



Bisoños Usuarios de Linux de Mallorca y Alrededores | Bergantells Usuaris de Linux de Mallorca i Afegitons

## Método para la ecualización del ancho de banda

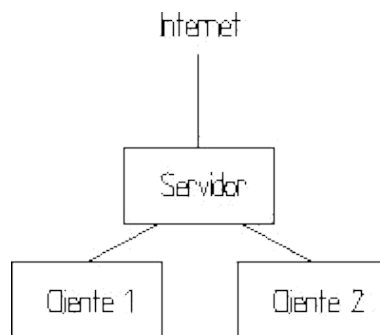
Por Jose Luis Nogueira, [scero](http://scero.homelinux.org:8080) (<http://scero.homelinux.org:8080>)  
Creado el 07/04/2003 21:02 y modificado por última vez el 07/04/2003 21:02

En este artículo expongo un método con el cual podréis "ecualizar" el ancho de banda de vuestra conexión, gracias a las herramientas QoS.

### 1. Introducción

Desde hace algún tiempo he estado intentado ecualizar el ancho de banda de mi conexión, con el fin de poder garantizar que siempre se dispone de un ancho de banda mínimo para un determinado servicio o máquina de una red.

Para entender la utilidad de esto supongamos que tenemos una red como la siguiente



es muy habitual encontrarse con el hecho de que el *cliente 1* comienza a utilizar ancho de banda de forma masiva y continua (ftp, p2p, etc...); de manera que si en un momento posterior el *cliente 2* intenta utilizar la conexión para cosas más livianas (navegar), este descubre que la conexión esta copada por el primero y todo el acceso le va muy lento.

En este artículo pretendo explicar un método para poder evitar esta u otras situaciones en las cuales el ancho de banda es utilizado de una manera poco justa entre los usuarios (servicios) o como mínimo de una manera no adecuada. Entre otras cosas pretendo explicar la forma de conseguir lo siguiente:

- Asegurar que no haya en la red usuarios (o servicios) que utilizan el ancho de banda ocasionando un acceso precario para los otros
- Establece prioridades entre las máquinas cliente de nuestra red (si esto fuese necesario)
- Garantizar un ancho de banda mínimo para cada servicio o grupo de servicios (ftp, telnet, ssh, www, etc ...)

No quiero que ninguno piense que este es el único método para poder ecualizar un ancho de banda, esto sería un error. Lo único que pretendo en este artículo es dar a conocer algunos de los servicios QoS (quality of service) que están disponibles en el núcleo de Linux; estos servicios se pueden utilizar de muchas formas y combinar entre ellos para solucionar problemas muy complejos. En general el método que os voy a explicar es muy potente y simple a la vez, de todos modos si alguno cree que puede mejorarse de alguna manera, estoy abierto a todo tipo de sugerencias.

En este artículo parto de la base de que todos los lectores:

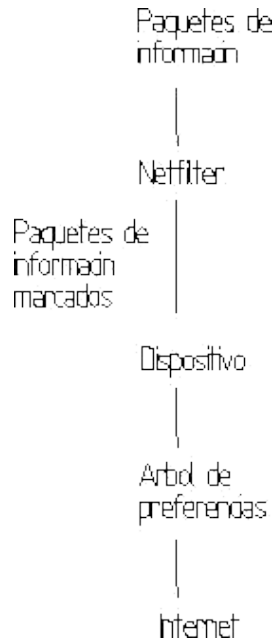
- Sabéis compilar el kernel de Linux



- Sabéis usar netfilter (iptables) de los kernels 2.4.x
- Tenéis conocimientos mínimos de administración de redes en Linux

## 2. Método utilizado para la ecualización

Este método se puede representar de manera gráfica de la siguiente manera:



**Aviso:** este gráfico es una simplificación de la realidad, únicamente pretende servir con fines didácticos; el proceso que realmente sigue Linux es un poco más complejo e intervienen más factores. Como se puede observar la idea es utilizar *Netfilter* para marcar los paquetes según el tratamiento que se desee darles, cuando estos paquetes lleguen al dispositivo de red por el que deben salir a internet (o cualquier otra red) son tratados según un árbol de preferencias que especifica el ancho de banda que le corresponde, la prioridad, etc...

En los siguientes puntos se encuentra una explicación detallada de cómo implementar este método.

### 2.1. Compilación de Kernel

Lo primero que se debe hacer es habilitar todos los soportes a utilizar en el Kernel.

Para poder emplear Netfilter, como mínimo:

```

Networking options -->
[*] Network packet filtering (replaces ipchains)

IP: Netfilter Configuration -->
<*> netfilter MARK match support
<*> Connection state match support
<*> Packet filtering
<*> Packet mangling
<*> MARK target support
  
```

Si adicionalmente se desea hacer *Masquerading* y demás historias también se deberían activar sus correspondientes soportes.

Para poder usar los servicios *QoS* necesarios para este método es necesario:



```

Networking options --->
  QoS and/or fair queueing --->
    [*] QoS and/or fair queueing
    <*> HTB packet scheduler
    <*> SFQ queue
    <*> Ingress Qdisc
    [*] QoS support
      [*] Rate estimator
    [*] Packet classifier API
      <*> TC index classifier
      <*> Routing table based classifier
      <*> Firewall based classifier
    [*] Traffic policing (needed for in/egress)

```

## 2.2. Creación de un árbol de preferencias para un dispositivo

Con árbol de *preferencias de preferencias* pretendo hacer referencia (de forma informal) a una estructura de ``clases'' o ``bandas'' (según la traducción) en la que hay dependencias padre-hijo entre sus miembros.

Para poder implementar esta estructura es necesario conocer los algoritmos que voy a emplear en cada una de sus clases. Hay que tener en cuenta que no son lo únicos que se pueden aplicar, existen otros con diferentes características:

### Algoritmo sfq (*Stochastic Fairness Queueing*)

Es una implementación sencilla de la familia de algoritmos de colas justas (*fair queueing*).

En este algoritmo el tráfico se divide en un número bastante grande de colas FIFO, una por cada conversación. Entonces se envía el tráfico de una manera parecida a round robin, dando a cada sesión por turnos la oportunidad de enviar datos.

Es decir es un algoritmo que busca una equidad entre todas las peticiones para repartir de la manera más justa posible el ancho de banda.

### Algoritmo htb (*Hierarchical Token Bucket*)

Es un algoritmo utilizado para dividir el ancho de banda según nos interese de manera que se puede especificar un *ancho de banda mínimo* y un *ancho de banda máximo*. De manera que el algoritmo asegura la disponibilidad del mínimo, y en caso de que haya ancho de banda libre se puede llegar hasta el máximo.

Existen otros algoritmos que hacen cosas similares, pero este es el único (por lo que he probado) que permite pedir *prestado* ancho de banda que no se use.

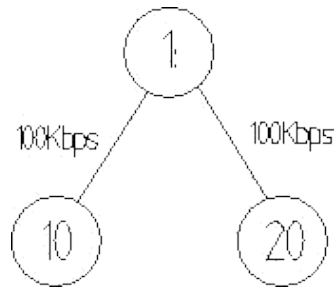
### Reparto de los anchos de banda en el árbol de preferencias

Para crear el árbol cada uno puede aplicar los criterios que más le convengan, en este apartado explicaré como hacerlo y daré un par de ideas que pueden resultar interesantes.

Para los ejemplos que hay a continuación se supone:

- La conexión a internet (podría ser cualquier otra red) es de 300Kbps
- La conexión a internet se realiza por el interface *eth1*
- Interesa crear dos *bandas* de similar *ancho de banda*, una se usará para navegar por la web y la otra para todos los demás servicios
- La *banda* empleada para navegar por la web será prioritaria frente a la otra

Para este caso tan sencillo, se puede representar el árbol de preferencias como



para conseguir esto se puede proceder de la siguiente manera:

```
tc qdisc add dev eth1 root handle 1: htb default 20
```

Con esto se crea una banda principal (*root*) con el nombre *1:*, que por defecto manda todos los paquetes a la banda 20.

```
tc class add dev eth1 parent 1: classid 1:10 htb rate 100kbps ceil 200kbps
```

Con esto se crea una banda que depende de la principal (*parent 1:*), tiene un ancho de banda mínimo de 100kbps (*rate 100kbps*) y puede llegar a tener hasta 200kbps (*ceil 200kbps*). Como no se ha indicado nada sobre la prioridad se supone que tiene prioridad máxima (es decir 0). A esta banda se le llama 10 y depende de la 1 (*1:10*).

```
tc class add dev eth1 parent 1: classid 1:20 htb rate 100kbps ceil 200kbps prio 1
```

Con esto se crea una banda de similares características a la anterior con la salvedad de que tiene menor prioridad (*prio 1*). Su nombre es 20, al depender de la 1 se denomina (*1:20*). Esta será la banda por defecto para los paquetes (*default 20*), como se indico en el primer comando.

En este árbol cada una de las bandas tiene un ancho de 100kbps y puede crecer hasta 200kbps. Es importante tener el cuenta que los comandos:

```
tc qdisc add dev eth1 root handle 1: htb default 20
```

```
tc class add dev eth1 parent 1: classid 1:10 htb rate 100kbps ceil 300kbps
```

```
tc class add dev eth1 parent 1: classid 1:20 htb rate 100kbps ceil 300kbps prio 1
```

En la práctica con estos comandos se genera un árbol de idénticas características al anterior, pues si la conexión solo dispone de 300kbps y 100kbps son asignados de forma fija para cada banda, solo quedan 100kbps sin ocupar que son los que pueden repartirse la banda 10 y 20.

En muchos casos este modelo es indeseable pues si solo se utiliza una de las bandas, por ejemplo la 10 (para navegar) se está desaprovechando mucha velocidad, pues como máximo se puede llegar a 200kbps en lugar de los 300kbps, por lo que se está infrautilizando la conexión.

La solución a este problema es conseguir que las bandas se *presten* velocidad si no lo usan, conseguir esto es muy sencillo ya que en las colas *htb* siempre se comportan de este modo, con la salvedad de que *la banda root (1) no puede prestar ancho de banda a sus hijas*. Por tanto la solución se podría dar como:

```
tc qdisc add dev eth1 root handle 1: htb default 20
```

```
tc class add dev eth1 parent 1: classid 1:1 htb rate 300kbps ceil 300kbps
```

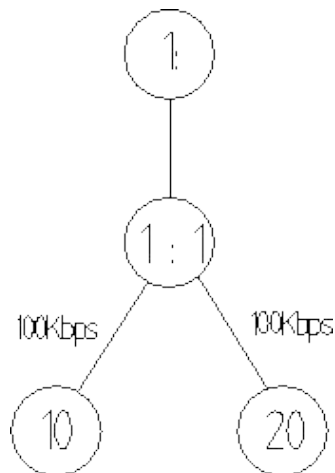
```
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 100kbps ceil 300kbps
```



```
tc class add dev eth1 parent 1:1 classid 1:20 htb rate 100kbps ceil 300kbps prio 1
```

En este caso la banda *root* (1). Tiene una única hija la banda (1:1) (que acapara todo el ancho de la conexión) la cual a su vez tiene dos hijas (1:10) y (1:20), que tienen características similares a las ya descritas en el caso anterior. En este caso la banda 1:1 puede prestar su conexión a sus hijas, de modo que cada hija tiene su ancho de banda mínimo 100kbps y puede llegar a tener hasta 300kbps, si este no se usa por otra.

Gráficamente este segundo ejemplo se puede representar como



Como es evidente este es el ejemplo más sencillo que puede darse con únicamente dos bandas, si fuese necesario se pueden crear tantas bandas como sea necesario (1:30, 1:40, ...) según necesidades.

### Combinación con colas sfq

Se puede ganar mucho en comportamiento si se combinan las clases *htb* con clases *sfq* que se encarguen de repartir justamente las peticiones que reciben, por ese motivo es muy recomendable combinar los *árboles htb* con estas. Esto es tan sencillo como asignar una cola de este tipo para cada banda final, esto se puede hacer de este modo:

```
tc qdisc add dev eth1 parent 1:10 handle 10: sfq
```

```
tc qdisc add dev eth1 parent 1:20 handle 20: sfq
```

...

## 2.3. Marcado de paquetes (Netfilter)

Ahora se debe utilizar la herramienta *iptables* para marcar los paquetes con valores; posteriormente los paquetes serán filtrados según los valores con los que ahora se marquen. Para esto no existe una receta "mágica" pues depende de lo que se desee en cada caso.

Por ejemplo si se tiene una red con un servidor que da acceso a internet y dos clientes (similar al primer gráfico que use en este artículo). De manera que lo que interesa es dar un tratamiento diferente a los paquetes que llegar de cada máquina, se pueden usar las siguientes reglas:

```
iptables -A FORWARD -s 192.168.1.13 -t mangle -j MARK --set-mark 1
```

```
iptables -A FORWARD -s 192.168.1.18 -t mangle -j MARK --set-mark 2
```

```
iptables -A POSTROUTING -o eth1 -p tcp -dport 20:21 -t mangle -j MARK --set-mark 3
```



De este modo los paquetes de la máquina con ip *192.168.1.13* son marcados con el valor *1*, los paquetes de la máquina con ip *192.168.1.18* con el valor *2* y los paquetes con destino a un ftp con el valor *3*.

## 2.4. Asociación de marcas con las bandas del árbol de preferencias

Para finalizar se debe asociar a cada marca una banda, para que de este modo según interese se de un tratamiento o otro a cada paquete. Esto se puede hacer con el comando:

```
tc filter add dev (dispositivo) protocol ip parent (clase_padre) handle 1 fw classid (clase_destino)
```

Si interesase que todos los paquetes marcados con el valor *1* fuesen a la banda *10*, se podría usar algo similar a:

```
tc filter add dev eth1 protocol ip parent 1: handle 1 fw classid 1:10
```

## 3. Ejemplo completo

Supongamos que dada una red se desea separar las conexión a ftp del resto de conexiones a internet, de manera que se de mucha más prioridad a las conexiones ftp. En esta supuesta red, el servidor (donde se implementará el sistema *QoS*) da servicio de conexión a toda la red por masquerading (sobre iptables). El supuesto servidor se comunica por el dispositivo *eth0* con la red local y con *eth1* con internet, la velocidad de conexión es de *512kbps*.

Para conseguir lo deseado se podría, por ejemplo, escribir los siguientes comandos:

```
# Creación del árbol de bandas

tc qdisc add dev eth1 root handle 1: htb default 20 # Por defecto toda la información irá a la banda 20

tc class add dev eth1 parent 1: classid 1:1 htb rate 512kbps ceil 512kbps

tc class add dev eth1 parent 1:1 classid 1:10 htb rate 300kbps ceil 512kbps

tc class add dev eth1 parent 1:1 classid 1:20 htb rate 200kbps ceil 512kbps prio 1 # Esta banda tiene menor prioridad y menor ancho de banda mínimo

# Asociación de colas sfq con bandas

tc qdisc add dev eth1 parent 1:10 handle 10: sfq

tc qdisc add dev eth1 parent 1:20 handle 20: sfq

# Se asocia la marca 1 con la banda 10

tc filter add dev eth1 protocol ip parent 1: handle 1 fw classid 1:10

# Reglas de filtrado (se marca con un 1 a todos los paquetes destinados a un ftp)

iptables -A FORWARD -i eth0 -o eth1 -p tcp -dport 20:21 -t mangle -j MARK --set-mark 1 #
Se marca con 1 todos los paquetes con destino a un ftp
```

## 4. Herramientas de verificación

Para controlar que el sistema se comporta de forma correcta se pueden usar dos comandos, el primero muestra el comportamiento de la banda *root (1:)* y las bandas *sfq*. Este comando es:

```
tc -s qdisc ls dev ethx
```



Hay que substituir la  $x$  por el valor del dispositivo a mirar.

Si se desea controlar el comportamiento de *htb* es necesario usar:

```
tc -s -d class show dev ethx
```

Hay que substituir la  $x$  por el valor del dispositivo a mirar.

Al controlar los *htb* es interesante sobre todo controlar los campos *lended* y *borrowed*; pues muestran el comportamiento de préstamos de ancho de banda.

Espero que os sirva de utilidad y lo disfrutéis. Si encontrais algún error en el documento agradecería que me lo comunicaseis. Gracias.

---

E-mail del autor: scero@ono.com

Podrás encontrar este artículo e información adicional en: <http://bulmalug.net/body.phtml?nIdNoticia=1727>