

Control de Ancho de Banda

Una tarea cada vez más solicitada a los administradores que no podíamos dejar de contemplar en nuestras páginas: iproute2 al rescate!

Tanto tiempo sin escribir un artículo me ha oxidado un poco: ya cuento 5 introducciones diferentes, y ninguna me gusta. Por supuesto, es totalmente cliché realizar el comentario que acabo de hacer, pero no me va a provocar un error cíclico de redundancia repetirlo. Si, nerd hasta la muerte!

Hoy les quiero presentar un artículo que les permita comprender, para posteriormente aplicar, los conceptos de administración del ancho de banda con las herramientas GNU GPL que trabajan en conjunto con el kernel Linux. En este caso hablaremos de Netfilter e iptables, iproute2 y una de sus utilidades: tc, y del Traffic Control Super Script, una linda aplicación que permite especificar reglas de limitación de ancho de banda en base a IP y puerto de origen y/o destino mediante unos simplifadamente complejos archivos de configuración. También hablaremos de CBQ (Class-Based Queuing: Su traducción puede sonar vulgar pero es “Encolamiento Basado en Clases”) a nivel conceptual, aunque al día de hoy ya contamos con otras disciplinas como HTB que veremos más adelante, en una próxima entrega si a los lectores les interesa.

En principio CBQ presenta la capacidad de otorgar el ancho de banda requerido por cada clase en un intervalo de tiempo especificado, si hubiera demanda del mismo. Esto se logra mediante un mecanismo similar al utilizado por los delay_pools de Squid para limitación de ancho de banda de Proxy HTTP, aplicando “esperas” entre las transferencias de paquetes.

En segunda instancia CBQ permite que las clases “tomen prestado” ancho de banda no utilizado por otras clases.

El Ancho de Banda en sí mismo es una función del tamaño y el tiempo: por ejemplo, la velocidad la medimos en metros por segundo. En el mundo de las comunicaciones, medimos bits, bytes o algún múltiplo por segundo. De esta forma, tenemos que en un vínculo de 512kbit/s logramos una velocidad o tasa de transferencia de 64 kilobytes por segundo, ya que $8 \text{ bit} = 1 \text{ byte}$ y por lo tanto $512 / 8 = 64$. (Olvidemos por un momento que en ADSL o Cablemodem tenemos diferentes anchos de banda dependiendo de si estamos enviando o recibiendo datos). De aquí que las limitaciones de ancho de banda se realicen intercalando esperas en la transmisión / recepción de datos, como en los delay_pools que comentábamos.

Si suponemos un escenario típico, tendremos direcciones IP (computadoras, o dispositivos) que intentan acceder a recursos IP de otras redes, pasando a través de un gateway. Esto se puede aplicar tanto a una configuración NAT clásica: direcciones IP privadas accediendo a internet a través de un gateway GNU+Linux con dos o más interfaces de red.

Como sea, logramos que todas las solicitudes hacia Internet pasen a través de nuestro gateway por lo que podremos controlar, no solo que protocolos o combinaciones de puertos o IP de origen o destino permitir, sino también el ancho de banda permitido.

En principio quise hablar del script CBQ.init, que permite armar facilmente reglas de CBQ mediante el comando “tc” del paquete iproute2, pero le quise dar la oportunidad a otra aplicación llamada Traffic Control Super Script / TCSS. De todas formas me gustaria comentar que CBQ.init resulta útil cuando deseamos aprender a crear nuestros propios comandos “tc”, ya que mediante su función “compile” podemos obtener la traducción de los archivos de configuración CBQ.init que hagamos a comandos “tc”.

Asimismo, existen otras aplicaciones para este mismo propósito, como Snitch y HTB.init. Pasemos, finalmente, a revisar el TCSS:

Traffic Control Super Script

Ante todo: el tcss no lo he visto ni en Slackware, ni Gentoo, ni SuSE ni Mandrake ni ninguna otra: van a tener que bajarlo del sitio oficial (ver recuadro).

IMPORTANTE: El autor recomienda que el firewalling y el control del ancho de banda lo realice un servidor GNU+Linux realizando bridging. El TCSS (y demás derivados de iproute2/tc) se puede aplicar aunque no trabajemos de dicha forma, la cual veremos documentada en nuestras páginas en un futuro cercano.

Al descargar TCSS obtendremos un archivo .tar.gz. Desempaquetémoslo a donde deseemos. El autor prefirió utilizar el directorio /etc/rc.d/tcss como contenedor de configuración y aplicación, ya que la misma es muy pequeña.

El primer paso consiste en editar el archivo de configuración, llamado, increíblemente, “config”. Ese archivo ya existe, nos limitaremos a cambiar algunos parámetros. Recuerden que tanto la aplicación como la configuración residen en un mismo directorio, en nuestro caso usaremos /etc/rc.d/tcss. Este archivo, aparte de los clásicos comentarios al principio del mismo, contiene estas líneas:

```
path='/etc/rc.d/tcss01f';
hostsfile="$path/tcss-hosts";
devicesfile="$path/tcss-devices";
shape_ports='all';
shape_parent_classid='0';
shape_protocol='tcp';
shape_prio='1';
debug=false;
version="0.1f";
allow_version_check=true;
```

Las líneas que nos interesa cambiar probablemente sean tan solo “path” para que apunte a “/etc/rc.d/tcss” y “allow_version_check” en false. Esta variable le indica a TCSS que automáticamente verifique contra el sitio oficial del tcss en internet si hay versiones nuevas disponibles. Si quieren aprovechar esta funcionalidad, sepan que necesitan tener el paquete “wget” instalado. Luego debemos editar el archivo tcss-devices, el cual indica cual es la interfaz de red conectada a Internet, y cual es la interfaz de nuestros “clientes”. Viene uno de ejemplo, y es el siguiente:

```
eth0 8139too dstif 100Mbit 10Mbit 10 cbq 1000 on
eth1 8139too srcif 100Mbit 10Mbit 11 cbq 1000 on
```

En este caso vemos que hay dos interfaces de 100Mbit (**Atención**, hablamos del ancho de banda físico del dispositivo, y no del ancho de banda con el que cuenta nuestra conexión a Internet!), eth0 y eth1, las cuales corresponden a la interfaz que lleva a nuestros clientes (eth0, configurada como “dstif”, que usa el módulo del kernel 8139too) y la que lleva a Internet (eth1, configurada como “srcif”, que también usa el módulo del kernel 8139too). Pero existen más parámetros en este archivo. Ya vimos las 4 primeras columnas. La quinta columna, según una regla estándar de CBQ, es el valor de la cuarta dividido 10. En este caso, $100\text{Mbit} / 10 = 10\text{Mbit}$. Luego, la sexta columna es un número identificador de la interfaz, único. El autor recomienda usar valores menores a 20 en este archivo. En este caso usa “10” y “11”. El séptimo campo indica que deseamos usar “cbq”. El octavo campo es el tamaño promedio de los paquetes. El autor indica “siempre 1000”. Un valor entre 1000 y 1500 es normal. La última columna simplemente “activa”, o no, la interfaz para ser usada en TCSS.

El último archivo a configurar es tcss-hosts. Este archivo indica por cada línea una regla de control de tráfico diferente. Cada regla consta de 13 parámetros, y cada valor / columna debe estar separado del otro exactamente por una tabulación. (Tecla “tab”). Los 13 parámetros, ordenados tal cual deben aparecer en el archivo, son: (1) nombre de la regla, (2) dirección IP a limitar (“cliente”), (3) IP o red contra el cual realizar el control de tráfico (relativo a la IP “cliente”), (4) identificador de la regla (numero **siempre** par empezando por 22: 22, 24, 26, etc), (5) sentido del tráfico a controlar, (6) puerto destino, (7) puerto de origen, (8) protocolo, (9) velocidad, (10) velocidad dividido 10, (11) regla activada o no, (12) regla bidireccional o no, y por último, (13) usar u32 o iptables para la clasificación.

El parámetro 1 tan solo es un nombre descriptivo de la regla. El parámetro 2 indica a que dirección IP o red limitaremos para la regla en cuestión. El tercer campo indica si esta regla sólo se aplica cuando el destino es una cierta IP o red. En combinación con los campos 6 y 7 podemos armar reglas que tengan en cuenta diferentes combinaciones de IP de origen / destino y puerto de origen / destino según protocolo (campo 8, tcp/udp)! El cuarto campo es un valor que, como recordarán, debe ser mayor o igual a 22, en incrementos de 2 en 2, ya que el identificador de dispositivos es menor o igual a 20. El quinto campo indica si es una regla de bajada (dstif) o subida (srcif). Este campo cobra especial significado si al parámetro 12 no lo configuramos como bidireccional (“bi” o “notbi”, según consideremos).

El campo 9 y, por consiguiente, el campo 10, son el eje de este artículo: el campo 9 es el ancho de banda que deseamos permitir que el cliente “consume”. El campo 10 es aproximadamente un 10% del valor del campo 9.

El campo 12 permite definir si el control de tráfico también se realizará en el sentido “de retorno”, relativo a la regla en cuestión (si es dstif o srcif). Podemos limitar la velocidad con la que un usuario baja archivos del puerto 80 de cualquier red de destino (0/0), pero quizá no deseamos limitar con que velocidad puede realizar solicitudes http!. Si deseamos aplicar la regla en ambos sentidos, especificaremos “bi”. Caso contrario, “nobi”. La última columna toma sentido si nuestros clientes se encuentran NATeados, o sea, no tienen IP

pública. En ese caso no se puede utilizar el clasificador u32 (“notables”), sino que se debe usar iptables, especificando el parámetro “tables”.

El archivo de configuración de ejemplo, es el siguiente:

```
kegan-ssh 10.10.11.2 0/0 22 dstif any 22 tcp      128Kbit 12Kbit on bi
notables
```

```
kegan-http 10.10.11.2 0/0 24 dstif 80 any tcp    64Kbit 6Kbit on bi
notables
```

```
kegan-all 10.10.11.2 0/0 26 dstif any any all   33Kbit 3Kbit on bi
notables
```

Para cerrar...

Investiguen Snitch con I7-filter, que permite armar reglas de control de tráfico en base al protocolo de capa 7. (HTTP o FTP, por ejemplo, sin hablar del puerto 80, 20 o 21).

Que lo disfruten!

Sitios para profundizar:

TCSS : <http://www.psimax.co.za> – Seccion “TCSS”
CBQ.init: <http://www.sourceforge.net/projects/cbqinit>
HTB.init: <http://www.sourceforge.net/projects/htbinit>
Snitch: <http://snitch.sourceforge.net/>

©Arturo A. Busleiman 2004

e-mail: buanzo@buanzo.com.ar

Este artículo es de distribución y modificación libres; el autor mantiene el derecho de copia.