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

Algunas experiencias con Balanceo de carga, bonding, y demás (57776 lectures)

Per **J Nicolas Castellano**, ***Nico*** (http://www.noconname.org)

Creado el 12/05/2003 21:45 modificado el 12/05/2003 21:59

Este es el relato de una experiencia que tuve en la party de Zaragoza (Redaton 2003[1]), montando un tinglado con un Linux que balancea la carga con 20 adsl's a 300 usuarios y haciendo bonding... También explico algunas soluciones a problemas que nos encontramos con las tarjetas de red y los switches Cisco catalyst.

Experiencia que tuve en la party de Zaragoza (Redaton 2003[1]), junto a LaraCroft, la mente de la parte *Linux+networking* en la party, digna de admirar, por cierto. (¡Mujeres!, debéis tomar ejemplo de ella, jeje).

Introducción

Era una party en la que fueron entre 300 y 400 personas y os puedo asegurar que lo que montamos fue espeluznante, ahora os cuento...

El que diseñó la red lo hizo en estrella, donde el centro era un Cisco Catalyst 4006 y en cada rama Catalyst 3500. En el centro de la estrella iban los servidores de juegos con Linux. Además de esto, por cuestiones ajenas, tuvimos que montárnoslo con una salida a Internet de 20 ADSL's de 2 MB (*20 * 2 = 40 Mbits/s o 5 Mbytes/s pero no son reales*, que conste que telefonica solo certifica el 10%).

Entonces, al ver esto, a lara se le ocurrió la brillante idea de utilizar balanceo de carga, (exactamente, reparto equitativo del trafico de Internet entre las 20 ADSL's) con un Linux, de forma que la máquina con Linux sea el *gateway* a Internet.

A parte de esto, quisimos prevenir en salud, y en vez de poner 2 tarjetas de red (una de tráfico para la red de participantes y otra a la red de los routers ADSL), pusimos 4, aprovechando el ancho de banda de las 2 tarjetas de red como si en realidad se comportasen como una sola (eth0 y eth1 como bond0, y lo mismo para eth2 y eth3 como bond1). Lo pusimos, no porque el ancho de banda no bastase (40 Mbps < 100 Mbps), sino porque el "*bonding*" (que es así como se llama en Linux) tiene la ventaja de que si una de las tarjetas de red deja de funcionar, automáticamente (con algo de retardo) la otra pasa a ser independiente como una tarjeta normal.

Además de esto tuvimos algunos problemas con los switches catalyst interactuando con las tarjetas de red Realtek y algunas 3Com (se ve que no se llevan muy bien y no acaban de negociar), por lo que hubo que forzar las tarjetas a *Half* o *Full duplex* (según preferencia) desde el Catalyst y luego forzarlas desde los Linux con '`mii-tool`'. En los Windows se forzaron desde un diskete que hay que lo hace por hardware.

Montamos 2 subredes lógicas, una clase C donde estaban los routers ADSL (192.168.1.x) de la .1 a la .20 y en el .254 el gateway Linux, y otra con clase A del tipo 10.x.x.x donde estaban todos los participantes (jeje, un poco exagerado). El Linux tenía 2 ips, bond0 (192.168.1.254 y 10.0.0.1).
Los routers ASDL tenían una ruta estática para poder encaminar a la red 10.x.x.x sin tener que fastidiar al Linux haciendo NAT.

Desgraciadamente, el canuto a Internet era INSUFICIENTE (los 40Mbps _NO_REALES_), por lo que Lara hizo algo de priorización de paquetes con CBQ; desgraciadamente no dispongo del script ya que todo fue a parar a /dev/null jeje.....

Empezamos con la parte Teórica

El Gateway era un Linux/Redhat 8.0, y aunque al final casi se me suicida Lara porque el paquete iproute2 venia con bugs, pudimos solucionarlo.

El balanceo que elegimos, no es nada más que un reparto equitativo del tráfico de salida a las puertas de enlace (en nuestro caso 20), por lo que desde la versión 2.4.15 del kernel de Linux, llevan incorporado la opción "*equal cost multi path*" (Advanced Router-Networking Options), que deberemos activar, junto un parche de Julian Anastov [2]al igual que *ip_conntrack* de Netfilter[3] por si luego queremos hacer movidas con IPTABLES (SNAT,DNAT,etc), Policy Routing por si queremos hacer tablas para decidir qué paquetes deben ir por un sitio u otro, rutas estáticas vamos, (las tablas no las hemos usamos para nada) y activar el *forwarding*.

Hecho esto, compilamos el kernel, y bajamos el paquete de iproute2 (ojo!, el paquete de iproute2 de la redhat8 no funciona bien y muestra el mensaje "dead link pervasive" (conexion muerta penetrante), por lo cual no va bien a la hora de tomar decisiones.
Entonces nos bajamos el paquete más reciente(en la fecha que lo hicimos nosotros tuvimos que bajarnos el tarball porque en .rpm no habia otro).

Bien, **¿como funciona esto?** Aquí esta el quid.
Se supone que hay otros métodos para hacerlo, pero según he leido, al recibir las tramas en el camino de retorno se harían por una sola IP o ruta, con lo cual sería algo lenta la transferencia, también he leido cosas acerca de hacerlo con DNAT, pero me parece a mi que todo eso que he visto planteado no hace reparto equitativo entre las diferentes lineas externas, sino que directamente asigna rutas estáticas, esta "solución" que se planteó es hacer rutas dinámicas, y os expongo una pequeña explicación de como funcionan las cosas:

- Si la máquina 10.0.0.2 quiere llegar a Internet, ha de formar un paquete primero, lo que hace cualquier programa es utilizar bind(), éste consulta al sistema de rutas del sistema operativo acerca de que "ip de origen debe coger", --| p.ej. si yo hago ping a 127.0.0.1, la ip de origen la tomará como 127.0.0.1, pero si hago ping a 192.168.1.6, me tomará como ip de orígen 192.168.1.254, ha quedado claro creo...
- Entonces, cuando llega a 10.0.0.1 (Gateway con Linux) nuestro sistema de rutas dinámico decide qué camino ha de coger mediante un algoritmo bastante raro me parece a mi.. por lo que pude leer del kernel, él va asignando conexiones a las distintas pasarelas, una por una (pim pam), y si por lo que fuera por una pasarela se mueren conexiones, el propio sistema se encarga de asignar nuevas conexiones por esa pasarela (supongo que para que haya igual número de conexiones en todas las rutas).
  Lo que no logré comprender (quizás porque no me dio tiempo a leer más) es que realmente el kernel asignaba conexiones segun ip de origen e ip de destino, es decir, desde 10.0.0.2 podia hacer 10 conexiones (cada una a diferente ip) y el kernel me encaminaba a la misma pasarela ADSL, pero vamos... será la parte que no me leí X-D.
- Siguiendo el hilo... se acciona el mecanismo de forwarding (si lo tenemos activado, claro) y hace que el paquete vaya al router adsl que corresponda. El router hace NAT hacia el router de la intranet del ISP y luego a Internet...
- Para el retorno de los paquetes no hay ningún problema porque cuando lo hace, el kernel reconoce este paquete y sabe retornar por dónde le toca (olé con el forwarding).

Creo que hasta aquí ya he dado mucho el coñazo con la parte teórica.

Continuamos con la parte Práctica

Empezaré con la parte del bonding.
Configurar un sistema con bonding es tan sencillo como activar el modulo 'bonding.o' o bien activarlo desde el kernel (está dentro de Networking Options).

Luego, levantar las tarjetas mediante un programa en c que está dentro de Documentation/networking del kernel que deberemos compilar...

```
# cd /usr/src/Linux/Documentation/networking
```

```
# gcc ifenslave.c -o ifenslave
```
(compilamos)
```
# cp ifenslave /sbin
```

Para hacer bonding no tendremos más que hacer:
```
# modprobe bonding max_bonds=2
```
(El máximo que se puede hasta la fecha según leí)
```
# /sbin/ifenslave bond0 eth0 eth1
```
(Lo que nosotros tocaremos a partir de ahora es bond0)

Ahora el balanceo:
Para el balanceo debemos aprender a llevarnos bien 'ip', y si os salta algún kernel panic, no os preocupéis, es que Linux 2.4 aun no va muy fino con ip.

Primero debemos arrancar sin ninguna configuración de ifconfig, así que si tenéis alguna, quitadla. Lo malo es que si ya teneis el pc arrancado no se borrará de la tabla tan facilmente (es decir, haciendo ifconfig ethx down).
```
# ip addr
```
(vemos que las ips siguen ahí)

Aquí os pego un pequeño script que no es de mi invención, para borrar las que haya:
```
# d=`ip -o link show | cut -d: -f2`
for i in $d ; do
ip addr flush $i
ip link set $i down
done
unset i d
```

Una vez hecho esto, al hacer "ip addr" no nos debería dar ninguna ip por interfaz, así que empezaremos dando de alta las interfaces y las ip's.
```
# ip addr add 127.0.0.1/8 dev lo (interface loopback)
# ip link set lo up
# ip route add 127.0.0.0/8 dev lo
```

Ahora configuramos las dos interfaces que irán para los usuarios que vengan de 10.x.x.x
```
# modprobe bonding max_bonds=2
```
(cargamos el modulo)
```
# ip link set bond0 up
```
(damos de alta el bond0 como interfaz)
```
# ip addr add 10.0.0.1/8 dev bond0
```
(le asignamos una ip)
```
# ip route add 10.0.0.0/8 dev bond0
```
(le asignamos una ruta estática para 10.0.0.0)
```
# /sbin/ifenslave bond0 eth0 eth1
```
(levantamos el bonding)

Acto seguido el bond1, que son la conjunción de las dos tarjetas que van a los routers ADSL.
```
# ip link set bond1 up
# ip addr add 192.168.1.254/24 dev bond1
# ip route add 192.168.1.0/24 dev bond1
# /sbin/ifenslave bond1 eth2 eth3
```

Las rutas mágicas...
```
# ip route add default proto static\
nexthop via 192.168.1.1 dev bond1\
nexthop via 192.168.1.2 dev bond1\
nexthop via 192.168.1.3 dev bond1\
```
... (hasta 20)

Y el forwarding... (porque si no, todo lo que hemos hecho no sirve para nada)
```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Y el forzado de las tarjetas de red.

Ayuda de '`mii-tool`'

```
  usage: mii-tool [-VvRrwl] [-A media,... | -F media] [interface ...]
       -V, --version              display version information
       -v, --verbose              more verbose output
```

```
       -R, --reset                reset MII to poweron state
       -r, --restart              restart autonegotiation
       -w, --watch                monitor for link status changes
       -l, --log                  with -w, write events to syslog
       -A, --advertise=media,...  advertise only specified media
       -F, --force=media          force specified media technology
media: 100baseT4, 100baseTx-FD, 100baseTx-HD, 10baseT-FD, 10baseT-HD,
       (to advertise both HD and FD) 100baseTx, 10baseT
```

Creo que está bien claro: Al hacer '`mii-tool`' a secas, te dice que 'media' está funcionando, y a qué estado ha negociado (o no). En el caso de los Catalysts, habría que forzar las tarjetas a lo que uno desee.
En nuestro caso los servers tuvieron que forzarse a 100baset, full duplex:
`# mii-tool -F 100baseTx-FD eth0` (por ejemplo).

Y creo que eso es todo lo que tenia que contaros; lo demás son problemas no relacionados con los Linux, y que tienen importancia, pero lo he visto publicado en Bulma con anterioridad, así que no le doy mucha importancia.

Notas

- Para saber esto, nos leimos el Nano-HOWTO, que explica pasito a pasito como hacerlo, lo que pasa es que el gacho del howto se enreda un monton y opino que, no explica muy claramente todos los conceptos y al final puedes hacerte una empanada mental, por ejemplo, con las tablas (en nuestro caso no necesitabamos tablas para nada).
- Cuidado con el bonding. Para hacer bonding, al otro lado ha de haber bonding también (o en nuestro caso etherchannel con los cisco catalyst), tened cuidado con el tipo de etherchannel que hagáis porque según me han comentado hay varios tipos de etherchannel, y no todos son compatibles con bonding.
- La opción "proto static" del balanceo es muy importante, si alguna ruta deja de funcionar, automáticamente es descartada por el kernel, y a los 60 segundos (por defecto) vuelve a intentar con la tabla de rutas que tiene, este intervalo lo podeis tocar a través de /proc/sys/net/ipv4/route/gc_interval.
- Yo lo tengo claro, pero por si alguien no tiene muchos conocimientos de networking quiero que se entienda: el balanceo no puede nunca coger las 20 ADLSL's para una conexión, es decir, no podemos coger una trama, dividirla en 20 y enviar cada una por cada ADSL, porque sino habria fragmentos que no tendrían cabecera, y en el supuesto caso de que las tuvieran, el destino no sabria como reensamblarlas al retornar desde ip's diferentes (habría que hacer movidas no contempladas, como crear un nuevo protocolo en una capa superior a IP y decirle que cuando los paquetes retornen se reensamblen por diferentes ip's, y eso creo que no está contemplado.)
- Para cercioraros de que el balanceo vaya bien, haced un 'ip route ls cache' y si tenéis algún problema con alguna ruta antigua 'ip route flush cache' y os borrará completamente las rutas que esté tomando en ese momento.
- No os preocupéis si os pegan kernel panics, en kernel 2.4 es normal, en 2.5 dicen que va fino, ¿es verdad?
- Las conexiones SSH con clientes Linux no funcionaban, en cambio con windows sí, supongo que el sistema de funcionamiento con las ips seria diferente, pero no me lo miré mucho.

Agradecimientos

Quiero agradecer a Pci y a deluxe_ por aclararme muchos conceptos que no tenia claro sobre cómo enfrentar el problema del balanceo, a LaraCroft por aguantar ahí siempre, y buscar soluciones para todo cuando creíamos que no iba a funcionar nada (que el mérito no es mío sino suyo), y a toda la gente de Zaragoza que es muy maja y se portaron estupendamente bien conmigo.

Sugerencias

Si tenéis algún comentario, alguna sugerencia o algo en lo que creáis que haya metido la pata, decídmelo y lo corrijo, os estaré eternamente agradecido :-)

Hasta otro artículo.

Lista de enlaces de este artículo:

1. http://www.redaton.com
2. http://www.ssi.bg/~ja/#routes
3. http://www.netfilter.org

E-mail del autor: jncastellano _ARROBA_ noconname.org