3] 0.0- 10.0 sec 32.1 MBytes 26.9 Mbits/sec
[3] Sent 22917 datagrams
[3] Server Report:
[3] 0.0- 9.7 sec 25.2 MBytes 21.8 Mbits/sec 0.289 ms 4940/22916 (22%)
[3] 0.0- 9.7 sec 1 datagrams received out-of-order
```

Contenidos

Gráficas de ancho de banda (I)

- Seleccionar en el menú de Wireshark `Statistics` $->$ `IO Graphs`.
- En el panel inferior, añadir filtros según los criterios que se deseen. Marcar con un tic a la izquierda del filtro para visualizar los paquetes filtrados en la gráfica.
- Para distinguir el tráfico de varias fuentes la condición de filtrado en `Display filter` puede incluir la comprobación de la dirección IP y puerto TCP/UDP. Ejemplo: `ip.src==11.0.0.1 and udp.dstport==5001`
- Si no se escribe nada en la condición de filtrado, se muestra la gráfica de todo el tráfico incluido en la captura.

Gráficas de ancho de banda (II)

- Es importante seleccionar adecuadamente las unidades que muestra la gráfica. Para las pruebas que vamos a realizar en las prácticas se recomienda:
  - Color: Uno diferente para cada filtro
  - Style: Line
  - Y Axis: Bits
  - Interval (común para todos los filtros): 1 sec