

ÍNDICE

1. Escenario.....	2
1.1. Redes y conexiones.....	2
1.2. Cliente	2
1.3. Balanceador de carga.....	3
1.4. Servidor	3
1.5. Conectividad	4
2. Balanceo de carga	5
2.1. Fichero haproxy.cfg	5
2.1.1. Estructura general	5
2.1.2. Parámetros globales	5
2.1.3. Parámetros proxy	5
2.2. Estudio de comportamiento a través de IP física	7
2.2.1. Estudio con algoritmo alternativo: source.....	10
3. Alta disponibilidad	11
3.1. Ficheros keepalived-lb1.conf y keepalived-lb2.conf	11
3.2. Análisis del tráfico.....	13
3.2.1. Comportamiento del escenario con alta disponibilidad.....	13
3.2.2. Parada de keepalive en el balanceador principal	16
3.2.3. Prueba de ping a virtual router en caída de interfaz	19

1. Escenario

El escenario de despliegue requiere del 2 clientes, 2 balanceadores y 3 servidores, que serán definidos y contruidos en el fichero *docker-compose.yml*. El fichero se puede dividir en tres partes: *version*, *services* y *networks*. En nuestro caso usaremos *version 3*, en el bloque *services* definiremos como construir los contenedores que lanzaremos. Por último, la sección *networks* las interfaces que existirán en nuestro entorno. Para ello, mostraremos lo más representativo de cada uno de los tipos de componentes de nuestro despliegue que se encuentran en el fichero.

1.1. Redes y conexiones

La siguiente imagen nos muestra las dos interfaces de red que vamos a montar:

```
networks:
  client_net:
    driver: bridge
    ipam:
      config:
        - subnet: 176.3.3.0/24
    driver_opts:
      com.docker.network.bridge.name: client_net
  server_net:
    driver: bridge
    ipam:
      config:
        - subnet: 180.2.5.0/24
    driver_opts:
      com.docker.network.bridge.name: server_net
```

1.2. Cliente

Ambos clientes son idénticos salvo el nombre del contenedor. Por lo que nos basta con mostrar uno de ellos.

```
h1:
  build: base/cliente
  hostname: h1
  cap_add:
    - SYS_ADMIN
    - NET_ADMIN
  depends_on:
    - "lb1"
  networks:
    client_net:
      ipv4_address: 176.3.3.204
```

Los clientes pertenecen a la red *client_net* definida en el apartado anterior. Los siguientes campos indican lo mismo para los diferentes contenedores independientemente de su función. El campo *build* nos indica el directorio dónde obtener la inicialización del contenedor. Además de la base dada en el directorio */base/cliente* se ha añadido un *bash* en el que se configura el nuevo gateway que tendrá el cliente, en esta primera sección será la ruta estática del balanceador lb1 176.3.3.200, aunque posteriormente se cambie por una VIP. Otro campo destacado es el de *depends_on* que establece que el contenedor no será creado si el contenedor lb1 no está ejecutándose.

```
#!/bin/bash
ip route del default dev eth0
ip route add default via 176.3.3.200
tail -f /dev/null
```

Contenido del fichero */base/cliente/routes.sh*

1.3. Balanceador de carga

El balanceador es el contenedor que más requerimientos necesita para el arranque.

```
lb1:
  build: base/haproxy
  volumes:
    - ./base/haproxy:/tmp
  hostname: lb1
  cap_add:
    - SYS_ADMIN
    - NET_ADMIN
  networks:
    client_net:
      ipv4_address: 176.3.3.200
  server_net:
    ipv4_address: 180.2.5.100
```

Además de los campos mencionados anteriormente, el balanceador incluye el campo *volumes*, que nos permite cargar */base/haproxy* dentro del contenedor en el directorio */tmp*. Los ficheros de configuración del balanceador incluyen *haproxy.cfg* y el archivo de configuración del keepalive, de los cuales hablaremos posteriormente.

1.4. Servidor

```
s1:
  hostname: s1
  build: base/servidor
  ports:
```

```
- "9000:80"
cap_add:
- SYS_ADMIN
- NET_ADMIN
depends_on:
- "lb1"
networks:
server_net:
ipv4_address: 180.2.5.101
```

Además de los ya nombrados campos, destaca en este caso *ports*, puerto dónde se realizarán las peticiones que serán HTTP. El servidor también tiene en su inicializador un fichero *routes.sh*, que será análogo a la que podemos encontrar en el cliente.

1.5. Conectividad

Se ha realizado ping desde un host y de un balanceador a lo más representativo del escenario. Obteniendo resultado favorable.

```
root@h1:/# ping 176.3.3.210
PING 176.3.3.210 (176.3.3.210) 56(84) bytes of data.
64 bytes from 176.3.3.210: icmp_seq=1 ttl=64 time=0.083 ms
64 bytes from 176.3.3.210: icmp_seq=2 ttl=64 time=0.102 ms
64 bytes from 176.3.3.210: icmp_seq=3 ttl=64 time=0.104 ms
^C
--- 176.3.3.210 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.083/0.096/0.104/0.012 ms
root@h1:/# ping 176.3.3.200
PING 176.3.3.200 (176.3.3.200) 56(84) bytes of data.
64 bytes from 176.3.3.200: icmp_seq=1 ttl=64 time=0.282 ms
64 bytes from 176.3.3.200: icmp_seq=2 ttl=64 time=0.098 ms
64 bytes from 176.3.3.200: icmp_seq=3 ttl=64 time=0.101 ms
^C
--- 176.3.3.200 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2053ms
rtt min/avg/max/mdev = 0.098/0.160/0.282/0.086 ms
root@h1:/# ping 176.3.3.205
PING 176.3.3.205 (176.3.3.205) 56(84) bytes of data.
64 bytes from 176.3.3.205: icmp_seq=1 ttl=64 time=0.273 ms
64 bytes from 176.3.3.205: icmp_seq=2 ttl=64 time=0.126 ms
^C
--- 176.3.3.205 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1030ms
rtt min/avg/max/mdev = 0.126/0.199/0.273/0.074 ms
root@h1:/# ping 180.2.5.101
PING 180.2.5.101 (180.2.5.101) 56(84) bytes of data.
64 bytes from 180.2.5.101: icmp_seq=1 ttl=63 time=0.317 ms
64 bytes from 180.2.5.101: icmp_seq=2 ttl=63 time=0.151 ms
^C
--- 180.2.5.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1014ms
rtt min/avg/max/mdev = 0.151/0.234/0.317/0.083 ms
root@h1:/#

alumno@lab27:~/Descargas/Practica2ARA/enunciado/enunciado$ docker exec -it c398507e2d9d /bin/bash
root@lb1:/# ping 176.3.3.204
PING 176.3.3.204 (176.3.3.204) 56(84) bytes of data.
64 bytes from 176.3.3.204: icmp_seq=1 ttl=64 time=0.081 ms
^C
--- 176.3.3.204 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.081/0.081/0.081/0.000 ms
root@lb1:/# ping 180.2.5.101
PING 180.2.5.101 (180.2.5.101) 56(84) bytes of data.
64 bytes from 180.2.5.101: icmp_seq=1 ttl=64 time=0.131 ms
64 bytes from 180.2.5.101: icmp_seq=2 ttl=64 time=0.091 ms
^C
--- 180.2.5.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1013ms
rtt min/avg/max/mdev = 0.091/0.111/0.131/0.020 ms
root@lb1:/#
```

2. Balanceo de carga

2.1. Fichero haproxy.cfg

2.1.1. Estructura general

El formato del fichero de configuración de un HAProxy viene dado por 3 bloques de parámetros principalmente: los argumentos de línea de comandos, que serán de mayor prioridad que el resto; la sección *global* del fichero, que establece los parámetros de todo el proceso; y por último, las secciones de proxy, que en nuestro caso serán *defaults*, *frontend*, *backend* y *listen*. La sintaxis del archivo consiste en líneas que comienzan con un parámetro, palabra reservada y ya definida.

2.1.2. Parámetros globales

En la siguiente imagen se muestra la sección *global* de nuestro fichero *haproxy.cfg*.

```
global
    log localhost local0
    log localhost local1 notice
    stats timeout 30s
```

El parámetro **log** agrega un servidor syslog global. Este parámetro se puede usar hasta un máximo de dos veces. Aquí nos crea un servidor en nuestra dirección. El parámetro **stats timeout** nos indica que el cuando se cree el socket de estadísticas el tiempo de espera preestablecido es de 10s, con esta opción lo cambiamos a 30s.

2.1.3. Parámetros proxy

2.1.3.1. Defaults

En esta sección se encuentran los parámetros que se establecen como predeterminados para el resto de secciones que se declaren a partir de su declaración.

```
defaults
    log global
    mode http
    option httplog
    option dontlognull
    retries 3
    timeout connect 5s
    timeout client 15s
    timeout server 15s
```

Destacar el campo **mode** definido para ser http. Varios campos nos indican los timeouts de respuesta antes de cerrar conexión, así como los intentos.

2.1.3.2. Frontend

En esta sección se describen los parámetros que modifican las conexiones aceptadas de clientes en los sockets que están escuchando.

```
frontend www.mybalancer.haproxy
  bind *:80
  mode http
  default_backend web_servers
```

En nuestro caso, se aceptan conexiones a través del puerto 80, http.

2.1.3.3. Backend

Por otro lado, aquí se describen las políticas de reenvío de conexiones entrantes entre el conjunto de servidores a los que se conecta el proxy.

```
backend web_servers
  balance roundrobin
  option httpchk HEAD /
  option forwardfor
  option http-server-close
  cookie SERVERUSED insert indirect nocache
  http-request set-header X-Forwarded-Port %[dst_port]
  server webserver0 180.2.5.101:80 cookie webserver0 check
  server webserver1 180.2.5.102:80 cookie webserver1 check
  server webserver2 180.2.5.103:80 cookie webserver2 check
```

Parámetros interesantes como el del algoritmo de balanceo, en esta muestra se usa RoundRobin. Así como los servidores disponibles para el balanceo. Cabe destacar, que aquí ya definimos las cookies que le asignaremos a cada servidor. Otro parámetro importante es añadir en los http-request, los campos X-Forwarded-Port y For. El último, lo usaremos por defecto y no requiere de entrada en el fichero. A *port*, lo hemos modificado para darle nuestro formato.

2.1.3.4. Listen

Por último, en esta sección describimos el proxy completo con su frontend y backend combinados. Generalmente es útil para el tráfico TCP-only.

```
listen stats
  bind *:8080
  stats enable
  stats uri /
  stats refresh 5s
```

2.2. Estudio de comportamiento a través de IP física

Una vez estudiado el fichero, vamos a lanzar el demonio HAProxy usando el fichero haproxy.cfg. El comando usado es `haproxy -f -D /tmp/haproxy.cfg`. Que es donde lo depositamos como hemos indicado en la sección primera. El protocolo se basa en HTTP/TCP, como vamos a ver a continuación:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000169989	180.2.5.100	180.2.5.103	HTTP	85	HEAD / HTTP/1.0
5	0.000203937	180.2.5.103	180.2.5.100	TCP	66	80 → 60006 [ACK] Seq=1 Ack=20 Win=65152 Len=0 TSval=334970928 TSecr=3795757220
6	0.000697164	180.2.5.103	180.2.5.100	HTTP	340	HTTP/1.1 200 OK
7	0.000734224	180.2.5.100	180.2.5.103	TCP	66	60006 → 80 [ACK] Seq=20 Ack=275 Win=64128 Len=0 TSval=3795757221 TSecr=334970929
8	0.000797858	180.2.5.100	180.2.5.103	TCP	66	60006 → 80 [FIN, ACK] Seq=20 Ack=275 Win=64128 Len=0 TSval=3795757221 TSecr=334970929
9	0.000865847	180.2.5.100	180.2.5.103	TCP	66	60006 → 80 [RST, ACK] Seq=21 Ack=275 Win=64128 Len=0 TSval=3795757221 TSecr=334970929
10	0.671794696	180.2.5.100	180.2.5.101	TCP	74	44262 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=713602822 TSecr=0 WS=128
11	0.671871532	180.2.5.101	180.2.5.100	TCP	74	80 → 44262 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=354833021 TSecr=713602822
12	0.671911772	180.2.5.100	180.2.5.101	TCP	66	44262 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=713602822 TSecr=3548330217
13	0.671960934	180.2.5.100	180.2.5.101	HTTP	85	HEAD / HTTP/1.0
14	0.671994542	180.2.5.101	180.2.5.100	TCP	66	80 → 44262 [ACK] Seq=1 Ack=20 Win=65152 Len=0 TSval=3548330217 TSecr=713602822
15	0.672546236	180.2.5.101	180.2.5.100	HTTP	340	HTTP/1.1 200 OK
16	0.672589539	180.2.5.100	180.2.5.101	TCP	66	44262 → 80 [ACK] Seq=20 Ack=275 Win=64128 Len=0 TSval=713602822 TSecr=3548330217
17	0.672708844	180.2.5.100	180.2.5.101	TCP	66	44262 → 80 [FIN, ACK] Seq=20 Ack=275 Win=64128 Len=0 TSval=713602823 TSecr=3548330217
18	0.672767502	180.2.5.101	180.2.5.100	TCP	66	80 → 44262 [FIN, ACK] Seq=275 Ack=21 Win=65152 Len=0 TSval=3548330218 TSecr=713602823
19	0.672781911	180.2.5.100	180.2.5.101	TCP	66	44262 → 80 [RST, ACK] Seq=21 Ack=275 Win=64128 Len=0 TSval=713602823 TSecr=3548330217
20	0.672801091	180.2.5.100	180.2.5.101	TCP	54	44262 → 80 [RST] Seq=21 Win=0 Len=0
21	1.360695104	180.2.5.100	180.2.5.102	TCP	74	33378 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1236733570 TSecr=0 WS=128
22	1.360798292	180.2.5.102	180.2.5.100	TCP	74	80 → 33378 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2389879880 TSecr=713602822

▶ Frame 13: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 1
 ▶ Ethernet II, Src: 02:42:b4:02:05:64 (02:42:b4:02:05:64), Dst: 02:42:b4:02:05:65 (02:42:b4:02:05:65)
 ▶ Internet Protocol Version 4, Src: 180.2.5.100, Dst: 180.2.5.101
 ▶ Transmission Control Protocol, Src Port: 44262, Dst Port: 80, Seq: 1, Ack: 1, Len: 19
 ▶ Hypertext Transfer Protocol
 ▶ HEAD / HTTP/1.0\r\n
 \r\n
 [HTTP request 1/1]
 [Response in frame: 15]

Comunicación entre balanceador servidor, mantener conexión

En la anterior traza, el balanceador establece conexión TCP con los distintos servidores, en concreto, las trazas [10-18] lo muestran con el servidor 180.2.5.101, con el que una vez establecida la conexión envía un HTTP request HEAD, para comprobar que el servidor sigue activo. Este ejemplo nos servirá para ver lo que pasará a continuación en la comunicación con el cliente.

En este escenario, nos es indiferente el balanceador que usemos ya que no tienen relación entre ellos todavía y sería mostrar información redundante. Por lo que vamos a realizar todo el estudio a través del balanceador con dirección IP 176.3.3.200 en *client_net* y 180.2.5.100 en *server_net*. Análogamente, con probar el compartimiento con el host 1, nos servirá para realizar este estudio.

Comencemos lanzando el navegador web *lynx* a través de la línea de comandos con dirección IP física del balanceador, la 176.3.3.200. Las trazas son las siguientes:

http.request.method == GET						
No.	Time	Source	Destination	Protocol	Length	Info
186	10.883529398	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
206	10.883813588	180.2.5.100	180.2.5.102	HTTP	375	GET / HTTP/1.0
324	17.338804415	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
334	17.339080471	180.2.5.100	180.2.5.103	HTTP	375	GET / HTTP/1.0
451	23.348151948	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
461	23.348413253	180.2.5.100	180.2.5.101	HTTP	375	GET / HTTP/1.0

[Calculated window size: 64256]	
[Window size scaling factor: 128]	
Checksum: 0x68c2 [unverified]	
[Checksum Status: Unverified]	
Urgent pointer: 0	
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps	
[SEQ/ACK analysis]	
[Timestamps]	
TCP payload (257 bytes)	

Hypertext Transfer Protocol	
GET / HTTP/1.0\r\n	
Host: 176.3.3.200\r\n	
Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01\r\n	
Accept-Encoding: gzip, compress, bzip2\r\n	
Accept-Language: en\r\n	
User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17\r\n\r\n	
[Full request URI: http://176.3.3.200/]	
[HTTP request 1/1]	
[Response in frame: 189]	

0135	0f d9 40 00 40 06	c2 4f b0 03 03 cc b0 03	55 00 00 00 00 00
0020	03 c8 cf 1a 00 50 56 e7	d9 95 7a df 68 ea 80 18	00 00 00 00 00 00
0030	01 f6 68 c2 00 00 01 01	08 0a e9 2b 37 35 3a 2a	00 00 00 00 00 00
0040	2a d9 4f 45 54 20 2f 20	48 54 54 50 2f 31 2e 30	00 00 00 00 00 00
0050	0d 0a 48 6f 73 74 3a 20	31 37 36 2e 33 2e 33 2e	00 00 00 00 00 00
0060	32 30 30 0d 0a 41 63 63	65 70 74 3a 20 74 65 78	00 00 00 00 00 00
0070	74 2f 68 74 6d 6c 2c 20	74 65 78 74 2f 70 6c 61	00 00 00 00 00 00
0080	69 6e 2c 20 74 65 78 74	2f 73 67 6d 6c 2c 20 74	00 00 00 00 00 00
0090	65 78 74 2f 63 73 73 2c	20 61 70 70 6c 69 63 61	00 00 00 00 00 00
00a0	74 69 6f 6e 2f 78 68 74	6d 6c 2b 78 6d 6c 2c 20	00 00 00 00 00 00
00b0	2a 2f 2a 3b 71 3d 30 2e	30 31 0d 0a 41 63 63 65	00 00 00 00 00 00
00c0	70 74 2d 45 6e 63 6f 64	69 6e 67 3a 20 67 7a 69	00 00 00 00 00 00
00d0	70 2c 20 63 6f 6d 70 72	65 73 73 2c 20 62 7a 69	00 00 00 00 00 00
00e0	70 32 0d 0a 41 63 63 65	70 74 2d 4c 61 6e 67 75	00 00 00 00 00 00
00f0	61 67 65 3a 20 65 6e 0d	0a 55 73 65 72 2d 41 67	00 00 00 00 00 00
0100	65 6e 74 3a 20 4c 79 6e	78 2f 32 2e 38 2e 39 64	00 00 00 00 00 00
0110	65 76 2e 31 36 20 6c 69	62 77 77 77 2d 46 4d 2f	00 00 00 00 00 00
0120	32 2e 31 3a 20 53 53 4c	2d 4d 4d 2f 31 2e 34 2e	00 00 00 00 00 00
0130	31 20 47 4e 55 54 4c 53	2f 33 2e 35 2e 31 37 0d	00 00 00 00 00 00
0140	0a 0d 0a		

En la imagen, filtramos las solicitudes GET, para ver más limpiamente la secuencia de peticiones que se realiza como consecuencia de la petición del cliente al balanceador de acceso a la web con dirección física IP 176.3.3.200, esto quiere decirnos que, a ojos del cliente, no existe el balanceador, si no un proveedor de contenido web. La primera petición que hemos realizado corresponde con los primeros mensajes. En el primer GET de la traza, el cliente solicita al balanceador como si fuera un servidor web. Tras la petición, el balanceador establece conexión con el servidor con IP 180.2.5.102. ¿Por qué con este servidor y no con otro? Como hemos visto en el apartado anterior, usamos el algoritmo Round Robin, que ha determinado el servidor final al que realizamos la petición. La primera petición, la que va de cliente a balanceador no depara ningún campo HTTP novedoso. Sin embargo, en la siguiente imagen, se muestran los campos de la petición del balanceador al servidor, donde aparecen campos, mencionados en el fichero *haproxy.cfg*, como son el X-Forwarded-For y X-Forwarded-Port, que nos indican el puerto de la petición y la dirección origen de la petición más allá del balanceador.

No.	Time	Source	Destination	Protocol	Length	Info
186	10.883529398	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
208	10.883813538	180.2.5.100	180.2.5.102	HTTP	375	GET / HTTP/1.0
324	17.339804415	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
334	17.33980471	180.2.5.100	180.2.5.103	HTTP	375	GET / HTTP/1.0
451	23.348151948	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
461	23.348413253	180.2.5.100	180.2.5.101	HTTP	375	GET / HTTP/1.0

Checksum: 0x742a [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (309 bytes)
Hypertext Transfer Protocol
GET / HTTP/1.0\r\n
Host: 176.3.3.200\r\n
Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01\r\n
Accept-Encoding: gzip, compress, bzip2\r\n
Accept-Language: en\r\n
User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17\r\n
X-Forwarded-Port: 80\r\n
X-Forwarded-For: 176.3.3.204\r\n
\r\n
[Full request URI: http://176.3.3.200/]
[HTTP request 1/1]
[Response in frame: 208]

0040	d1 81 47 45 54 20 2f 20	48 54 54 50 2f 31 2e 30	-GET / HTTP/1.0
0050	0d 0a 48 6f 73 74 3a 20	31 37 36 2e 33 2e 33 2e	-Host: 176.3.3.
0060	32 30 30 0d 0a 41 63 63	65 70 74 3a 20 74 65 78	200-Accept: tex
0070	74 2f 68 74 6d 6c 2c 20	74 65 78 74 2f 70 6c 61	t/html, text/pla
0080	69 6e 2c 20 74 65 78 74	2f 73 6f 6d 6c 2c 20 74	in, text /sgml, t
0090	65 78 74 2f 63 73 73 2c	20 61 70 70 6c 69 63 61	ext/css, applica
00a0	74 69 6f 6e 2f 78 68 74	6d 6c 2b 78 6d 6c 2c 20	tion/xhtml+xml,
00b0	2a 2f 2a 3b 71 3d 30 2e	30 31 0d 0a 41 63 63 65	*/*;q=0.01-Accep
00c0	70 74 2d 45 6e 63 6f 64	69 6e 67 3a 20 67 7a 69	pt-Encoding: gzi
00d0	70 2c 20 63 6f 6d 70 72	65 73 73 2c 20 62 7a 69	p, compress, bzi
00e0	70 32 0d 0a 41 63 63 65	70 74 2d 4c 61 6e 67 75	p2-Accept-Langu
00f0	61 67 65 3a 20 65 6e 0d	0a 55 73 65 72 2d 41 67	age: en-User-Ag
0100	65 6e 74 3a 20 4c 79 6e	78 2f 32 2e 38 2e 39 64	ent: Lynx/2.8.9d
0110	65 76 2e 31 36 20 6c 69	62 77 77 77 2d 46 4d 2f	ev.16 libwww-FM/
0120	32 2e 31 34 20 53 53 4c	2d 4d 4d 2f 31 2e 34 2e	2.14 SSL-MM/1.4.
0130	31 20 47 4e 55 54 4c 53	2f 33 2e 35 2e 31 37 0d	1 GNUTLS /3.5.17-
0140	0a 58 2d 46 6f 72 77 61	72 64 65 64 2d 50 6f 72	-X-Forwarded-Port
0150	74 3a 20 38 30 0d 0a 30	2d 4d 4d 2f 31 2e 34 2e	t: 80-X-Forwarded
0160	65 64 2d 46 6f 72 3a 20	31 37 36 2e 33 2e 33 2e	80-For: 176.3.3.
0170	32 30 34 0d 0a 0d 0a		204-...

Finalmente, se inicia el proceso inverso, es decir, enviar la información solicitada al cliente. Proporcionando la página. Y dando una cookie, dónde se indica el servidor de dónde la ha obtenido por el Round Robin, esto se puede ver en la imagen siguiente. Otro campo en el que nos debemos fijar es en el de Connection, con valor close, es decir, de momento, las conexiones no son persistentes con los servidores.

El balanceador no persiste con el mismo servidor. Si solo son operaciones de consulta esto no debe suponer un problema, pero si los servidores requieren de guardar estados del cliente, se debe mantener persistencia entre servidores, como veremos en el apartado 3.

Volviendo a la imagen anterior, vemos que cada vez que llega una petición de página al balanceador, este realiza una petición de página a un servidor distinto, debido al algoritmo RoundRobin, el cual, hace turnos de servidores y a cada petición que le llega se la asigna al siguiente servidor del turno de acuerdo a unos pesos. Además es ideal para servidores que tengan misma función, ya que reparte el tiempo a partes iguales.. En la siguiente subsección, veremos el comportamiento con otro algoritmo, denominado en la jerga de HAProxy como source.

Content-Encoding: gzip\r\n
 Content-Length: 3138\r\n
 Connection: close\r\n
 Content-Type: text/html\r\n
 Set-Cookie: SERVERUSED=webserver1; path=/\r\n
 Cache-control: private\r\n
 \r\n
 [HTTP response 1/1]
 [Time since request: 0.001549971 seconds]
 [Request in frame: 186]
 [Request URI: http://176.3.3.200/]
 Content-encoded entity body (gzip): 3138 bytes -> 10918 bytes
 File Data: 10918 bytes
 Line-based text data: text/html (375 lines)
 \n
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\n
 <html xmlns="http://www.w3.org/1999/xhtml">\n
 <!--\n
 Modified from the Debian original for Ubuntu\n
 Last updated: 2016-11-16\n
 See: https://launchpad.net/buns/1288690\n

Frame (677 bytes) Reassembled TCP (3507 bytes) Uncompressed entity body (10918 bytes)

2.2.1. Estudio con algoritmo alternativo: source

A diferencia del Round Robin, source usa el IP física origen, la del cliente, para asignar un servidor. Esto va a hacer que el mismo cliente siempre acabe en el mismo servidor, ya que se aplica el método de tabla hash, donde se calcula un número a partir de su IP, para luego dividirlo entre el número de servidores. Solo viendo las peticiones GET que se realizan veremos que efectivamente se verifica lo expuesto:

No.	Time	Source	Destination	Protocol	Length	Info
209	11.466405620	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
226	11.466876695	180.2.5.100	180.2.5.103	HTTP	375	GET / HTTP/1.0
379	19.471798704	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
380	19.472047427	180.2.5.100	180.2.5.103	HTTP	375	GET / HTTP/1.0
503	25.599078487	176.3.3.204	176.3.3.200	HTTP	323	GET / HTTP/1.0
510	25.599194651	180.2.5.100	180.2.5.103	HTTP	375	GET / HTTP/1.0

▶ Frame 226: 375 bytes on wire (3000 bits), 375 bytes captured (3000 bits) on interface 1
 ▶ Ethernet II, Src: 02:42:b4:02:05:64 (02:42:b4:02:05:64), Dst: 02:42:b4:02:05:67 (02:42:b4:02:05:67)
 ▶ Internet Protocol Version 4, Src: 180.2.5.100, Dst: 180.2.5.103
 ▶ Transmission Control Protocol, Src Port: 47108, Dst Port: 80, Seq: 1, Ack: 1, Len: 309
 ▶ Hypertext Transfer Protocol
 ▶ GET / HTTP/1.0\r\n
 Host: 176.3.3.200\r\n
 Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01\r\n
 Accept-Encoding: gzip, compress, bzip2\r\n
 Accept-Language: en\r\n
 User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17\r\n
 X-Forwarded-Port: 80\r\n
 X-Forwarded-For: 176.3.3.204\r\n
 \r\n
 [Full request URI: http://176.3.3.200/]
 [HTTP request 1/1]
 [Response in frame: 228]

0000 02 42 b4 02 05 67 02 42 b4 02 05 64 08 00 45 00 B...g.B...d...E...
 0010 01 69 84 97 40 00 40 06 42 28 b4 02 05 64 b4 02 .1...@.B(...d...
 0020 05 67 b8 04 00 50 17 6e 3e 8e c2 f5 24 e6 80 18 .g...P.n>...\$...
 0030 01 f6 74 2b 00 00 01 01 08 0a e2 ba 62 7c 14 72 .t+...b|r...
 0040 fa 08 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 30 .GET / HTTP/1.0
 0050 0d 0a 48 6f 73 74 3a 20 31 37 36 2e 33 2e 33 2e .Host: 176.3.3.

3. Alta disponibilidad

3.1. Ficheros keepalived-lb1.conf y keepalived-lb2.conf

Para obtener alta disponibilidad y no depender de que falle un balanceador, vamos a tener varios routers, dos balanceadores en nuestro caso, que se enmascaran bajo una misma IP a la que llamaremos Virtual IP, como si fueran un único router, a lo que llamaremos Virtual Router. Para conseguir esto vamos a usar el protocolo VRRP. En particular, hay dos tipos de routers dentro del Virtual Router, el *master*, que es el que ejerce la función de router y *slave*, que en caso de fallo del *master*, pasa a cumplir la función de este.

El fichero *keepalived-lb1.conf* será la implementación del *master router*. El fichero *keepalived-lb2.conf* hará la de *slave* o *backup* en la nomenclatura de la herramienta *keepalive*, instalada, que usará los ficheros para obtener el nuevo escenario. Veamos más detenidamente los parámetros del fichero:

```
global_defs {  
    notification_email {  
        admin@example.com  
    }  
    notification_email_from noreply@example.com  
    smtp_server 127.0.0.1  
    smtp_connect_timeout 60  
}  
  
vrrp_sync_group VG1 {  
    group {  
        RH_EXT  
        RH_INT  
    }  
}  
  
vrrp_instance RH_EXT {  
    state MASTER  
    interface eth0  
    virtual_router_id 50  
    priority 150  
    advert_int 2  
    authentication {  
        auth_type PASS  
        auth_pass passw123  
    }  
    virtual_ipaddress {  
        176.3.3.220  
    }  
}  
  
vrrp_instance RH_INT {  
    state MASTER  
    interface eth1  
    virtual_router_id 2  
    priority 150  
    advert_int 2
```

```
authentication {
    auth_type PASS
    auth_pass passw123
}
virtual_ipaddress {
    180.2.5.220
}
}
```

Imagen correspondiente *keepalived-lb1.conf*

Así, el fichero de configuración posee una estructura de bloques, jerarquizados. Cada bloque se centra y tiene como objetivo una característica específica del demonio.

De esta manera, el primer bloque y más importante es el de definiciones globales, *global_defs*. Aquí hemos definido, parámetros de notificación de correo, además de datos de su servidor SMTP.

Los siguientes bloques lo representan las configuraciones del protocolo VRRP. Primero definimos un bloque *vrrp_sync_group*. Hemos definido los nombre de los siguientes bloques, que serán las VIP de cada interfaz del balanceador. Crear el grupo busca la sincronización entre las instancias que creemos de VRRP.

Las instancias, precisamente, son la clave para la configuración del protocolo. Definidas como *vrrp_instance*, el primer parámetro que hallamos en ella es *state*, en el caso de la imagen anterior, el valor es master, pero en el fichero del esclavo será BACKUP. Se define la interfaz sobre la que se va a aplicar la VIP, así como su dirección VIP. Otro parámetro destacado es el *virtual_router_id*, usado para diferenciar virtual routers con la misma configuración. Entre los esclavos, es importante el campo *priority*, el cual, quien tenga mayor prioridad será *master*, en caso de que haya que elegir a un nuevo master. Cabe destacar, el campo *advert_int*, que nos indica el intervalo de VRRP advert en segundos. Por último, el subbloque *authentication* nos permite pedir contraseña con texto plano o con IPSEC, aunque no se recomienda esta opción, además, el [RFC 3768] eliminó este campo para VRRPv2.

3.2. Análisis del tráfico

3.2.1. Comportamiento del escenario con alta disponibilidad

Veamos ahora como se modifica el escenario del apartado anterior. En esta sección vamos a comprobar la aparición de los protocolos IGMP y VRRP al escenario, dónde ARP pasará a ser tener un papel más relevante del que tenía en apartados anteriores, sin quitar importancia a lo que ya venía realizando sobre la red.

Tras activar el primer balanceador, comunica a la dirección Broadcast de las dos interfaces a través del protocolo ARP que en para su dirección MAC es dónde se responde a la virtual IP 176.3.3.220 por una interfaz y 180.2.5.220 para la otra. Además, el balanceador con su IP comienza a la lanzar VRRP Announcements, con los que también indica que sigue teniendo él la potestad sobre la dirección virtual. Como podremos ver en la siguiente imagen.

No.	Time	Source	Destination	Protocol	Length	Info
463	25.356600444	180.2.5.100	180.2.5.103	TCP	54	38862 → 80 [RST] Seq=21 Win=0 Len=0
464	25.850960630	02:42:b0:03:03:c8	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request)
465	25.851197913	02:42:b0:03:03:c8	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request)
466	25.851320689	02:42:b0:03:03:c8	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request)
467	25.851440624	02:42:b0:03:03:c8	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request)
468	25.851558414	02:42:b0:03:03:c8	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request)
469	25.851620774	176.3.3.200	224.0.0.18	VRRP	54	Announcement (v2)
470	25.852114836	02:42:b4:02:05:64	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request)
471	25.852316641	02:42:b4:02:05:64	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request)
472	25.852428679	02:42:b4:02:05:64	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request)
473	25.852546821	02:42:b4:02:05:64	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request)
474	25.852651083	02:42:b4:02:05:64	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request)
475	25.852701762	180.2.5.100	224.0.0.18	VRRP	54	Announcement (v2)
476	26.022397263	180.2.5.100	180.2.5.101	TCP	74	36210 → 80 [SYN] Seq=0 Win=64240 Len=0 M

> Frame 469: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
 > Ethernet II, Src: 02:42:b0:03:03:c8 (02:42:b0:03:03:c8), Dst: IPv4mcast_12 (01:00:5e:00:00:12)
 > Internet Protocol Version 4, Src: 176.3.3.200, Dst: 224.0.0.18
 ∨ Virtual Router Redundancy Protocol
 > Version 2, Packet type 1 (Advertisement)
 Virtual Rtr ID: 50
 Priority: 150 (Non-default backup priority)
 Addr Count: 1
 Auth Type: Simple Text Authentication [RFC 2338] / Reserved [RFC 3768] (1)
 Adver Int: 2
 Checksum: 0x06b1 [correct]
 [Checksum Status: Good]
 IP Address: 176.3.3.220
 Authentication String: passw123

Lb1 se activa

La dirección de destino de los paquetes VRRP siempre será la del multicast 224.0.0.18. Siguiendo en esta línea, entra en escena el protocolo IGMP, a través del cual el balanceador se va añadir al grupo multicast de los balanceadores que están bajo la misma virtual_router_id. De hecho esto ocurre antes que lo mencionado anteriormente.

387	21.862513697	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources
389	21.978500521	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources
1407	60.230498182	176.3.3.210	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources
1408	60.418499651	176.3.3.210	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources
4834	155.862492841	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Leave group 224.0.0.18
4873	156.722498359	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Leave group 224.0.0.18
5585	175.454509011	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources
5612	176.206499558	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources
7597	229.982509438	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources
7623	230.350474818	176.3.3.200	224.0.0.22	IGMPv3	54 Membership Report / Join group 224.0.0.18 for any sources

Frame 1408: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
 Ethernet II, Src: 02:42:b0:03:03:d2 (02:42:b0:03:03:d2), Dst: IPv4mcast_16 (01:00:5e:00:00:16)
 Internet Protocol Version 4, Src: 176.3.3.210, Dst: 224.0.0.22
 Internet Group Management Protocol

[IGMP Version: 3]
 Type: Membership Report (0x22)
 Reserved: 00
 Checksum: 0xf9eb [correct]
 [Checksum Status: Good]
 Reserved: 0000
 Num Group Records: 1
 Group Record : 224.0.0.18 Change To Exclude Mode
 Record Type: Change To Exclude Mode (4)
 Aux Data Len: 0
 Num Src: 0
 Multicast Address: 224.0.0.18

IGMP reports

El balanceador internamente registra todos estos procedimientos en su fichero log interno.

```
root@lb1:/# keepalived -I -D -n -f /tmp/keepalived-lb1.conf
Fri Dec 13 19:04:38 2019: Starting Keepalived v1.3.9 (10/21,2017)
Fri Dec 13 19:04:38 2019: Opening file '/tmp/keepalived-lb1.conf'.
Fri Dec 13 19:04:38 2019: Starting Healthcheck child process, pid=19
Fri Dec 13 19:04:38 2019: Starting VRRP child process, pid=20
Fri Dec 13 19:04:38 2019: Opening file '/tmp/keepalived-lb1.conf'.
Fri Dec 13 19:04:38 2019: Registering Kernel netlink reflector
Fri Dec 13 19:04:38 2019: Registering Kernel netlink command channel
Fri Dec 13 19:04:38 2019: Registering gratuitous ARP shared channel
Fri Dec 13 19:04:38 2019: Opening file '/tmp/keepalived-lb1.conf'.
Fri Dec 13 19:04:38 2019: VRRP_Instance(RH_EXT) removing protocol VIPs.
Fri Dec 13 19:04:38 2019: VRRP_Instance(RH_INT) removing protocol VIPs.
Fri Dec 13 19:04:38 2019: Using LinkWatch kernel netlink reflector...
Fri Dec 13 19:04:38 2019: VRRP sockpool: [ifindex(52), proto(112), unicast(0), fd(9,10)]
Fri Dec 13 19:04:38 2019: VRRP sockpool: [ifindex(56), proto(112), unicast(0), fd(11,12)]
Fri Dec 13 19:04:40 2019: VRRP_Instance(RH_EXT) Transition to MASTER STATE
Fri Dec 13 19:04:40 2019: VRRP_Instance(RH_INT) Transition to MASTER STATE
Fri Dec 13 19:04:42 2019: VRRP_Instance(RH_EXT) Entering MASTER STATE
Fri Dec 13 19:04:42 2019: VRRP_Instance(RH_EXT) setting protocol VIPs.
Fri Dec 13 19:04:42 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:04:42 2019: VRRP_Instance(RH_EXT) Sending/queueing gratuitous ARPs
on eth0 for 176.3.3.220
Fri Dec 13 19:04:42 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:04:42 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:04:42 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:04:42 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
```

```
Fri Dec 13 19:04:42 2019: VRRP_Group(VG1) Syncing instances to MASTER state
Fri Dec 13 19:04:42 2019: VRRP_Instance(RH_INT) Entering MASTER STATE
Fri Dec 13 19:04:42 2019: VRRP_Instance(RH_INT) setting protocol VIPs.
Fri Dec 13 19:04:42 2019: Sending gratuitous ARP on eth1 for 180.2.5.220
Fri Dec 13 19:04:42 2019: VRRP_Instance(RH_INT) Sending/queueing gratuitous ARPs
on eth1 for 180.2.5.220
```

Log de lb1 cuando arranca el demonio

Tras activar el demonio, el primer paso es registrarse en el grupo multicast y llevar a cabo la sincronización con los miembros del grupo. Como solo está el lb1, adquiere el master y envía los paquetes ARP ya mencionados.

Una vez creado el grupo de sincronización con la pertenencia del primer balanceador, veamos como reacciona el escenario a la entrada en el grupo de un segundo router que desempeñará la función de BACKUP por defecto y con una prioridad inferior a la del master.

Arrancamos el demonio en este segundo balanceador, produciendo paquetes IGMP análogos al del primero, para entrar en el grupo de sincronización multicast. Lo interesante se halla en el log del balanceador.

```
root@lb2:/# keepalived -l -D -n -f /tmp/keepalived-lb2.conf
Fri Dec 13 19:05:16 2019: Starting Keepalived v1.3.9 (10/21,2017)
Fri Dec 13 19:05:16 2019: Opening file '/tmp/keepalived-lb2.conf'.
Fri Dec 13 19:05:16 2019: Starting Healthcheck child process, pid=20
Fri Dec 13 19:05:16 2019: Opening file '/tmp/keepalived-lb2.conf'.
Fri Dec 13 19:05:16 2019: Starting VRRP child process, pid=21
Fri Dec 13 19:05:16 2019: Registering Kernel netlink reflector
Fri Dec 13 19:05:16 2019: Registering Kernel netlink command channel
Fri Dec 13 19:05:16 2019: Registering gratuitous ARP shared channel
Fri Dec 13 19:05:16 2019: Opening file '/tmp/keepalived-lb2.conf'.
Fri Dec 13 19:05:16 2019: VRRP_Instance(RH_EXT) removing protocol VIPs.
Fri Dec 13 19:05:16 2019: VRRP_Instance(RH_INT) removing protocol VIPs.
Fri Dec 13 19:05:16 2019: Using LinkWatch kernel netlink reflector...
Fri Dec 13 19:05:16 2019: VRRP_Instance(RH_EXT) Entering BACKUP STATE
Fri Dec 13 19:05:16 2019: VRRP_Instance(RH_INT) Entering BACKUP STATE
Fri Dec 13 19:05:16 2019: VRRP sockpool: [ifindex(54), proto(112), unicast(0), fd(9,10)]
Fri Dec 13 19:05:16 2019: VRRP sockpool: [ifindex(58), proto(112), unicast(0), fd(11,12)]
```

Log de lb2 cuando arranca el demonio

Se puede observar que tras realizar los primeros pasos de sincronización, entra a las instancias de VRRP como esclavo, puesto que ya hay un maestro y además al tener por defecto ser esclavo no intentará ser el nuevo maestro.

Una vez desplegado el escenario realizamos una consulta HTTP a la virtual IP 176.3.3.220 creada para dar alta disponibilidad.

Time	Source	Destination	Protocol	Length	Info
4163	137.450091453	180.2.5.103	HTTP	340	HTTP/1.1 200 OK
4178	137.812488000	180.2.5.101	HTTP	340	HTTP/1.1 200 OK
4191	138.123946470	180.2.5.101	HTTP	340	HTTP/1.1 200 OK
4203	138.480330757	180.2.5.102	HTTP	340	HTTP/1.1 200 OK
4215	138.774746691	176.3.3.204	HTTP	323	GET / HTTP/1.0
4217	138.775047204	180.2.5.100	HTTP	375	GET / HTTP/1.0
4221	138.775828760	176.3.3.204	HTTP	677	HTTP/1.1 200 OK (text/html)
4222	138.775698702	180.2.5.102	HTTP	3506	HTTP/1.1 200 OK (text/html)
4235	138.783385203	180.2.5.102	HTTP	340	HTTP/1.1 200 OK
4246	139.154902893	180.2.5.103	HTTP	340	HTTP/1.1 200 OK
4255	139.451962636	180.2.5.103	HTTP	340	HTTP/1.1 200 OK
4267	139.813539045	180.2.5.101	HTTP	340	HTTP/1.1 200 OK
4280	140.125782347	180.2.5.101	HTTP	340	HTTP/1.1 200 OK
4292	140.482419653	180.2.5.102	HTTP	340	HTTP/1.1 200 OK


```

frame 4215: 323 bytes on wire (2584 bits), 323 bytes captured (2584 bits) on interface 0
ethernet II, Src: 02:42:b0:03:03:cc (02:42:b0:03:03:cc), Dst: 02:42:b0:03:03:c8 (02:42:b0:03:03:c8)
Internet Protocol Version 4, Src: 176.3.3.204, Dst: 176.3.3.220
Transmission Control Protocol, Src Port: 33496, Dst Port: 80, Seq: 1, Ack: 1, Len: 257
Hypertext Transfer Protocol
  GET / HTTP/1.0\r\n
    Host: 176.3.3.220\r\n
    Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01\r\n
    Accept-Encoding: gzip, compress, bzip2\r\n
    Accept-Language: en\r\n
    User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17\r\n
    \r\n
    [Full request URI: http://176.3.3.220/]
    [HTTP request 1/1]
    [Response in frame: 4221]

```

En la anterior imagen se realiza la consulta HTTP y se aplica el algoritmo de balanceo Round Robin visto en el apartado anterior. Lo destacado del comportamiento de esta consulta es, que mientras el virtual router recibe el tráfico a través de la virtual IP de la interfaz eth0 y responde al cliente a través de ella, que a la que se ha pedido la consulta. No se realiza esta simetría cuando el virtual router tiene que realiza la consulta a los servidores, y en vez de realizarla a través de la dirección virtual 180.2.5.220 la realiza como si el router maestro la realizase el, sin vincularse con la virtual IP de esa interfaz a pesar de que él es el master.

En las siguientes secciones veremos como se realiza esta sincronización y como intervienen los balanceadores esclavos en situaciones de salida del router maestro, ya sea una situación porque se quiere parar la función de alta disponibilidad en él o por una situación adversa como puede ser la caída de una de las interfaces de red.

3.2.2. Parada de keepalive en el balanceador principal

Ahora comprobaremos como reacciona el protocolo ante la salida controlada del balanceador master. En la imagen de trazas de IGMP hemos apreciado dos tipos de mensajes, de entrada al grupo y de salida. Pues bien, cuando hacemos una salida controlada, la solicitud de salirse del grupo de multicast en el que se haya el balanceador se genera. Por otro lado, todo el entorno es consciente de la salida voluntaria del master, cuando paramos el demonio. En la siguiente imagen se muestra el log del balanceador maestro y como se gestiona el cambio en el interior del virtual router:


```
Fri Dec 13 19:04:47 2019: Sending gratuitous ARP on eth1 for 180.2.5.220
^CFri Dec 13 19:06:52 2019: VRRP_Instance(RH_EXT) sent 0 priority
Fri Dec 13 19:06:52 2019: Stopped
Fri Dec 13 19:06:52 2019: VRRP_Instance(RH_EXT) removing protocol VIPs.
Fri Dec 13 19:06:52 2019: Stopping
Fri Dec 13 19:06:52 2019: VRRP_Instance(RH_INT) sent 0 priority
Fri Dec 13 19:06:52 2019: VRRP_Instance(RH_INT) removing protocol VIPs.
Fri Dec 13 19:06:53 2019: Stopped
Fri Dec 13 19:06:53 2019: Stopped Keepalived v1.3.9 (10/21,2017)
```

Log de lb1 en el momento de la parada del demonio

Es destacado como el balanceador cambia su prioridad a 0, quitándose toda opción de coger el master del virtual router, de modo que otro router sea el encargado de balancear a partir de ese momento.

```
Fri Dec 13 19:06:52 2019: VRRP_Instance(RH_INT) Transition to MASTER STATE
Fri Dec 13 19:06:52 2019: VRRP_Group(VG1) Syncing instances to MASTER state
Fri Dec 13 19:06:52 2019: VRRP_Instance(RH_EXT) Transition to MASTER STATE
Fri Dec 13 19:06:53 2019: VRRP_Instance(RH_EXT) Entering MASTER STATE
Fri Dec 13 19:06:53 2019: VRRP_Instance(RH_EXT) setting protocol VIPs.
Fri Dec 13 19:06:53 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:06:53 2019: VRRP_Instance(RH_EXT) Sending/queueing gratuitous ARPs
on eth0 for 176.3.3.220
Fri Dec 13 19:06:53 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:06:53 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:06:53 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:06:53 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:06:54 2019: VRRP_Instance(RH_INT) Entering MASTER STATE
Fri Dec 13 19:06:54 2019: VRRP_Instance(RH_INT) setting protocol VIPs.
```

Log de lb2 en el momento de la parada del demonio

La trama de mensajes del log, nos muestra que al router le ha llegado un mensaje de transición a estado maestro, si hubieran más esclavos aparte de lb2, se impondría el de mayor prioridad. Aquí, lb2 se convierte en maestro y esto se hará patente en los siguientes VRRP Announcements, dónde se ahora la IP 176.3.3.210 responde para la VIP 176.3.3.220.

4832	155.843358430	176.3.3.200	224.0.0.18	VRRP	54 Announcement (v2)
4861	156.457930879	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
4896	157.064884870	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
4970	159.065701645	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5041	161.066260499	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5137	163.067074265	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5215	165.067895991	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5286	167.068709597	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5361	169.069528075	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5431	171.070344158	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5503	173.071188083	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5572	175.072009543	176.3.3.210	224.0.0.18	VRRP	54 Announcement (v2)
5658	177.445019531	176.3.3.200	224.0.0.18	VRRP	54 Announcement (v2)

> Frame 4896: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
 > Ethernet II, Src: 02:42:b0:03:03:d2 (02:42:b0:03:03:d2), Dst: IPv4mcast_12 (01:00:5e:00:00:12)
 > Internet Protocol Version 4, Src: 176.3.3.210, Dst: 224.0.0.18
 ✓ Virtual Router Redundancy Protocol

> Version 2, Packet type 1 (Advertisement)
 Virtual Rtr ID: 50
 Priority: 100 (Default priority for a backup VRRP router)
 Addr Count: 1
 Auth Type: Simple Text Authentication [RFC 2338] / Reserved [RFC 3768] (1)
 Adver Int: 2
 Checksum: 0x38b1 [correct]
 [Checksum Status: Good]
 IP Address: 176.3.3.220
 Authentication String: passw123

VRRP Announcements

Analizando el tráfico de mensajes, hemos filtrado por el protocolo VRRP, en el que podemos observar, como se produce el cambio de IP que anuncia que tiene la potestad sobre la virtual IP. En la cabecera del protocolo se observan varios de los parámetros que anteriormente hemos introducido en los ficheros *keepalived*, como son la prioridad, el tipo de autenticación, el tiempo de advertisement o el virtual router ID.

A nivel ARP se produce en error de duplicidad de dirección IP, en la que aparecen dos MACs asignadas a la misma dirección. Esto se solventa al tiempo.

Time	Source	Destination	Protocol	Length	Info
4861.156.457930879	176.3.3.210	224.0.0.18	VRRP	54	Announcement (v2)
4873.156.722498359	176.3.3.200	224.0.0.22	IGMPv3	54	Membership Report / Leave group 224.0.0.18
4874.156.686508460	180.2.5.100	224.0.0.22	IGMPv3	54	Membership Report / Leave group 224.0.0.18
4888.157.064186142	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
4889.157.064438802	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
4890.157.064584610	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
4892.157.064699674	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
4894.157.064817646	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
4896.157.064884870	176.3.3.210	224.0.0.18	VRRP	54	Announcement (v2)
4941.158.459012953	02:42:b4:02:05:6e	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request) (duplicate use of 180.2.5.220 detected!)
4942.158.459291608	02:42:b4:02:05:6e	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request) (duplicate use of 180.2.5.220 detected!)
4943.158.459415583	02:42:b4:02:05:6e	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request) (duplicate use of 180.2.5.220 detected!)
4944.158.459540776	02:42:b4:02:05:6e	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request) (duplicate use of 180.2.5.220 detected!)
4945.158.459665160	02:42:b4:02:05:6e	Broadcast	ARP	42	Gratuitous ARP for 180.2.5.220 (Request) (duplicate use of 180.2.5.220 detected!)
4946.158.459732049	180.2.5.110	224.0.0.18	VRRP	54	Announcement (v2)
4970.159.065701645	176.3.3.210	224.0.0.18	VRRP	54	Announcement (v2)
5016.160.460355655	180.2.5.110	224.0.0.18	VRRP	54	Announcement (v2)
5041.161.066260499	176.3.3.210	224.0.0.18	VRRP	54	Announcement (v2)
5075.162.066219649	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
5077.162.066588482	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
5078.162.066718202	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
5080.162.066827762	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
5082.162.066941900	02:42:b0:03:03:d2	Broadcast	ARP	42	Gratuitous ARP for 176.3.3.220 (Request) (duplicate use of 176.3.3.220 detected!)
5092.162.460971514	180.2.5.110	224.0.0.18	VRRP	54	Announcement (v2)
5107.162.745874827	02:42:b0:03:03:d2	Broadcast	ARP	42	Who has 176.3.3.204? Tell 176.3.3.220 (duplicate use of 176.3.3.220 detected!)

Frame 4970: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
 Ethernet II, Src: 02:42:b0:03:03:d2 (02:42:b0:03:03:d2), Dst: IPv4mcast_12 (01:00:5e:00:00:12)
 Internet Protocol Version 4, Src: 176.3.3.210, Dst: 224.0.0.18
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
 Total Length: 40

3.2.3. Prueba de ping a virtual router en caída de interfaz

En este caso, vamos a realizar ping desde uno de los clientes a la dirección virtual que da soporte de alta disponibilidad. Durante el ping, el router maestro va a perder la interfaz eth0, la que conecta con los clientes. Primero veamos la sincronización que se lleva a cabo en el interior del virtual router al detectar que no se ha caído la interfaz.

```

root@lb1:~# ifconfig eth0 down
root@lb1:~# Fri Dec 13 19:07:51 2019: Kernel is reporting: interface eth0 DOWN
Fri Dec 13 19:07:51 2019: VRRP_Instance(RH_EXT) Entering FAULT STATE
Fri Dec 13 19:07:51 2019: VRRP_Instance(RH_EXT) removing protocol VIPs.
Fri Dec 13 19:07:51 2019: VRRP_Instance(RH_EXT) Now in FAULT state
Fri Dec 13 19:07:51 2019: VRRP_Group(VG1) Syncing instances to FAULT state
Fri Dec 13 19:07:52 2019: VRRP_Instance(RH_INT) Entering FAULT STATE
Fri Dec 13 19:07:52 2019: VRRP_Instance(RH_INT) removing protocol VIPs.
Fri Dec 13 19:07:52 2019: VRRP_Instance(RH_INT) Now in FAULT state

```

Log del lb1

El kernel detecta que una de las interfaces se ha caído, esto provoca un efecto cascada afectando no solo a la interfaz caída si no marcando todas las instancias VRRP para el balanceador en estado FAULT, es decir este balanceador no puede ser usado de momento. Veamos ahora como reacciona el router esclavo.

```

Fri Dec 13 19:07:53 2019: VRRP_Instance(RH_INT) Transition to MASTER STATE
Fri Dec 13 19:07:53 2019: VRRP_Group(VG1) Syncing instances to MASTER state
Fri Dec 13 19:07:53 2019: VRRP_Instance(RH_EXT) Transition to MASTER STATE
Fri Dec 13 19:07:55 2019: VRRP_Instance(RH_INT) Entering MASTER STATE
Fri Dec 13 19:07:55 2019: VRRP_Instance(RH_INT) setting protocol VIPs.
Fri Dec 13 19:07:55 2019: Sending gratuitous ARP on eth1 for 180.2.5.220

```

```

Fri Dec 13 19:07:55 2019: VRRP_Instance(RH_INT) Sending/queueing gratuitous ARPs
on eth1 for 180.2.5.220
Fri Dec 13 19:07:55 2019: Sending gratuitous ARP on eth1 for 180.2.5.220
Fri Dec 13 19:07:55 2019: Sending gratuitous ARP on eth1 for 180.2.5.220
Fri Dec 13 19:07:55 2019: Sending gratuitous ARP on eth1 for 180.2.5.220
Fri Dec 13 19:07:55 2019: Sending gratuitous ARP on eth1 for 180.2.5.220
Fri Dec 13 19:07:56 2019: VRRP_Instance(RH_EXT) Entering MASTER STATE
Fri Dec 13 19:07:56 2019: VRRP_Instance(RH_EXT) setting protocol VIPs.
Fri Dec 13 19:07:56 2019: Sending gratuitous ARP on eth0 for 176.3.3.220
Fri Dec 13 19:07:56 2019: VRRP_Instance(RH_EXT) Sending/queueing gratuitous ARPs
on eth0 for 176.3.3.220

```

Log de lb2

Nuevamente y como en la sección anterior se produce una transición a estado maestro. Como pasó como antes. Si mencionamos la imagen donde aparecen todos los paquetes IGMP, observamos que no hay una caída como la hubo en el caso de la sección anterior, ya que, a diferencia de la salida del demonio, lb1 no está saliendo del grupo sino que sigue aunque en modo FAULT y no podría coger el maestro en caso de que el actual maestro fallara.

Por otra parte, ¿qué es lo que ve el cliente cuando todo esto falla? El cliente solo va a detectar que en cierto tiempo t , una serie de paquetes de ICMP que contienen el ping no van a obtener respuesta alguna por el virtual router. Pero una vez que el esclavo ha pasado a ser maestro y a tomar el control ya es capaz de responder a los próximos paquetes ping. Esto lo podemos ver en la siguiente imagen.

Time	Source	Destination	Protocol	Length	Info
6946	213.006594570	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=12/3072, ttl=64 (reply in 6947)
6947	213.006652168	176.3.3.204	ICMP	98	Echo (ping) reply id=0x001b, seq=12/3072, ttl=64 (request in 6946)
6961	213.467405139	176.3.3.200	VRRP	54	Announcement (v2)
6985	214.030580816	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=13/3328, ttl=64 (no response found!)
6998	214.473334055	180.2.5.100	VRRP	54	Announcement (v2)
7022	215.054548247	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=14/3584, ttl=64 (no response found!)
7058	216.078600344	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=15/3840, ttl=64 (no response found!)
7070	216.478396613	180.2.5.100	VRRP	54	Announcement (v2)
7094	217.089055571	176.3.3.210	VRRP	54	Announcement (v2)
7095	217.088681366	180.2.5.110	VRRP	54	Announcement (v2)
7096	217.102563274	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=16/4096, ttl=64 (no response found!)
7132	218.126588532	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=17/4352, ttl=64 (no response found!)
7169	219.150581846	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=18/4608, ttl=64 (no response found!)
7175	219.091601256	180.2.5.110	VRRP	54	Announcement (v2)
7220	220.079221916	176.3.3.210	VRRP	54	Announcement (v2)
7222	220.174598840	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=19/4864, ttl=64 (reply in 7223)
7223	220.174670199	176.3.3.204	ICMP	98	Echo (ping) reply id=0x001b, seq=19/4864, ttl=64 (request in 7222)
7257	221.198582604	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=20/5120, ttl=64 (reply in 7259)
7258	221.091865175	180.2.5.110	VRRP	54	Announcement (v2)
7259	221.198641846	176.3.3.220	ICMP	98	Echo (ping) reply id=0x001b, seq=20/5120, ttl=64 (request in 7257)
7285	222.080394506	176.3.3.210	VRRP	54	Announcement (v2)
7298	222.222583335	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=21/5376, ttl=64 (reply in 7299)
7299	222.222630339	176.3.3.204	ICMP	98	Echo (ping) reply id=0x001b, seq=21/5376, ttl=64 (request in 7298)
7334	223.092179598	180.2.5.110	VRRP	54	Announcement (v2)
7335	223.246589032	176.3.3.204	ICMP	98	Echo (ping) request id=0x001b, seq=22/5632, ttl=64 (reply in 7337)