EL COMANDO tc

Autores:
Sergio de la Rubia García-Carpintero
Alicia Martín-Benito Escalona

Índice

- Introducción
- Qué es *tc*
  - Elementos del control de tráfico
  - Objetos que intervienen en el control del tráfico
- Trabajando con *tc*

Introducción

- Enlaces y conexión muy veloces.
- **Multipropósito**
  - Internet, correo corporativo, videoconferencia, canales de voz, etc.
- Difícil alcanzar las velocidades de transmisión supuestas para esa red.
  - **Colisiones** que presenta *ethernet*.
- Definir:
  - Cuáles son los servicios fundamentales.
  - Cuánto ancho de banda requieren.

Qué es *tc* (1/11)

DEFINICIÓN:

- Es un **comando** de los sistemas *Linux* que se utiliza para **mostrar y manipular la configuración del tráfico de una red**.

Qué es *tc* (2/11)

ELEMENTOS DEL CONTROL DE TRÁFICO:

- ***SHAPING*** (modelado)
  - **Mecanismo por el cual la emisión de paquetes se retrasa,** colocándolos en una cola de salida, **para cumplir con la tasa de salida deseada**.
  - Reducir el ancho de banda disponible.
  - Suavizar los picos de tráfico.

Qué es *tc* (3/11)

ELEMENTOS DEL CONTROL DE TRÁFICO: (cont.)

- ***SCHEDULING*** (planificación)
  - **Mecanismo por el cual los paquetes se ordenan (o reordenan) entre la entrada y la salida de una cola determinada.**
  - Posibilita mejorar la interactividad para el tráfico que lo necesita *(**priorización**)*.
  - Sigue garantizando el ancho de banda para las otras transferencias.

Qué es *tc* (4/11)

ELEMENTOS DEL CONTROL DE TRÁFICO: (cont.)

- ***POLICING*** (políticas)
  - **Mecanismo que mide y limita el tráfico de una cola de entrada particular.**
  - Si el paquete que va a entrar no supera el límite, se permite el encolado.
  - En otro caso, se realiza otra operación.

Qué es *tc* (5/11)

ELEMENTOS DEL CONTROL DE TRÁFICO: (cont.)

- ***DROPPING*** (reducción)
  - **Mecanismo por el cual se descartan todos los paquetes de un tipo determinado.**
  - Recomendable si el tráfico excede un ancho de banda determinado.
  - Aplicable tanto para el tráfico entrante como para el saliente.

Qué es *tc* (6/11)

OBJETOS QUE INTERVIENEN:

- ***QDISCS*** (disciplina de cola)
  - El kernel **necesita enviar un paquete por una interfaz.**
  - Lo **encola en la *qdisc* configurada para esa interfaz.**
  - El kernel trata de **obtener de la *qdisc* tantos paquetes como le sea posible**.
  - **Los manda al driver** del adaptador de red.

Qué es *tc* (7/11)

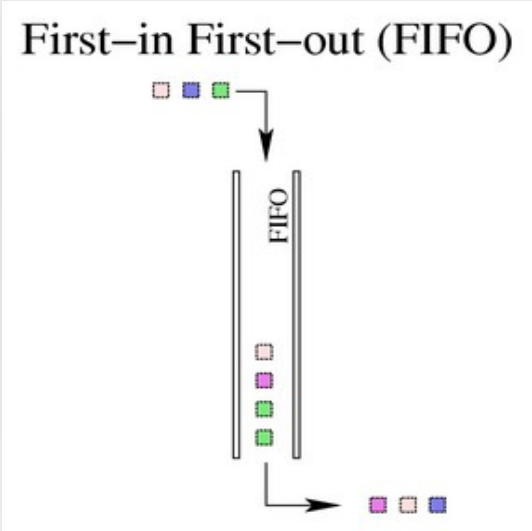
OBJETOS QUE INTERVIENEN: (cont.)

- **CLASES**
  - Algunas *qdisc* pueden contener clases, que **contienen a su vez más *qdisc*.**
  - El tráfico puede ser encolado en alguna de esas *qdisc* internas.
- **FILTROS**
  - Es utilizado por una *qdisc* para **determinar en qué clase será encolado un paquete.**

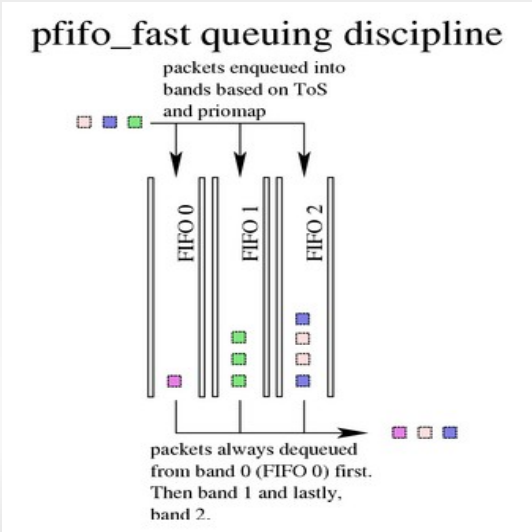
Qué es *tc* (8/11)

QDISCS SIN CLASES:

- **fifo**
  - Son las *qdisc* más simples.
  - Comportamiento FIFO puro.



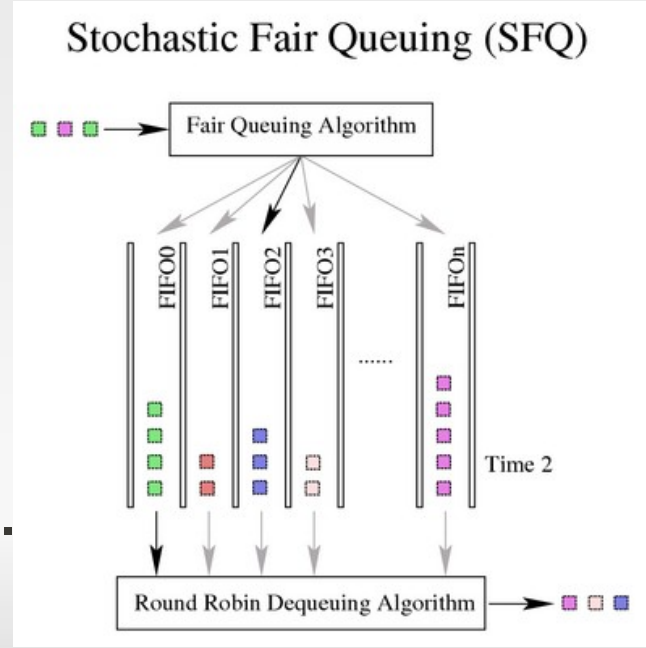
- **pfifo_fast**
  - *Qdisc* por defecto.
  - Cola de tres bandas (FIFOs)
  - Encolado: según prioridad en 0, 1 ó 2
  - Desencolado: primero 0, luego 1 y finalmente 2.



Qué es *tc* (9/11)

***QDISCS* SIN CLASES:** (cont.)

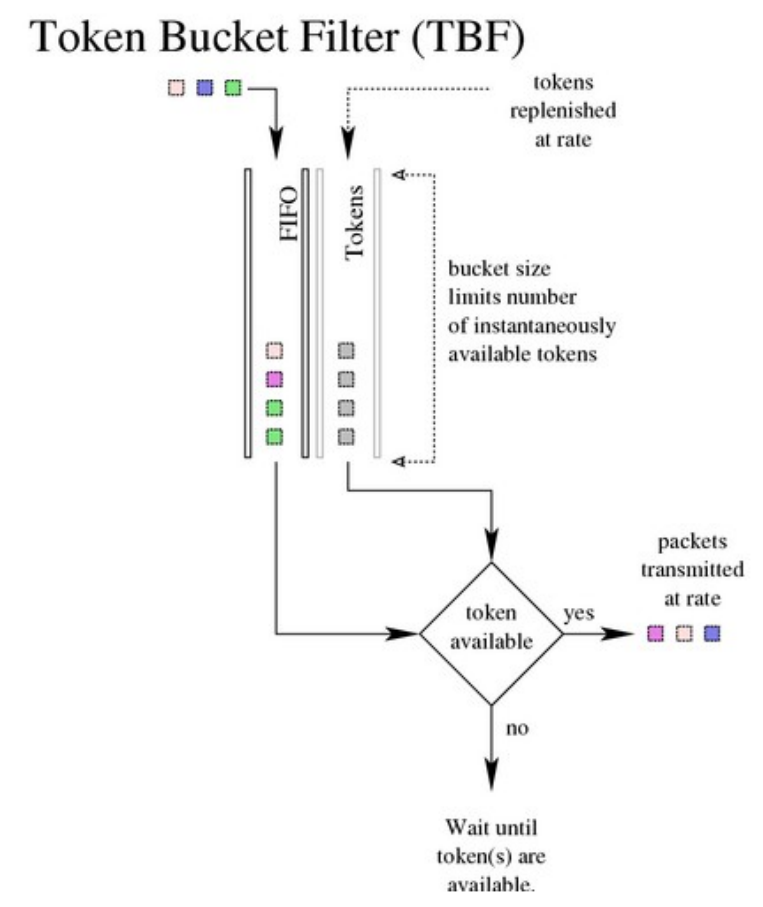
- ***red*** (Random Early Detection)
  - Simula la congestión física mediante la reducción aleatoria de paquetes.
  - Aplicaciones con un ancho de banda muy grande.
- ***sfq*** (Stochastic Fair Queueing)
  - Intenta de distribuir equitativamente la posibilidad de retransmitir un paquete a la red entre un número arbitrario de flujos.



Qué es *tc* (10/11)

QDISCS SIN CLASES: (cont.)

- ***tbf*** (Token Bucket Filter)
  - Pasan paquetes si no se supera la tasa especificada,
  - Posibilidad de emitir ráfagas que exceden esta tasa.



Qué es *tc* (11/11)

<u>*QDISCS* CON CLASES:</u>

- ***CBQ*** (Class Based Queueing)
  - Implementa jerarquía de clases de intercambio.
  - Capacidad de modelar elementos y de priorizar.
- ***HTB*** (Hierarchy Token Bucket)
  - *tokens* y *buckets* + clases y filtros.
  - Control complejo y granular del tráfico.
- ***PRIO*** (Priority Scheduler)
  - Cuando se puede desencolar un paquete se mira si la primera clase tiene algún paquete.
  - Si no lo tiene, se mira la siguiente.

Trabajando con *tc* (1/6)

- Comandos:
  - ***add***: añade *qdisc*, clase o filtro al nodo correspondiente.

```
tc qdisc add dev (device_name) [parent qdisc-id|root] [handle
        qdisc-id] qdisc [qdisc specific parameters]

tc class add dev (device_name) parent qdisc-id [classid class-id]
            qdisc [qdisc specific parameters]

tc filters add  dev (device_name) [ parent qdisc-id | root ]
protocol (protocol) prio priority filtertype [ filtertype specific
                parameters ] flowid flow-id
```

  - ***del***: elimina *qdisc*.

```
tc qdisc del dev (device_name) [parent qdisc-id|root] [handle
                    qdisc-id] qdisc
```

Trabajando con *tc* (2/6)

- ***change***: cambiar características de las entidades creadas.

```
tc qdisc change dev (device_name) [parent qdisc-id|root] [handle qdisc-id] qdisc [qdisc specific parameters]

tc class change dev (device_name) parent qdisc-id [ classid class-id ] qdisc [ qdisc specific parameters ]

tc filters change dev (device_name) [ parent qdisc-id | root ] protocol (protocol) prio priority filtertype [ filtertype specific parameters ] flowid flow-id
```

- ***replace***: reemplaza un nodo por otro.

```
tc qdisc replace dev (device_name) [parent qdisc-id|root] [handle qdisc-id] qdisc [qdisc specific parameters]

tc class replace dev (device_name) parent qdisc-id [ classid class-id ] qdisc [ qdisc specific parameters ]

tc filters replace dev (device_name) [ parent qdisc-id | root ] protocol (protocol) prio priority filtertype [ filtertype specific parameters ] flowid flow-id
```

Trabajando con *tc* (3/6)

- ***link***: realiza sustitución, parecido a *replace*. Solo para *qdisc*.

```
tc qdisc link dev DEV [parent qdisc-id|root] [handle qdisc-id]
qdisc [qdisc specific parameters]
```

- ***show***: muestra información, tiene las siguientes opciones:
  - *-s, -stats, - statistic.*
  - *-d, -details.*
  - *-r, -raw.*
  - *-p, -pretty.*
  - *-iec*.

```
tc [ -s, -d, -r, -p, -iec ] qdisc show [ dev DEV ]

tc [ -s, -d, -r, -p, -iec ] class show dev DEV

tc filter show dev DEV
```

Trabajando con *tc* (4/6)

- Ejemplos:
  - Ejemplo de creación de *qdisc*:

```
tc qdisc add dev ppp0 root tbf rate 220kbit latency 50ms burst 1540
```

  - Ejemplo de creación de una clase:

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 10 mbit burst 15k
```

  - Ejemplo de creación de un filtro:

```
tc filter add dev eth0 parent 1:0 protocol ip prio 10 u32 match ip tos 0x10 0xff flowid 1:4
```

  - Ejemplo de eliminación de *qdisc*:

```
tc qdisc del dev ra0 root handle 1: prio
```

  - Ejemplo para visuzliar elementos creados:

```
tc -s qdisc show dev eth0
```

Trabajando con *tc* (5/6)

- Parámetros *qdisc* específicos más utilizados:
  - *burst*: tamaño del cubo en bytes.
  - *rate*: tasa de velocidad.
  - *latency*: periodo máximo de tiempo que puede pasar entre paquetes.
  - *ceil* o *cell*: tiempo de transmisión escalonado en función del tamaño del paquete.
  - *limit*: cantidad de paquetes que serán encolados.
  - *mpu*: tamaño mínimo del paquete.

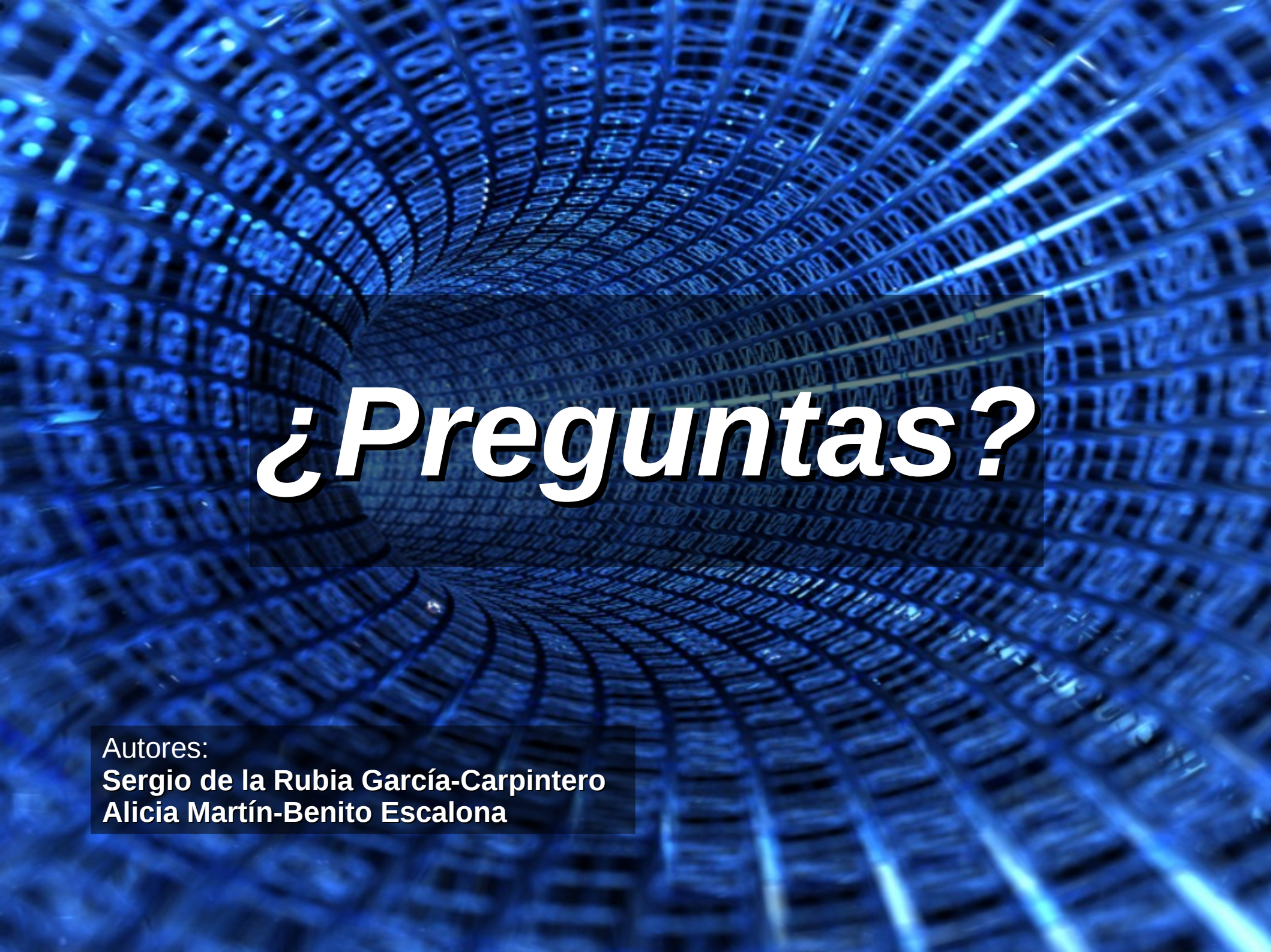
Trabajando con *tc* (6/6)

- Ejemplo práctico:
  - Creamos una *qdisc htb*, dentro de la cual vamos a crear dos subclases *htb* con el mismo ancho de banda.

```
# tc qdisc add dev eth0 root handle 1: htb default 1

# tc class add dev eth0 parent 1: classid 1:1 htb  rate 3125kbps

# tc class add dev eth0 parent 1: classid 1:2 htb  rate 3125kbps
```

  - Una vez que la *qdisc* y las subclases están creadas, se crean filtros para ambas subclases.

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 10 u32 match
ip tos 0x10 0xff flowid 1:2


# tc filter add dev eth0 parent 1:0 protocol ip prio 9 u32 match
ip tos 0x08 0xff flowid 1:1
```



Autores:
Sergio de la Rubia García-Carpintero
Alicia Martín-Benito Escalona