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

Uso de QoS para equilibrar tráfico por IP y limitar tráfico P2P (72996 lectures)

Per **Daniel Cabezas**, ***kput*** (http://jopi.seriousworks.net)

Creado el 24/08/2004 01:44 modificado el 26/08/2004 23:27

> *Publico aquí un mail enviado a varias comunidades inalámbricas que creo que vale también como artículo:*
>
> Hace ya bastantes meses que trabajo montando sistemas de QOS para instalaciones wireless bastante complejas. Me sorprende que nadie en las listas de correo de ninguna comunidad wireless haya propuesto sistemas para control del tráfico P2P .
>
> Estas notas requieren conocimientos medios de iptables (saberse al dedillo el digrama de paquetes del kernel, en concreto no voy a explicar porque hay que marcar paquetes en POSTROUTING, FORWARD o INPUT) y avanzados del funcionamiento de QOS, particularmente en Linux. Algun punto es muy avanzado, pero será la excepcion. Básicamente bastará con conocer el funcionamiento de schedulers PRIO y HTB, disciplinas de cola, conocer ECN y como funciona el sistema de control de congestión TCP.

Voy a ser sincero, se que hay gente que se lleva el conocimiento de la comunidad para su uso comercial. No tengo excesivos problemas con eso, pero buena parte de las horas para implementar este tipo de sistemas lo he hecho como profesional, es decir, en jornada laboral, y cobro (o debería cobrar xD) por ello. De modo que voy a dejar la "receta" al alcance de todos, no un script automático que lo haga todo sin haberse tomado la molestia de leer un poco. Cosa que además es imposible, cada instalación es un mundo, asi que ...

En fin, manos a la obra. Queremos un super script que :

a) Nos ajuste la latencia de los servicios importantes (tráfico interactivo, paquetes SYN, ssh, tráfico UDP / ICMP)
b) pueda limitar el tráfico P2P
c) limite el ancho de banda que se otorga a cada cliente POR IP (no por conexion)
d) Reparta EQUITATIVAMENTE el ancho de banda (de forma que los p2p no nos roben mas kbs por usar muchas conexiones a la vez)
e) funcione con NAT
g) limite el número máximo de conexiones concurrentes desde y hacia uno/varios host, para no saturar la tabla MAC de los router
h) Extra: pueda limitar tanto trafico entrante como saliente
i) Extra: haga accounting, para estadisticas y/o facturacion

Instrucciones de como hacer el huevo frito :

Ultimísima versión del kelmer, ahora mismo la 2.6.8.1 , no aceptaremos nada diferente :P

Visitar la cesta de la compra, http://www.netfilter.org/ -> Tenemos la ultima version de iptables aqui :

http://www.netfilter.org/files/iptables-1.2.11.tar.bz2

y del patch-o-matic aqui

http://www.netfilter.org/files/patch-o-matic-ng-20040302.tar.bz2

También tendréis que bajar el último iproute de

http://developer.osdl.org/dev/iproute2/

Descargamos todo esto y los parches de:

- ESFQ (http://fatooh.org/esfq-2.6/),
- IMQ (http://www.linuximq.net o http://pupa.da.ru/imq/),
- WRR (http://wipl-wrr.sourceforge.net/) ,
- IPP2P (http://rnvs.informatik.uni-leipzig.de/ipp2p/index_en.html) y layer-7 (http://l7-filter.sourceforge.net/), estos dos ultimos intercambiables, recomiendo el segundo porque la firma suele bastante mas correcta para determina los protocolos p2p.

Parcheamos el kernel, iptables e iproute2 todo a base de bien con los diferentes parches de cada proyecto. Una forma más facil es usar la colección de parches qnet (http://kem.p.lodz.pl/~peter/qnet/) , que trae casi todo.

Nota: el parche de IMQ de pupa.da.ru es mas estable y se puede usar para ingress y en la cadena INPUT de iptables. Además, el WRR no funciona bien para cualquier otra versión del qnet, estáis avisados.

Explicación rápida de que-es-cada-cosa :

- Patch-o-matic-ng, patch-o-matic new generation es la nueva versión del script de la gente del proyecto netfilter para parchear automáticamente el kernel y el programa de control del firewall, iptables, con diferentes mejoras, correción de errores y (esto es lo que nos interesa) parches experimentales.

- IMQ, Intermediate Queueing device, o dispositivo de cola intermedio, es un dispositivo virtual de red, que nos permite hacer un "hook" de nuestras reglas. Dicho de modo sencillo, es un dispositivo al que podemos enviar paquetes marcados para crear colas y limitar cuanto trafico puede llegar al dispositvo real (la tarjeta de red o modem).

- ESFQ, Enhanced Stochastic Fairness queueing discipline, o disciplina estocástica mejorada de imparcialidad. A grosso modo, es una disciplina de cola para nuestro algoritmo de scheduling para que al llenarse una cola de prioridad, el ancho de banda se reparta equitativamente entre todas las conexiones.

- WRR, Weighted Round Robin, o Round Robin por peso, un scheduler con objetivo parecidos al de ESFQ, aunque a nivel de gestión de las colas. También permite poner mas "peso" a una ip concreta (mas ancho de banda) y crear jerarquias balanceadas de trafico.

- IPP2P y layer-7 netfilter. Aunque layer7 tenga muchas mas posibilidades, las resumo en que ambos sistemas permiten crear "matches" o correspondencias de un paquete en iptables de acuerdo al protocolo de este paquete. Nosotros lo usaremos para marcar tráfico p2p (emule, kazaa, bittorrent...).

Bueno, ahora vamos a ver como se consigue cada uno de los puntos :

a) Ajustar la latencia de los servicios importantes (tráfico interactivo, paquetes SYN, ssh, tráfico UDP / ICMP) :

- Esto es lo más sencillo de hacer que no expolicaré porque viene en casi cualquier tutorial: podemos usar marcado de paquetes con iptables y luego esa marca para que el tc los reconzoca, o comernos la cabeza y usar la horrible sintaxis de tc para determinar que paquetes van a cada cola. Al crear la cola padre de todas las demás ajustamos CEIL, el "techo" o velocidad máxima, a un ~80% de la capacidad de nuestra línea, para evitar que se creen colas en el ISP que den al traste con nuestra latencia.

Ejemplo:

#tc filter add dev ethX parent 1:0 protocol ip prio 1 handle 1 fw classid 1:10

#iptables -t mangle -A FORWARD -p tcp -m tcp --dport 22 -j MARK --set-mark 0x1

b) limitar el tráfico p2p y g) limitar el número máximo de conexiones concurrentes desde y hacia uno/varios host:

- Aquí se llega por varias aproximaciones. Entre los parches incluidos en el patch-o-matic, nos detendremos en los nombres connlimit y dstlimit.

Con el connlimit tenemos una correspondencia que nos permite restringir el número de conexiones TCP paralelas a un host.

Ejemplo:

#limitamos a 2 conexiones paralelas por ssh
#que vayan a cualquier ip de fuera

iptables --t filter -A FORWARD -s 192.168.0.0/24 -o eth1 -p tcp --syn --dport 22 -m connlimit --connlimit-above 2 -j REJECT --reject-with tcp-reset

o

#limitamos las conexiones a la propia maquina (para evitar DOS)

iptables -I INPUT -p tcp --dport 80 -m connlimit --connlimit-above 2 -j REJECT --reject-with tcp-reset

y con el dstlimit lo propio pero para una ip de destino concreta.

Y por el otro, marcamos los paquetes p2p mediante las correspondencias de los parches ipp2p o layer 7 para ponerlos en prioridad minima.

Ejemplo:

marca 0x3 del fw para identificar AIM

iptables -t mangle -A FORWARD -m layer7 --l7proto aim -j MARK --set-mark 3

Odiamos kazaa a muerte

iptables -t mangle -A FORWARD -p tcp -m ipp2p --kazaa -j DROP

c) limitar el ancho de banda que se otorga a cada cliente POR IP y
d) Reparta EQUITATIVAMENTE el ancho de banda.

Aqui llega un hueso duro de roer, que requiere conocimientos sobre ECN ,
Explicit Congestion Notification y RED, Random Early Detection, mecanismos para el trafico tcp para determinar congestion de un router y hacer DROP de conexiones de forma aleatoria hasta ajustar el trafico. Aunque parezca

mentira, le mejor forma de controlar el ancho de banda que nos consume el trafico de red es, directamente, descartandolo! Pero no descartandolo todo, ojo. El mecanismo de congestion de TCP reduce el mismo la velocidad de transmision hasta que no haya perdidas, y el truco consiste en saber discernir que trafico podemos descartar. Los paquetes SYN, de establecimiento de conexion por ejemplo, no podemos descartarlos. Ni los ACK, ni nada que no sea TCP (una vez nos ha llegado, el trafico UDP ya nos ha consumido el ancho de
banda, si lo descartamos no hemos ganado nada).

Por otro lado, tenemos a nuestros amigos ESFQ y WRR. La mejora que presenta ESFQ ante otras disciplinas de cola, es que este reparto se puede hacer en base a la dirección IP de la conexión. Hasta ahora ese reparto se hacía siempre por conexión; es decir, tres clientes conectados con el mismo número de conexiones reciben el mismo ancho de banda. Pero los programas p2p se aprovechan de este sistema aumentando el número de conexiones concurrentes, de forma que de esos tres clientes, uno tendria 200 conexiones abiertas mientras que los otros dos solo una respectivamente, consumiendo la mayor parte del ancho de banda.

Asi que tenemos dos posibles aproximaciones al problema : marcar el trafico TCP susceptible de ser descartado y hacer uso de RED (deteccion de la congestion) para irlo liberando, o usar ESFQ para limitar equitativamente por IP. La experiencia dice que la primera aproximacion es la mas efectiva, sobre todo cuando hay mucho trafico p2p de por medio. Un pseudo-ejemplo podria ser :

```
tc qdisc add dev eth0 parent 1:15 handle 1000: wrr dest ip ip.del.cliente 0
tc qdisc change handle 1000: dev eth0 wrr qdisc wmode1=3 wmode2=0


# Y añadir una cola RED bajo nuestro scheduler :

tc qdisc add dev eth0 parent 1000:1 handle 1000: red
bandwidth 200k probability 0.02
limit $RED_LIMIT min $RED_MIN max $RED_MAX avpkt 800
burst $[((2*$RED_MIN)+($RED_MAX))/(3*800)+1] ecn

# Marcamos nuestro trafico tcp descartable

iptables -t mangle -A FORWARD -p tcp -m tcp --tcp-flags ! SYN,RST,ACK ACK -j MARK --set-mark 15

# dirigimos el trafico marcado a la cola

tc filter add dev eth0 parent 1: prio 1 protocol ip handle 15 fw flowid 1:15
```

h) Pueda limitar tanto trafico entrante como saliente . Esto es una pesadilla para mucha gente de la lista de lartc (www.lartc.org). ya que el filtrado INGRESS en Linux esta muy mal documentado y casi nadie sabe usarlo. Aqui llega en nuestro rescate el dispositivo intermedio de cola, nuestro amigo IMQ . Como se explica a si mismo, pongo un par de ejemplos para dar por acabado el tutorial :

```
# usando imq para limitar las descargas a 64 kbit en la gateway NAT
tc qdisc add dev imq0 handle 1: root htb default 1
tc class add dev imq0 parent 1: classid 1:1 htb rate 64kbit
tc qdisc add dev imq0 parent 1:1 handle 2 sfq
iptables -t mangle -A POSTROUTING -o eth0 -j IMQ --todev 0

# esto en la gateway, con destino la ip 80.61.123.212,
# no deja llegar losdatos a mas velocidad

iptables -t mangle -A FORWARD -i eth0 -d 80.61.123.212 -j IMQ
```

Con lo cual podemos limitar no solo la descarga sino tambien la subida de datos. Ojo que el IMQ no hace mas que encolar trafico para ajustar el ancho de banda, el ajuste optimo es usarlo en conjuncion con descarte de paquetes que he explicado mas arriba.

el apartado i) es parte de lo que no explicare por lo que he dicho al
principio del artículo

Bueno, bastante informacion por hoy, espero que pueda ser de ayuda para todos aquellos que se pelean con multiples usuarios conectados a su nodo.

P.D. : lo de "nodo" es porque el mensaje original iba dirigido a instalaciones wireless :P

Saludos, Dani

E-mail del autor: daniel _ARROBA_ seriousworks.net