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

Wonder Shaper : Gestiona eficientemente el ancho de banda de tu conexión adsl (o cable) (61659 lectures)

Per **Kiko Piris**, ***kiko*** ()

Creado el 02/10/2002 09:11 modificado el 02/10/2002 09:11

Cuando "compraste" tu conexión adsl (o cable) te dijeron que el ancho de banda de bajada era independiente del de subida. Pero por más pruebas que haces, ¿ no consigues que éso parezca real ?

** ¿ Quieres poder navegar a la velocidad de siempre (o casi) mientras subes megas a saco a tu servidor ?*
** ¿ Quieres poder trabajar "en condiciones" con una sesión de ssh (o incluso navegar decentemente) mientras te bajas megas a todo trapo ?*

Si es así, sigue leyendo ...

Linux dispone de unos algoritmos de "Quality of Service" (gestión del ancho de banda o "bandwidth management") que ya quisieran muchos routers.

Primero un poco de teoría -light- (puedes saltartelo si tienes mucha prisa :-) :

Si el ancho de banda de subida y de bajada son independientes, ¿ por qué la cosa no funciona como debería desde un principio ?

Por culpa de las colas. Cuando el tráfico de subida es grande, se forman colas largas de paquetes en la cola "Send-Q" (del router). El servidor del cual intentamos bajarnos algo deja de enviarnos paquetes hasta que recibe los ack's (y esos ack's deben atravesar una cola "Send-Q" congestionada).

¿ Qué se puede hacer ?

Limitar la velocidad de subida un poco por debajo de nuestro ancho de banda. Así conseguimos que las colas "largas" se formen en nuestro equipo (y no en el router). Ahora podemos hacer que los paquetes del tipo "ack" y de tráfico interactivo pasen por delante de los otros y no tengan que "hacer cola".

Nos queda un punto por resolver. La cola de bajada. Esta cola se encuentra en los equipos de nuestro ISP. Si la cola de bajada es demasiado larga, el tráfico interactivo se verá afectado negativamente. La solución para evitar ésto es limitar la velocidad de bajada un poquito con el siguiente "truco" : si los paquetes llegan "demasiado rápido", los descartaremos y entonces el emisor adecuará su velocidad de transmisión a la nuestra.

De esta manera **sacrificando un poquito** de nuestro ancho de banda conseguiremos 2 cosas: **aprovechar los anchos de banda independientes de subida y de bajada**, y que el tráfico "masivo" **no nos deje la latencia del tráfico interactivo por los suelos**. Casi nada !

Ahora lo práctico :

¿ Qué necesito ?

Wonder Shaper[1] es un script muy sencillo que nos definirá las colas necesarias para implementar lo que he comentado en el punto anterior. De hecho la información está sacada directamente del README, os recomiendo su lectura.

Pero antes de ponernos con el script, necesitamos habilitar (si no lo están ya) un par de cosillas en nuestro kernel 2.4 (o superior).

Debemos tener al menos las siguientes opciones activadas dentro de "**Networking Options**" / "**QoS and/or Fair Queueing**" : **CBQ**, **PRIO**, **SFQ**, **Ingress Qdisc**, **QoS support**, **Rate Estimator**, **Packet classifier**, **Firewall based classifier**. **U32 classifier** y **Traffic Policing**. Aunque puedes activar todo lo de dentro de **QoS and/or Fair Queueing** sin problemas.

También hemos de **instalar el paquete iproute**[2].

Wonder Shaper[1] nos proporciona dos versiones del script : "**wshaper**" que nos servirá con cualquier kernel 2.4 y que usa el algoritmo CBQ. Y "**wshaper.htb**" que usa el algoritmo HTB[3] (**HTB no viene incluido en todos los kernels 2.4**, viene **a partir del 2.4.21**).

CBQ es un algoritmo más usado y testeado pero técnicamente inferior a HTB (?). La tendencia parece ser que es hacia HTB. Yo lo he implementado usando HTB y no he tenido ningun problema (os lo recomiendo).

Pero **antes hay que hacer una comprobación:** si vamos a usar HTB, la versión del algoritmo HTB del kernel debe corresponderse con la versión del binario tc que viene en el paquete iproute. Si **al ejecutar el script wshaper.htb os da errores** y en el syslog se os queja de la versión del HTB, buscad una versión más reciente de iproute (el enlace lo teneis al principio de la página del Wonder Shaper[1]).

Nota para debianitas: Si usais Debian Sid, no problem. Si usais Debian Woody, lo más sencillo es tirar de backports[4]. Concretamente de este[5].

Una vez tenemos nuestro **kernel listo** y el **iproute adecuado instalado**, pasamos al script que define las colas.

Aquí me centraré en wshaper.htb que es el que he usado yo (aunque tampoco es que haya mucho que comentar ...).

Lo único que tenemos que hacer es definir los anchos de banda de bajada y de subida y el interfaz de red donde tenemos conectado el router. Modificando las siguientes líneas (son Kilobits/s) :

DOWNLINK=208
UPLINK=120
DEV=eth0

Los valores mostrados son los que me han funcionado a mí con una línea **adsl de telefónica de 256 Kbits**. Se trata de ir probando y ajustando hasta encontrar los valores óptimos para nuestra conexión (cómo dicen los ingleses: YMMV).

Copiamos el script en /etc/init.d/ y ponemos los enlaces simbólicos correspondientes en /etc/rc.X/ (ojo, esto es para Debian, para otras distros, consultar la documentación de los scripts de arranque de los servicios, YMMV).

En el script wshaper.htb pueden personalizarse otras cosillas, para mas información leeros el README y los comentarios que encontrareis en el mismo script.

Con HTB[3] pueden hacerse **auténticas virguerías**, pero éso se escapa del objetivo de este artículo.

PD.: Si alguien lo prueba con otra conexión a Internet, estaría bien que pusiera los valores que haya encontrado como comentarios para compartir la información.

Actualización para los usuarios de mldonkey :

Acabo de encontrar (02/04/2003) en la sección de howto's[6] de la página de mldonkey[7] un artículo[8] donde comenta (en la página 3[9]) que para que el shaping funcione correctamente con el mldonkey hay que hacer una pequeña modificación al script wshaper.htb.

Se trata de añadir las siguientes lineas :

```
  # mldonkey UDP source propagation packets are small, but go to
# bulk priority class
tc filter add dev $DEV parent 1:0 protocol ip prio 10 u32 \
        match ip protocol 17 0xff \
        match ip sport 4666 0xffff \
        flowid 1:30
```

Entre los párrafos :

```
  # ICMP (ip protocol 1) in the interactive class 1:10 so we
# can do measurements & impress our friends:
```

y

```
  # To speed up downloads while an upload is going on, put ACK packets in
# the interactive class:
```

Nota: Cuando digo párrafo, quiero decir el comentario citado y la/s instrucciones a que hace referencia dicho comentario.

Lista de enlaces de este artículo:

1. http://lartc.org/wondershaper/
2. ftp://ftp.inr.ac.ru/ip-routing/
3. http://luxik.cdi.cz/~devik/qos/htb/
4. http://backports.org/
5. http://backports.org/package.php?search=iproute
6. http://mldonkey.berlios.de/modules.php?name=Content
7. http://mldonkey.berlios.de/
8. http://mldonkey.berlios.de/modules.php?name=Content&pa=showpage&pid=5
9. http://mldonkey.berlios.de/modules.php?name=Content&pa=showpage&pid=5&page=3

E-mail del autor: bulma _ARROBA_ pirispons.net
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1527