Práctica 6

Encaminamiento IP

Resulta fundamental para el intercambio correcto de datagramas IP la selección adecuada de la ruta hacia el destino. Tradicionalmente, y debido al principio de optimización, las reglas para la selección del encaminamiento están basadas en el método de *siguiente salto*, tomando en cuenta sólo la dirección IP del destino. Linux provee este comportamiento, pero también proporciona métodos más flexibles. Las rutas se pueden seleccionar basándose en otras características de los datagramas, y hablaremos entonces de *políticas de encaminamiento*.

En esta práctica se afianzarán los conceptos básicos de encaminamiento IP, en primer lugar mediante rutas estáticas para posteriormente estudiar uno de los protocolos de encaminamiento interno más sencillos: el RIP (*Routing Information Protocol*).

6.1. Configuración de la red

En primer lugar crearemos tres máquinas virtuales, como en la figura 6.1. Usamos `ifconfig` para configurar la interfaz `eth0` de `uml1` con una dirección de la red de clase B `172.16.0.0`:

```
# ifconfig eth0 172.16.0.1 up
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:16:2D:10:C7:7B
          inet addr:172.16.0.1  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:19 Base address:0x6000
```

Como no hemos especificado ni la máscara de subred ni la dirección de difusión, la orden `ifconfig` ha establecido los predeterminados para la clase de la red.

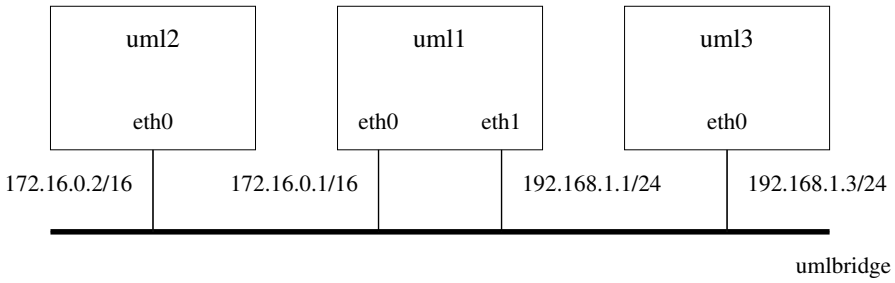

FIGURA 6.1: Configuración de las máquinas virtuales

Usamos de nuevo ifconfig para configurar la interfaz `eth1`, esta vez con una dirección de la red de clase C `192.168.1.0`:

```
# ifconfig eth1 192.168.1.1 up
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:27:3E:21:D8:8C
  inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:19 Base address:0x6000
```

En las máquinas `uml2` y `uml3` configuramos sólo una interfaz en cada una, de manera que puedan conectarse a `uml1`. Comprobar que desde `uml1` son alcanzables `uml2` y `uml3`, pero que éstas no pueden comunicarse entre sí.

6.2. Encaminamiento en Linux

Linux utiliza diversas tablas de rutas para la toma de decisiones de encaminamiento. La tabla ***cache*** se conoce también con el nombre de Tabla de Información de Reenvío (FIB, *Forwarding Information Base*). Esta tabla almacena las entradas de rutas usadas recientemente por el sistema. Es la primera que se consulta a la hora de tomar una decisión de encaminamiento. Si existe una entrada apropiada en esta tabla, el *kernel* reenviará el datagrama inmediatamente y no consultará las otras tablas.

Para visualizar el contenido de la tabla *cache* se pueden utilizar las órdenes `route`, `ip` y `netstat`:

```
# route -C -n
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
130.10.80.26    132.123.131.192 130.10.80.1           0      0        4 eth0
```

```
130.10.80.26    78.142.214.24   130.10.80.1           0      0        1 eth0
125.0.199.161   130.10.80.26    130.10.80.26    l     0      0        1 lo
130.10.80.26    86.108.83.226   130.10.80.1           0      0        0 eth0
206.46.109.58   130.10.80.26    130.10.80.26    l     0      0       14 lo
```

```
# ip route show cache
local 130.10.80.26 from 130.10.1.9 dev lo  src 130.10.80.26
    cache <local>  iif eth0
78.142.214.2 via 130.10.80.1 dev eth0  src 130.10.80.26
    cache  mtu 1420 advmss 1380 hoplimit 64
broadcast 130.10.81.255 from 130.10.81.88 dev lo  src 130.10.80.26
    cache <local,brd,src-direct>  iif eth0
80.59.234.233 via 130.10.80.1 dev eth0  src 130.10.80.26
    cache  mtu 1420 advmss 1380 hoplimit 64
```

```
# netstat -rnC
Kernel IP routing cache
Source          Destination     Gateway         Flags   MSS Window  irtt Iface
130.10.81.117   130.10.81.255   130.10.81.255   ibl       0 0          0 lo
140.90.128.70   130.10.80.26    130.10.80.26    l         0 0          0 lo
127.0.0.1       127.0.0.1       127.0.0.1       l     16436 0          0 lo
130.10.50.93    255.255.255.255 255.255.255.255 bl        0 0          0 lo
130.10.19.159   255.255.255.255 255.255.255.255 bl        0 0          0 lo
130.10.80.26    75.111.60.233   130.10.80.1            1500 0          0 eth0
```

6.3. Visualización de la tabla de rutas

La tabla de encaminamiento del kernel puede verse mediante las órdenes `netstat`, `route` o `ip`.

Consultar las opciones de la orden netstat mediante:

```
# man netstat
```

O bien:

```
# netstat --help
```

Visualizamos la tabla de rutas de la máquina `uml1` con la orden:

```
# netstat -nr
```

Obtendríamos algo parecido a:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
172.16.0.0      0.0.0.0         255.255.0.0     U         0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth1
```

Podemos comprobar, usando la orden `ping`, que desde nuestro puesto puede verse cualquier otro de cualquiera de ambas redes.

Hagamos la prueba con la red `192.168.1.0`. La máscara de red por defecto, `255.255.255.0`, permite que todos los equipos conectados al mismo dominio de difusión se vean formando parte de la misma red IP. Por ese motivo, no precisan de un encaminador para intercambiar datagramas entre sí.

6.4. Encaminamiento mediante rutas dinámicas. RIP (*Routing Information Protocol*)

La manera más sencilla de gestionar el encaminamiento es mediante tablas estáticas introducidas a mano. Sin embargo, esto sólo resulta práctico para redes muy pequeñas y que no cambien de topología a menudo, o para añadir la ruta predeterminada. Sin embargo, para sistemas con un número moderado de redes es muy difícil mantener la coherencia de las tablas de rutas estáticas y, además, son incapaces de adaptarse de manera automática a las condiciones cambiantes de la red (p.e. en el caso de fallo de alguno de los encaminadores) En estos casos deben mantenerse las tablas de rutas de manera autómatica mediante alguno de los **protocolos de encaminamiento**.

En esta práctica estudiaremos uno de los protocolos de pasarela interior (**IGP**, *Interior Gateway Protocol*) más sencillos, el RIP.

En Linux, el paquete `quagga` agrupa los demonios que gestionan los protocolos de encaminamiento más extendidos, como RIP, OSPF, RIPng, OSPFv3, BGP, IS-IS, además de un *superdemonio* denominado `zebra`, que es el encargado de reunir la información recolectada por cada uno de los otros demonios activos.

La configuración básica de `quagga` se realiza a través de su consola, `vtysh`, y se guarda de manera permantente en los archivos situados en `/etc/quagga`.

En el archivo `/etc/quagga/daemons` especificamos qué protocolos queremos emplear. En nuestro caso, sólo activaremos los demonios *zebra* y *ripd*:

```
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
isisd=no
```

6.4.1. zebra

La configuración del demonio *zebra* se guarda en el archivo `zebra.conf`:

```
cat zebra.conf
!
! Zebra configuration saved from vty
```

```
!   2007/11/28 14:13:23
!
hostname zebra
password zebra
enable password zebra
log file /var/log/zebra/zebra.log
!
interface eth0
!
interface eth1
 no ipv6 nd suppress-ra
 ipv6 nd prefix fec0:3::/64 86400 3600
 ipv6 nd ra-interval 200
 ipv6 nd ra-lifetime 1800
!
interface lo
!
!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!
line vty
!
```

A continuación veremos algunas de las opciones posibles:

hostname El nombre que queremos asignar al demonio `zebra` dentro de la máquina. No está relacionado con el *hostname* de la máquina.

password La contraseña de autenticación para el acceso mediante `telnet`.

enable password La contraseña para pasar al modo de configuración.

log file El archivo donde se guardan los mensajes de información (cuaderno de bitácora).

interface El nombre de la interfaz que participa en el encaminamiento. Cada interfaz además es configurable con distintas opciones. En el ejemplo, la interfaz `eth1` está configurada para que anuncie prefijos de autoconfiguración de IPv6.

ip route Permite definir rutas estáticas.

line vty Habilita el acceso mediante telnet (con la orden `telnet localhost zebra`).

6.4.2. ripd

El demonio `ripd` implementa el protocolo de encaminamiento RIP, versiones 1 y 2. El archivo donde se guarda la configuración es `/etc/quagga/ripd.conf`:

```
! -*- rip -*-
!
! RIPd sample configuration file
!
! $Id: ripd.conf.sample,v 1.1.1.1 2002/12/13 20:15:30 paul Exp $
!
hostname ripd
password zebra
!
! debug rip events
! debug rip packet
!
router rip
! network 11.0.0.0/8
! network eth0
! route 10.0.0.0/8
! distribute-list private-only in eth0
!
!access-list private-only permit 10.0.0.0/8
!access-list private-only deny any
!
!log file /var/log/quagga/ripd.log
!
!log stdout
```

Algunas de las opciones son:

router rip Habilita el protocolo RIP. Marca el comienzo de la sección de configuración de RIP. En esta sección se pueden especificar parámetros relativos al funcionamiento de este protocolo. A continuación se indican algunas de las opciones más comunes.

[no] network *network* Habilita o deshabilita el protocolo RIP para todas las interfaces asociadas a la dirección de *network*. Por ejemplo, si se habilita RIP para la red 172.16.1.0/24, todas las interfaces configuradas con una dirección IP entre la 172.16.1.0 y la 172.16.1.255 quedan habilitadas para RIP.

[no] network *iface* Habilita o deshabilita el envío y recepción de mensajes RIP por la interfaz *iface*.

passive-interface *iface* Deshabilita el envío por difusión y multidifusión de mensajes RIP a través de dicha interfaz. Los mensajes recibidos se procesan normalmente.

version *version* Establece la versión RIP. Puede ser 1 ó 2; de manera predeterminada es 2.

Tras la sección `router rip`, podemos configurar cada interfaz en particular, aunque no suele ser necesario.

interface *iface* Se inicia la sección de configuración específica para el interfaz *iface*. Dentro de una sección de interfaz pueden aparecer, entre otras, las siguientes opciones:

ip rip send version *version* En este caso *version* puede valer "1", "2" ó "1 2". Los mensajes se enviarán por este interfaz empleando la versión 1, la 2, o ambas, independientemente de la configuración global (definida en la sección `router rip`. En el caso de "1 2", los mensajes se envían tanto por difusión (como en la versión 1) como por multidifusión (versión 2).

[no] ip rip split-horizon Activa o desactiva el uso del mecanismo de horizonte dividido para minimizar el problema de la convergencia lenta.

Importante: Para iniciar *quagga* se emplea la orden `service quagga restart`.

6.5. Ejercicios

1. Configurar cuatro máquinas virtuales como se muestra en la figura 6.2. Se recomienda que en cada máquina se configuren las interfaces con una dirección IP de la forma <red>.<máquina>. Prestar especial atención a las máscaras de red indicadas en la figura; estamos utilizando CIDR y las máscaras de red no son las predeterminadas según la clase. Configurar `ripd` para utilizar la versión 1 del protocolo. Comprobar con `wireshark` los mensajes que se intercambian. Ver la tabla de rutas en cada máquina virtual con `ip route show`.

2. Activar el reenvío de datagramas mediante la orden `ip forwarding` desde la consola `vtysh` en todas las máquinas. Comprobar que desde cualquier máquina puede hacerse `ping` a cualquier interfaz de cualquier otra máquina. Por ejemplo, desde `uml4` puede alcanzarse la dirección de `uml2` asociada a la red `172.16.23.0`.

3. Configurar RIP para que emplee la versión 2 del protocolo. Comprobar las tablas de rutas y los mensajes que se intercambian. Destacar las diferencias entre los mensajes de la versión 1 con los de la versión 2.

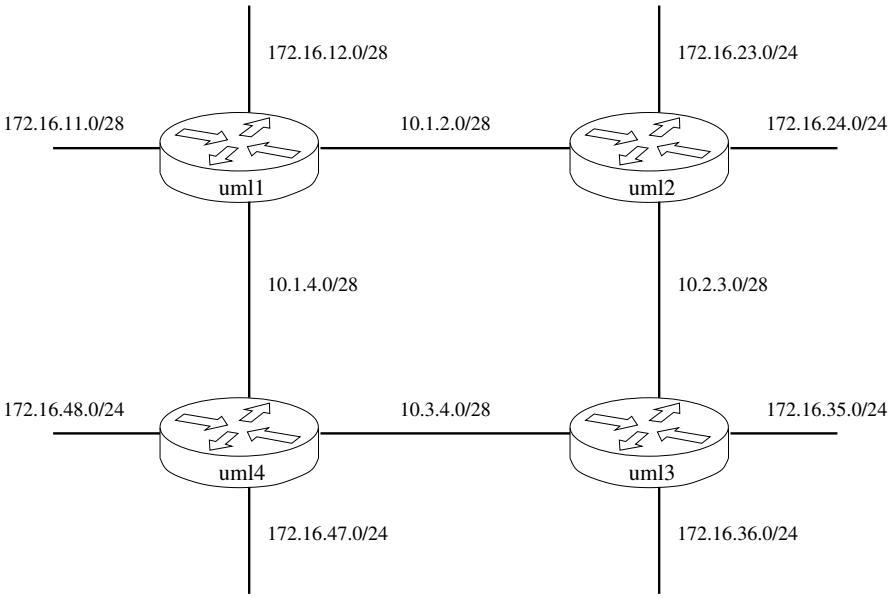

FIGURA 6.2: Configuración de las máquinas virtuales

4. Desactivar el demonio `ripd` en la máquina `uml3` mediante la orden `service quagga stop`. Comprobar cómo cambian las rutas en las demás máquinas.

5. Realizar un `ping` desde `uml2` hasta la dirección IP de la interfaz de `uml3` que la une con `uml4`. ¿Qué sucede?

6.6. Encaminamiento IPv6

Los conceptos sobre encaminamiento en IPv6 no difieren mucho de los vistos para IPv4. Las rutas se pueden definir de manera estática (en los *hosts*) o dinámica (en los encaminadores) mediante algoritmos de encaminamiento.

La configuración de las máquinas de usuario se ve facilitada por el mecanismo de autoconfiguración. Basta con que exista un encaminador IPv6 que anuncie el prefijo de red para que cualquier máquina de usuario obtenga una dirección válida. En casos más específicos, también se pueden emplear servidores DHCPv6 (*Dynamic Host Configuration Protocol*), o combinar ambos métodos.

En cuanto al encaminamiento dinámico, se han definido versiones para IPv6 de los protocolos más comunes: RIP, OSPF y BGP. La versión RIP compatible con IPv6 se denomina **RIPng** (*RIP next generation*). La versión de OSPF que soporta IPv6 es la versión 3, y BGPv4+ es compatible tanto con IPv4 como con IPv6.

6.6.1. RIP de nueva generación (RIPng)

El protocolo RIPng es una adaptación de RIP para IPv6. Por tanto, es un algoritmo de encaminamiento de vector distancia. El funcionamiento es muy similar a su predecesor RIP y adolece de sus mismos inconvenientes, como el de la convergencia lenta.

El paquete `quagga` también soporta el protocolo RIPng, y será el utilicemos en esta ocasión. La configuración se guarda en el archivo `/etc/quagga/ripngd.conf`:

```
! -*- rip -*-
!
! RIPngd sample configuration file
!
! $Id: ripngd.conf.sample,v 1.1 2002/12/13 20:15:30 paul Exp $
!
hostname ripngd
password zebra
!
! debug ripng events
! debug ripng packet
!
!
router ripng
 network eth0

!
!ipv6 access-list local-only permit 3ffe:506::/32
!ipv6 access-list local-only permit 2001:db8::/32
!ipv6 access-list local-only deny any
!
log stdout
```

Antes de configurar RIPng es necesario activar el demonio mediante la entrada correspondiente en el archivo `/etc/quagga/daemons` y reiniciar `quagga`. Mediante la orden `ipv6 forwarding` de `vtysh` debe permitirse el reenvío de datagramas IPv6.

La configuración de RIPng es muy parecida a la de RIP. Prestar mucha atención al comienzo de la sección de RIPng, que se especifica por la orden `router ripng` en lugar de `router rip`.

Ejercicio: Crear la topología de redes mostrada en la figura 6.3. Activar el protocolo RIPng en las máquinas y comprobar mediante `ping6` y `traceroute6` que todos los destinos son alcanzables. Examinar mediante `wireshark` los mensajes RIPng intercambiados y estudiar su formato. Nota: debe desactivarse el anuncio de prefijos para autoconfiguración por parte de los encaminadores, en caso de tenerlo activo del ejercicio anterior.

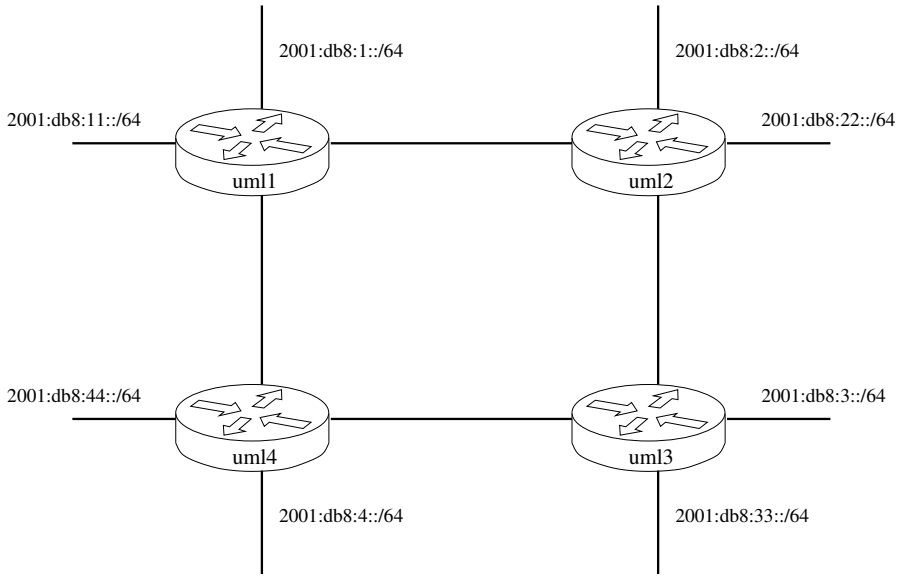

FIGURA 6.3: Configuración de las máquinas virtuales

Comprobar las rutas que ha aprendido cada máquina, su distancia y el siguiente salto. ¿Es el resultado esperado? ¿Por qué?

6.7. Encaminamiento OSPF

El algoritmo RIP, debido a las problemas de convergencia, es sólo adecuado para redes pequeñas. El protocolo más empleado como encaminamiento interior es OSPF, de la familia de protocolos de "estado del enlace".

OSPF se puede usar tanto para IPv4 (OSPFv2) como para IPv6 (OSPFv3). Es necesario activar el demonio correspondiente en el archivo `/etc/quagga/daemons`:

```
zebra=yes
ospfd=yes
ospf6d=yes
```

La configuración de ambos protocolos se guarda en los archivos `ospfd.conf` y `ospf6d.conf`, respectivamente. Para OSPFv2:

```
!
router ospf
 router-id 0.0.0.1
 passive-interface eth0
 passive-interface eth1
 network 10.1.2.0/24 area 0.0.0.0
```

```
 network 10.1.4.0/24 area 0.0.0.0
 network 172.16.1.0/28 area 0.0.0.0
 network 172.16.2.0/28 area 0.0.0.0
!
```

Y para OSPFv3:

```
!
interface eth0
 ipv6 ospf6 passive
 ipv6 ospf6 network broadcast
!
interface eth1
 ipv6 ospf6 passive
 ipv6 ospf6 network broadcast
!
interface eth2
 ipv6 ospf6 network broadcast
!
interface eth3
 ipv6 ospf6 network broadcast
!
router ospf6
 router-id 0.0.0.1
 interface eth0 area 0.0.0.0
 interface eth1 area 0.0.0.0
 interface eth2 area 0.0.0.0
 interface eth3 area 0.0.0.0
!
```

Ejercicio: Crear la topología de redes mostrada en las figuras 6.2 y 6.3. Considerando que todas las redes están en el área troncal, configurar OSPF de manera adecuada. Comprobar, además de las rutas aprendidas, las relaciones de vecindad establecidas entre los encaminadores: `show ip[v6] ospf neighbor` y la base de datos topológica: `show ip[v6] ospf database`.