Redes en GNU/Linux

La teoría de las redes llevada a la práctica en GNU/Linux

Manuel Angel Rubio Jiménez

Redes en GNU/Linux: La teoría de las redes llevada a la práctica en GNU/Linux

Manuel Angel Rubio Jiménez

Resumen

Este documento es un repaso general a los aspectos de Redes que se reflejan en las cuatro capas del modelo TCP/IP a través de los comandos de GNU/Linux típicos para poder monitorizar y/o configurar cada parte.

Tabla de contenidos

Capítulo 1. ARP

ARP son las siglas en inglés de Address Resolution Protocol (Protocolo de resolución de direcciones).

Es un protocolo de nivel de red responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP. Para ello se envía un paquete (ARP request) a la dirección de multidifusión de la red (broadcast (MAC = ff ff ff ff ff ff)) conteniendo la dirección IP por la que se pregunta, y se espera a que esa máquina (u otra) responda (ARP reply) con la dirección Ethernet que le corresponde. Cada máquina mantiene una caché con las direcciones traducidas para reducir el retardo y la carga. ARP permite a la dirección de Internet ser independiente de la dirección Ethernet, pero esto solo funciona si todas las máquinas lo soportan.

ARP está documentado en el RFC(Request For Comments) 826.

En Ethernet, la capa de enlace trabaja con direcciones físicas. El protocolo ARP se encarga de traducir las direcciones IP a direcciones MAC (direcciones físicas).Para realizar ésta conversión, el nivel de enlace utiliza las tablas ARP, cada interfaz tiene tanto una dirección IP como una dirección física MAC.

ARP se utiliza en 4 casos referentes a la comunicación entre 2 hosts:

- Cuando 2 hosts están en la misma red y uno quiere enviar un paquete a otro.
- Cuando 2 host están sobre redes diferentes y deben usar un gateway/router para alcanzar otro host.
- Cuando un router necesita enviar un paquete a un host a través de otro router.
- Cuando un router necesita enviar un paquete a un host de la misma red.

Para visualizar las direcciones ARP almacenadas en la caché de una máquina en GNU/Linux se utiliza el comando `arp`. Este comando se puede emplear como se muestra a continuación:

```
#  arp
Address         HWtype  HWaddress           Flags Mask  Iface
example         ether   00:14:22:5B:EF:22   C           eth0

# arp -n
Address         HWtype  HWaddress           Flags Mask  Iface
192.168.10.100 ether   00:14:22:5B:EF:22   C           eth0
```

Como se puede apreciar, la única dirección almacenada en la caché es la dirección de la pasarela (gateway) de la red para acceso a Internet. La pasarela hace de intermediaria entre las máquinas de Internet y la máquina que está dentro de una red privada, por lo que la única conexión de la red interna es la pasarela y otras máquinas con las que se tenga comunicación en ese momento (o se haya tenido en un corto espacio de tiempo).

Para solicitar una dirección MAC de una máquina conocida, podemos realizar un `ping` a nivel de ARP con el comando `arping`. Un ejemplo del uso del comando:

```
# arping -f xtremecore
ARPING 192.168.10.100 from 192.168.10.13 eth0
Unicast reply from 192.168.10.100 [00:14:22:5B:EF:22]  1.344ms
Sent 1 probes (1 broadcast(s))
Received 1 response(s)
```

Direcciones MAC

En redes de computadoras la dirección MAC (Media Access Control address) es un identificador hexadecimal de 48 bits que se corresponde de forma única con una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada por el IEEE (los primeros 24 bits) y el fabricante (los 24 bits restantes).

MAC opera en la capa 2 del modelo OSI, encargada de hacer fluir la información libre de errores entre dos máquinas conectadas directamente. Para ello se generan tramas, pequeños bloques de información que contienen en su cabecera las direcciones MAC correspondiente al emisor y receptor de la información.

Cada interfaz de red tiene su propia dirección MAC, esta puede apreciarse como dirección física (hardware address) en la ejecución del comando `ifconfig`:

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:13:8F:7E:14:FA
          inet addr:192.168.10.13  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::213:8fff:fe7e:14fa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50146 errors:3 dropped:0 overruns:0 frame:3
          TX packets:57827 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14041182 (13.3 MiB)  TX bytes:8144060 (7.7 MiB)
          Interrupt:209 Base address:0x6000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:229 errors:0 dropped:0 overruns:0 frame:0
          TX packets:229 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:62717 (61.2 KiB)  TX bytes:62717 (61.2 KiB)
```

Como se aprecia en el listado, el `HWaddr` (dirección hardware o física) es `00:13:8F:7E:14:FA`. Esto se da en las interfaces de tipo Ethernet (eth), pero no en la local (lo - loopback), la cual es una interfaz *virtual*.

Este listado tiene mucha más información sobre la que volveremos más adelante, en la sección de IP.

Aunque normalmente la dirección MAC es fija en cada elemento de red, hay elementos de algunos fabricantes que permiten el cambio de esta dirección por otra cualquiera. Si tienes una tarjeta que permita el cambio de dirección MAC, desde GNU/Linux puedes utilizar el siguiente comando para cambiar tu dirección MAC:

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:08
# ifconfig eth0 up
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:01:02:03:04:08
          inet addr:192.168.10.13  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::213:8fff:fe7e:14fa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50146 errors:3 dropped:0 overruns:0 frame:3
          TX packets:57827 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14041182 (13.3 MiB)  TX bytes:8144060 (7.7 MiB)
          Interrupt:209 Base address:0x6000
```

DHCP

DHCP son las siglas en inglés de Protocolo de configuración dinámica de servidores (Dynamic Host Configuration Protocol). Es un protocolo de red en el que un servidor posee una lista de direcciones IP dinámicas y las va asignando a usuarios conforme estas van estando libres, sabiendo en todo momento quien ha estado en posesión de esa IP, cuanto tiempo la ha tenido y a quien se la ha asignado después. En los Hosting las IP's están compartidas. Provee los parámetros de configuración a las computadoras conectadas a la red informática que lo requieran (máscara, puerta de enlace y otros) y también incluyen mecanismo de asignación de direcciones de IP.

Este protocolo apareció como un protocolo estándar en octubre de 1993. En el RFC 2131 se puede encontrar la definición más actualizada. Los últimos esfuerzos describiendo *DHCPv6*, *DHCP en una red IPv6*, fue publicado como RFC 3315.

DHCP usa los mismos puertos asignados por el IANA (Autoridad de Números Asignados en Internet según siglas en inglés) en BOOTP: 67/udp para las computadoras servidor y 68/udp para las computadoras cliente. El protocolo funciona de la siguiente forma:

- **DHCP Discover**: La computadora cliente emite peticiones masivamente en la subred local para encontrar un servidor disponible, mediante un paquete de broadcast. El router puede ser configurado para redireccionar los paquetes DHCP a un servidor DHCP en una subred diferente. La implementación cliente crea un paquete UDP (Protocolo de Datagramas de Usuario según siglas en inglés) con destino 255.255.255.255 y requiere también su última dirección IP conocida, aunque esto no es necesario y puede llegar a ser ignorado por el servidor.
- **DHCP Offer** El servidor determina la configuración basándose en la dirección del soporte físico de la computadora cliente especificada en el registro CHADDRvbnv. El servidor especifica la dirección IP en el registro YIADDR. Como la cual se ha dado en los demás parámetros.
- **DHCP Request** El cliente selecciona la configuración de los paquetes recibidos de DHCP Offer. Lo anterior debido a que puede haber más de un servidor DHCP en el segmento y el cliente puede recibir más de un paquete DHCPOFFER. Una vez más, el cliente solicita una dirección IP, pero en esta ocasión utiliza la específica que le indicó el servidor del cual desea recibir la oferta.
- **DHCP Acknowledge** El servidor confirma el pedido y lo publica masivamente en la subred. Se espera que el cliente configure su interfaz de red con las opciones que se le han otorgado.
- **DHCP Inform** El cliente envía una petición al servidor de DHCP: para solicitar más información que la que el servidor ha enviado con el DHCPACK original; o para repetir los datos para un uso particular - por ejemplo, los navegadores usan DHCP Inform para obtener la configuración de los proxies a través de WPAD. Dichas peticiones no hacen que el servidor de DHCP refresque el tiempo de vencimiento de IP en su base de datos.
- **DHCP Release** El cliente envía una petición al servidor DHCP para liberar su dirección DHCP. Como los clientes generalmente no saben cuándo los usuarios pueden desconectarles de la red, el protocolo no define el envío del DHCP Release como obligatorio.

En GNU/Linux, como cliente tenemos `dhclient` (usado en la mayoría de distros), el cual realiza la petición a la red de una dirección IP de la siguiente forma:

```
# dhclient eth0
Internet Software Consortium DHCP Client 2.0pl5
Copyright 1995, 1996, 1997, 1998, 1999 The Internet Software Consortium.
All rights reserved.

Please contribute if you find this software useful.
For info, please visit http://www.isc.org/dhcp-contrib.html

sit0: unknown hardware address type 776
sit0: unknown hardware address type 776
Listening on LPF/eth0/00:13:8f:7e:14:fa
```

```
Sending on   LPF/eth0/00:13:8f:7e:14:fa
Sending on   Socket/fallback/fallback-net
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
receive_packet failed on eth0: Network is down
DHCPOFFER from 192.168.10.100
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.10.100
bound to 192.168.10.13 -- renewal in 10800 seconds.
```

El programa realiza la petición `DHCPDISCOVER` sobre la interfaz `eth0`, recibe un `DHCPOFFER` del servidor `192.168.10.100`, le solicita la dirección IP que le ofrece (no se ve en el listado cual es) a través de `DHCPREQUEST` y el servidor se la reserva con `DHCPACK`, la dirección 192.168.10.13, avisándole que tiene que renovar su petición en 10800 segundos. Si miramos los procesos veremos que `dhclient` queda en memoria para poder realizar esa renovación automáticamente, por lo que no tendremos que preocuparnos más.

Filtrado de paquetes ARP

Un cortafuegos (o firewall en inglés), es un elemento de hardware o software utilizado en una red de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas según las políticas de red que haya definido la organización responsable de la red. Su modo de funcionar es indicado por la recomendación RFC 2979, que define las características de comportamiento y requerimientos de interoperabilidad.

En este nivel existe un software en GNU/Linux denominado `arptables` que se encarga de realizar el filtrado de los paquetes ARP que pasan por las interfaces de red, además de permitir o denegar el acceso de ciertas direcciones MAC. El estudio de este tipo de herramientas es algo que requiere de mucho tiempo y práctica, por lo que pongo a continuación algunos ejemplos de lo que permite y dejo el resto a práctica personal de cada uno:

- Listar las reglas existentes:

```
# arptables -L
Chain INPUT (policy ACCEPT)

Chain OUTPUT (policy ACCEPT)

Chain FORWARD (policy ACCEPT)
```

- Denegar las peticiones ARP a la dirección MAC 00:14:2A:EF:4A:35 (xtreme2):

```
# arptables -A INPUT --source-mac 00:14:2A:EF:47:E1 -j DROP
# arptables -L
Chain INPUT (policy ACCEPT)
-j DROP --src-mac 00:14:2a:ef:47:e1

Chain OUTPUT (policy ACCEPT)

Chain FORWARD (policy ACCEPT)
```

- Limpiar las tablas de ARP y cambiar la política para denegarlo todo por defecto:

```
# arptables -F
# arptables -P INPUT DROP
# arptables -L
Chain INPUT (policy DROP)

Chain OUTPUT (policy ACCEPT)

Chain FORWARD (policy ACCEPT)
```

Capítulo 2. Nivel de red: IP

Una dirección IP es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red o nivel 3 del modelo de referencia OSI. Dicho número no se ha de confundir con la dirección MAC que es un número hexadecimal fijo que es asignado a la tarjeta o dispositivo de red por el fabricante, mientras que la dirección IP se puede cambiar, generalmente.

Los sitios de Internet que por su naturaleza necesitan estar permanentemente conectados, generalmente tienen una dirección IP fija (se aplica la misma reducción por IP fija o IP estática), es decir, no cambia con el tiempo. Los servidores de correo, DNS, FTP públicos, y servidores de páginas web necesariamente deben contar con una dirección IP fija o estática, ya que de esta forma se permite su localización en la red.

A través de Internet, los ordenadores se conectan entre sí mediante sus respectivas direcciones IP. Sin embargo, los seres humanos debemos utilizar otra notación más fácil de recordar y utilizar, como los nombres de dominio; la traducción entre unos y otros se resuelve mediante los servidores de nombres de dominio DNS, esto lo veremos más adelante.

Direcciones IP

En su versión 4, una dirección IP se representa mediante un número binario de 32 bits (IPv4). Las direcciones IP se pueden expresar como números de notación decimal: se dividen los 32 bits de la dirección en cuatro octetos. El valor decimal de cada octeto puede ser entre 0 y 255 (el número binario de 8 bits más alto es 11111111 y esos bits, de derecha a izquierda, tienen valores decimales de 1, 2, 4, 8, 16, 32, 64 y 128, lo que suma 255 en total).

En la expresión de direcciones IPv4 en decimal se separa cada octeto por un carácter ``.''. Cada uno de estos octetos puede estar comprendido entre 0 y 255, salvo algunas excepciones. Los ceros iniciales, si los hubiera, se pueden obviar. Ejemplo de representación de dirección IPv4: 164.12.123.65

Hay tres clases de direcciones IP que una organización puede recibir de parte de la *Internet Corporation for Assigned Names and Numbers* (ICANN): clase A, clase B y clase C. En la actualidad, ICANN reserva las direcciones de clase A para los gobiernos de todo el mundo (aunque en el pasado se le hayan otorgado a empresas de gran envergadura como, por ejemplo, Hewlett Packard) y las direcciones de clase B para las medianas empresas. Se otorgan direcciones de clase C para todos los demás solicitantes. Cada clase de red permite una cantidad fija de equipos (hosts).

- En una red de clase A, se asigna el primer octeto para identificar la red, reservando los tres últimos octetos (24 bits) para que sean asignados a los hosts, de modo que la cantidad máxima de hosts es 224 (menos dos: las direcciones reservadas de broadcast [tres últimos octetos a 255] y de red [tres últimos octetos a 0]), es decir, 16 777 214 hosts.

- En una red de clase B, se asignan los dos primeros octetos para identificar la red, reservando los dos octetos finales (16 bits) para que sean asignados a los hosts, de modo que la cantidad máxima de hosts es 216 (menos dos), o 65 534 hosts.

- En una red de clase C, se asignan los tres primeros octetos para identificar la red, reservando el octeto final (8 bits) para que sea asignado a los hosts, de modo que la cantidad máxima de hosts es 28 (menos dos), o 254 hosts.

- La dirección 0.0.0.0 es utilizada por las máquinas cuando están arrancando o no se les ha asignado dirección.

- La dirección que tiene su parte de host a cero sirve para definir la red en la que se ubica. Se denomia dirección de red.

- La dirección que tiene su parte de host a unos sirve para comunicar con todos los hosts de la red en la que se ubica. Se denomia dirección de broadcast.

- Las direcciones 127.x.x.x se reservan para pruebas de retroalimentación. Se denomina dirección de bucle local o loopback.

Hay ciertas direcciones en cada clase de dirección IP que no están asignadas y que se denominan direcciones privadas. Las direcciones privadas pueden ser utilizadas por los hosts que usan traducción de dirección de red (NAT) para conectarse a una red pública o por los hosts que no se conectan a Internet. En una misma red no puede existir dos direcciones iguales, pero sí se pueden repetir en dos redes privadas que no tengan conexión entre sí o que se sea a través de NAT. Las direcciones privadas son:

- Clase A: 10.0.0.0 a 10.255.255.255 (8 bits red, 24 bits hosts)
- Clase B: 172.16.0.0 a 172.31.255.255 (16 bits red, 16 bits hosts)
- Clase C: 192.168.0.0 a 192.168.255.255 (24 bits red, 8 bits hosts)

A partir de 1993, ante la previsible futura escasez de direcciones IPv4 debido al crecimiento exponencial de hosts en Internet, se empezó a introducir el sistema CIDR, que pretende en líneas generales establecer una distribución de direcciones más fina y granulada, calculando las direcciones necesarias y ``desperdiciando'' las mínimas posibles, para rodear el problema que las distribución por clases había estado gestando. Este sistema es, de hecho, el empleado actualmente para la delegación de direcciones.

Muchas aplicaciones requieren conectividad dentro de una sola red, y no necesitan conectividad externa. En las redes de gran tamaño a menudo se usa TCP/IP. Por ejemplo, los bancos pueden utilizar TCP/IP para conectar los cajeros automáticos que no se conectan a la red pública, de manera que las direcciones privadas son ideales para ellas. Las direcciones privadas también se pueden utilizar en una red en la que no hay suficientes direcciones públicas disponibles.

Las direcciones privadas se pueden utilizar junto con un servidor de traducción de direcciones de red (NAT) para suministrar conectividad a todos los hosts de una red que tiene relativamente pocas direcciones públicas disponibles. Según lo acordado, cualquier tráfico que posea una dirección destino dentro de uno de los intervalos de direcciones privadas no se enrutará a través de Internet.

La configuración de la red en GNU/Linux se lleva a cabo con varias herramientas, las herramienta más conocida es `ifconfig`. El cálculo de redes, aunque deberíamos de saber hacerlo perfectamente, podemos hacerlo o comprobarlo con la herramienta `ipcalc`.

En principio, tenemos la red 130.117.110.0/25, vamos a darle esta información a `ipcalc` para que nos proporcione toda la información referente al tipo de red, dirección de multidifusión, rangos de IP disponibles en la red, etc.:

```
# ipcalc 130.117.110.0/25
Address:   130.117.110.0        10000010.01110101.01101110.0 0000000
Netmask:   255.255.255.128 = 25 11111111.11111111.11111111.1 0000000
Wildcard:  0.0.0.127            00000000.00000000.00000000.0 1111111
=>
Network:   130.117.110.0/25     10000010.01110101.01101110.0 0000000
HostMin:   130.117.110.1        10000010.01110101.01101110.0 0000001
HostMax:   130.117.110.126      10000010.01110101.01101110.0 1111110
Broadcast: 130.117.110.127      10000010.01110101.01101110.0 1111111
Hosts/Net: 126                   Class B
```

Ahora, sabiendo que tenemos desde el 130.117.110.1 al 130.117.110.126, vamos a configurar un equipo para que esté dentro de esta red con el siguiente comando:

```
# ifconfig eth0:0 130.117.110.1 netmask 255.255.255.128 up
# ifconfig eth0:0
```

```
eth0:0    Link encap:Ethernet  HWaddr 00:13:8F:7E:14:FA
          inet addr:130.117.110.1  Bcast:130.117.110.127  Mask:255.255.255.128
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:209 Base address:0x6000
```

Como se puede apreciar, no ha hecho falta informarle al sistema acerca de la dirección de multidifusión, ella la calcula a través de la dirección de red junto con la máscara de red.

Desactivar la interfaz se haría con el comando `ifconfig eth0 down`.

Para comprobar si nuestro equipo tiene conexión con otras máquinas dentro de la misma red podemos utilizar el comando `ping`, el cual envía paquetes ICMP a otras máquinas, retornando un eco del mismo como comprobante (o ACK) de que llegó a su destino. Además, esta utilidad nos sirve para comprobar la latencia entre ambas máquinas en milisegundos. La utilización básica es:

```
# ping -c 4 130.117.110.2
PING 130.117.110.2 (130.117.110.2) 56(84) bytes of data.
64 bytes from 130.117.110.2 (130.117.110.2): icmp_seq=1 ttl=64 time=3.28 ms
64 bytes from 130.117.110.2 (130.117.110.2): icmp_seq=2 ttl=64 time=0.140 ms
64 bytes from 130.117.110.2 (130.117.110.2): icmp_seq=3 ttl=64 time=0.142 ms
64 bytes from 130.117.110.2 (130.117.110.2): icmp_seq=4 ttl=64 time=0.142 ms

--- 130.117.110.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.140/0.926/3.281/1.359 ms
```

No obstante, el control de latencia y saltos que realiza un paquete hasta llegar a su destino se controla mejor con `traceroute`, especificando un conjunto de tiempos de demora entre saltos, que son, por defecto, tres intentos de alcanzar el salto en que se encuentra. Un ejemplo de su uso (desde la oficina de Córdoba hacia el router del nodo de Madrid):

```
# traceroute -q 1 130.117.110.1
traceroute to 130.117.110.1 (130.117.110.1), 30 hops max, 40 byte packets
 1  xtremenetworks.dynalias.org (192.168.10.100)  0.280 ms
 2  192.168.153.1 (192.168.153.1)  55.754 ms
 3  18.Red-81-46-73.staticIP.rima-tde.net (81.46.73.18)  54.699 ms
 4  141.Red-80-58-81.staticIP.rima-tde.net (80.58.81.141)  51.720 ms
 5  So7-0-0-0-grtmadde2.red.telefonica.wholesale.net (84.16.8.113)  51.444 ms
 6  So1-2-0-0-grtparix3.red.telefonica-wholesale.net (213.140.43.186)  77.566 ms
 7  p10-0.core02.par02.atlas.cogentco.com (130.117.1.25)  71.664 ms
 8  p5-0.core01.bio01.atlas.cogentco.com (130.117.0.22)  93.350 ms
 9  p13-0.core01.mad05.atlas.cogentco.com (130.117.0.81)  336.396 ms
10  v101.mpd01.mad05.atlas.cogentco.com (130.117.1.42)  115.413 ms
11  130.117.110.1 (130.117.110.1)  113.053 ms
```

Enrutado

Una vez se tiene a una máquina con una dirección IP, normalmente una dirección IP privada, debemos de indicarle a la máquina por donde alcanzar a otras redes. Esto se hace a través de unas máquinas denominadas pasarelas o gateways (en inglés), aunque ya se está tomando el término de encaminador o router (en inglés) de forma más cotidiana.

La configuración de esta salida se realiza con el comando `route`, la cual también nos sirve para mostrar las rutas actuales. Si lo ejecutamos en este momento, sin haber especificado la ruta, veremos la ruta específica de la red, indicando la interfaz que utilizará para esa salida. Después creamos una salida para *default* y volvemos a mostrarlo:

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
130.117.110.0   0.0.0.0         255.255.255.128 U     0      0        0 eth0

# route add default gw 130.117.110.126
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
130.117.110.0   0.0.0.0         255.255.255.128 U     0      0        0 eth0
0.0.0.0         130.117.110.126 0.0.0.0         UG    0      0        0 eth0
```

Con esto, la máquina sabe que las peticiones que no vayan dirigidas a la red 130.117.110.0/25 debe de enviarlas a la máquina 130.117.110.126, a la cual sabe llegar por la ruta anterior. Los *flags* nos dicen que esta ruta es de *Gateway*.

En caso de tener una máquina servidora, en esta deberíamos de tener dos redes configuradas, una en cada interfaz de red, el modo de conectar estas dos interfaces para que las máquinas de una red puedan comunicarse con las máquinas de la otra red se hace de dos formas distintas, y con distintos resultados.

Una de ellas es a través del encaminado (routed), que consiste que una máquina puede acceder a otra máquina de otra red a través de un encaminador (router), aunque realmente, esto no es solo un encaminamiento, sino también un enmascarado, lo veremos más adelante.

La otra técnica es el puente (bridge), que consiste en que la máquina que hace de puente hace pasar los paquetes de una interfaz a otra estando todas las interfaces en el mismo direccionamiento de red. Lo veremos más adelante.

Direcciones IP múltiples

Normalmente, en GNU/Linux, las interfaces de red se listan desde 0, con las tres primeras letras de la palabra *ethernet*, para indicar de qué tipo son las conexiones de red a nivel físico. Por ello, si tenemos dos interfaces conectadas a nuestro sistema, una de ellas será `eth0` y otra será `eth1`.

Además, a cada dirección MAC le pueden corresponder varias direcciones IP, al igual que a una dirección IP le pueden corresponder varios nombres de dominio (DNS, lo cual veremos más adelante). Por ello, la interfaz tiene una dirección base, que será la que le hayamos asignado (130.117.110.1, en caso del ejemplo), y podemos asignarle otra dirección dentro de la misma red o de otra red distinta a la que pueda tener acceso mediante la misma conexión (y a la cual pertenezca).

La numeración sería, entonces, poniendo como ejemplo `eth0`, la dirección base sería la que está en `eth0`, y para las demás direcciones IP de esta interfaz se agregaría un sufijo numeral incremental: `eth0:0`, `eth0:1`, `eth0:2`, ... y así en adelante.

La configuración es exactamente igual, se usa `ifconfig` para realizarla y se especifica la interfaz a usar (junto con su sufijo en caso de configurar una de estas subinterfaces).

Enmascarado

Cuando una máquina se conecta con dos interfaces a dos redes distintas (A y B), las máquinas que pertenezcan a la red A y necesiten acceder a equipos de la red B, pueden configurar una ruta específica con la red B y que la pasarela sea la máquina que está entre las dos redes.

Hasta aquí todo parece lógico pero, ¿cómo sabe la máquina que tiene que pasar un paquete de una red a otra?, la máquina tiene que ser configurada para tomar los paquetes que provienen de la red A, con destino a la red B y realizar el paso. Esto se realiza, normalmente, modificando el contenido del fichero `/proc/sys/net/ipv4/ip_forward` para que contenga `1`, en lugar de `0`. Aunque se puede utilizar el comando `ipmasq`, que realiza los ajustes de forma automática.

Puentes

En las topologías de maya, especialmente, cuando hay, por ejemplo, cinco equipos y cada uno de ellos tienen instaladas cuatro tarjetas de red, o una tarjeta de red de cuatro interfaces, cada cable está directamente conectado de un equipo a otro, sin conmutadores de por medio. En este caso, el MTU es muy útil para el cálculo del camino más corto entre nodos, además de que todas las interfaces deberían de tener, no solo la misma red, sino también la misma dirección IP, así al mismo equipo, con la misma dirección IP pueden acceder cualquiera de los equipos que se encuentran en la red.

Por lo tanto, un bridge, tal como sonaría su traducción, es un puente que te sirve para unir dos o más interfaces de red (tarjetas de red - NICs) para que el tráfico fluya entre ellas de manera libre y en forma transparente. Es decir, una vez configurada una PC como bridge, la puedes conectar en tu LAN tal cual como si se tratara de un concentrador/conmutador. Las PCs que estén detrás del bridge no "verán" el bridge y su tráfico fluirá libremente y los paquetes pasarán de un lado al otro en forma transparente.

En GNU/Linux, el comando para hacer un puente es `brctl`, pero esto requiere de un proceso a seguir para que el puente salga bien. En principio, tenemos que crear la interfaz del puente, esta se creará como `br0`, al igual que `eth0`, esta interfaz puede tener sufijos para más direcciones IP y puedes crear más puentes donde agregues otras interfaces. Después se agregan las interfaces que vayan a conformar el puente, se les quita su dirección IP y se ponen en modo promiscuo (esto es para que lo escuche todo, no penséis cosas raras :-D) y por último se levanta la interfaz `br0`.

```
# brctl addbr br0
# brctl addbr br0 eth0
# brctl addbr br0 eth1
# ifconfig eth0 0.0.0.0 promisc
# ifconfig eth1 0.0.0.0 promisc
# ifconfig br0 up
```

IP Virtual

El proyecto de Linux Virtual Server proporciona una capa de transporte para el balanceo de carga dentro del núcleo de Linux, llamado así conmutador de capa-4. IPVS se ejecuta en una máquina actuando como un balanceador de carga al frente de un clúster de servidores reales, este puede direccionar solicitudes de servicios basados en TCP/UDP para los servidores reales y hacer que los servicios de los servidores reales aparezcan como un servicio virtual en una única dirección IP.

Para utilizar esta característica, normalmente necesitaremos recompilar el núcleo de Linux y activar el soporte de IPVS.

Algunos documentos que revisar sobre esta tecnología:

- **LVS-HOWTO** [http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/]
- **LVS Documentation** [http://www.linuxvirtualserver.org/Documents.html]
- **LVS Performance, Initial Tests with a single Realserver LVS** http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.htm [l]
- **LVS Cluster Configuration HOWTO** [http://www.redhat.com/support/resources/howto/piranha/]

Balanceo de Carga

Teniendo dos proveedores de Internet se puede hacer que una máquina que haga de router haga un balanceo entre los dos Internets para su salida con tan solo modificar la tabla de rutas, con la ayuda

de una herramienta del paquete `iproute`, conocida como `ip`. Esta herramienta proporciona muchas más características que solo el manejo de rutas, puesto que sustituye a las conocidas `ifconfig` y `route`, además de poder hacer más cosas aún.

Balancear la salida a Internet por dos proveedores es algo para lo que se emplea `ip`, en los siguientes enlaces se pueden ver algunas prácticas de ello:

- **Routing for multiple uplinks/providers** [http://lartc.org/howto/lartc.rpdb.multiple-links.html]
- **Enrutado en base a marcas de paquetes. Iproute + Iptables** [http://bulma.net/body.phtml?nIdNoticia=1615]

Capítulo 3. Nivel de transporte: TCP y UDP

La comunicación entre los equipos se llevará a través del establecimiento de una conexión, esta se realizará sobre el protocolo IP, como ya se conoce, pero el formato de transporte de estos paquetes a un nivel más bajo puede ser, principalmente, de dos tipos: TCP y UDP.

El Protocolo de Control de Transmisión (TCP en sus siglas en inglés, Transmission Control Protocol que fue creado entre los años 1973 - 1974 por Vint Cerf y Robert Kahn) es uno de los protocolos fundamentales en Internet. Muchos programas dentro de una red de datos compuesta por ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales enviarse datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

User Datagram Protocol (UDP) es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación, ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco sabemos si ha llegado correctamente, ya que no hay confirmación de entrega o de recepción. Su uso principal es para protocolos como DHCP, BOOTP, DNS y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos.

Puertos Software

Los puertos software a los que nos referimos en este apartado, son las conexiones lógicas que se realizan para comunicar a dos programas y permitir el intercambio de información entre los mismos. Este tipo de conexiones pueden llevarse a cabo dentro de la misma máquina o entre máquinas distintas en conexión a través de una red.

Los puertos software se diferencian por protocolo (TCP y UDP) y por estar en un rango de numeración comprendido entre 1 y 65535. Muchos de estos puertos están asignados, o preferentemente asignados, a ciertos protocolos. Esta asignación es una estandarización que lleva a cabo IANA. Por ejemplo, el puerto 21 es el usado por FTP (File Transfer Protocol), el 22 por SSH (Secure Shell), el 23 por Telnet, el 25 por SMTP (Simple Mail Transfer Protocol), el 80 por HTTP (HyperText Transfer Protocol), etc.

En GNU/Linux, se pueden ver las asignaciones a estos puertos en el fichero `/etc/services`, un extracto del fichero:

```
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
```

```
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
tacacs          49/tcp                          # Login Host Protocol (TACACS)
tacacs          49/udp
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
tacacs-ds       65/tcp                          # TACACS-Database Service
tacacs-ds       65/udp
bootps          67/tcp                          # BOOTP server
bootps          67/udp
bootpc          68/tcp                          # BOOTP client
```

Además, podemos establecer el estado de utilización de estos puertos de dos formas: pasiva o activa.

La utilización pasiva se refiere a que el programa, generalmente un servidor, ocupa el puerto a la espera de conexiones entrantes a las que atender. A esto se llama, comúnmente *escuchar en un puerto* (listen).

La utilización de un puerto de forma activa se refiere a que el programa, generalmente un cliente, utiliza un puerto de su sistema para acceder a otro local o remoto.

Un programa en GNU/Linux que nos permite ver el uso de los puertos es `netstat`. Este tiene muchos parámetros, los más conocidos son ``punta" (Programa, UDP, No resolver puerto ni IP inversamente en DNS, TCP y All), así obtenemos un listado como el que se muestra a continuación:

```
# netstat -punta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:33153           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 192.168.10.13:36134     207.46.106.45:1863      ESTABLISHED
tcp        0      0 192.168.10.13:51763     64.4.37.18:1863         ESTABLISHED
tcp        0      0 192.168.10.13:55598     216.155.193.141:5050    ESTABLISHED
tcp        0      0 192.168.10.13:57288     207.46.26.35:1863       ESTABLISHED
tcp        0      0 192.168.10.13:38740     66.230.200.228:80       ESTABLISHED
tcp        0      0 192.168.10.13:33724     130.117.110.4:143       ESTABLISHED
tcp        0      0 192.168.10.13:53335     65.54.228.44:1863       ESTABLISHED
tcp        0      0 192.168.10.13:55023     66.102.9.147:80         ESTABLISHED
tcp        0      0 192.168.10.13:34856     192.168.10.239:143      ESTABLISHED
tcp        0      0 192.168.10.13:56782     145.97.39.155:80        ESTABLISHED
tcp        0      0 192.168.10.13:56783     145.97.39.155:80        ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::5432                 :::*                    LISTEN
udp        0      0 127.0.0.1:32768         127.0.0.1:32768         ESTABLISHED
udp        0      0 0.0.0.0:32769           0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:1021            0.0.0.0:*
```

La primera columna se refiere al tipo de puerto que es, viendo que disponemos de `tcp`, `tcp6` (de

IP versión 6) y `udp`. Las columnas siguientes de `Recv-Q` y `Send-Q` normalmente estarán a cero, a menos que se tenga un tráfico desmesurado. Estas son las colas de paquetes a la espera de ser atendidos por el software.

Las dos siguientes columnas, `Local` y `Foreign Address`, son la conexión que se realiza desde el equipo hacia el exterior o, en caso de que sea `0.0.0.0:*`, que está escuchando para conexiones entrantes. La conexión de un puerto activo externo con uno pasivo local se mostrará en una segunda línea, puesto que cada puerto puede tener varias conexiones entrantes, no solo una.

El `State` muestra el estado de ese puerto específico, pueden ser: `LISTEN` (que está a la escucha en ese puerto. En la columna `Local Address` dará información de sobre la interfaz que está escuchando. En caso de ser 0.0.0.0 indicará que escucha de todas las interfaces conectadas en el sistema), `ESTABLISHED`, `SYN_SENT`, `SYN_RECV`, `FIN_WAIT1`, `FIN_WAIT2`, `TIME_WAIT`, `CLOSE`, `CLOSE_WAIT`, `LAST_ACK`, `CLOSING` y `UNKNOWN`.

Filtrado de paquetes

El filtrado de paquetes permite asegurar el ordenador de conexiones no deseadas a ciertos servicios que deseemos mantener abiertos para nuestros usos específicos, pero no para todos en general. El filtrado permite el acceso o denegación a un servicio específico (o puerto software).

Sobre cortafuegos está el libro *Firewalls Linux*, el cual trata todos los aspectos de filtrado en detalle. Aquí solo veremos detalles de introducción.

En principio, cabe destacar que el sistema de filtrado del núcleo Linux (*netfilter*), consta de tres reglas generales para el filtrado de paquetes. INPUT para los paquetes de entrada al sistema, OUTPUT para los paquetes que salen del sistema y FORWARD para los paquetes que pasan por el sistema (entre interfaces de red, a modo de encaminado).

El estado de estas reglas se puede observar con el comando `iptables` (en la versión 2.4 del núcleo Linux el programa se llamaba `ipchains`) de la siguiente forma:

```
# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Aquí se diferencias dos partes, una es la política, es decir, lo que se hace por defecto en cada regla, y otra son las reglas que se incluyen en cada una de las cadenas existentes. Por ejemplo, si queremos denegar el acceso a un equipo de la red, porque nos corta la música cuando accede por SSH:

```
# iptables -A INPUT -s xtreme4 -j REJECT
# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  192.168.10.13        0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Además, podemos cambiar la política, por ejemplo, por si queremos acceder a la pasarela, única-

mente, y que deniegue todo lo demás, para lo cual cambiaríamos las políticas y añadiríamos las reglas para la pasarela:

```
# iptables -P INPUT REJECT
# iptables -P OUTPUT REJECT
# iptables -P FORWARD REJECT
# iptables -A INPUT -s xtremecore -j ACCEPT
# iptables -A OUTPUT -s xtremecore -j ACCEPT
# iptables -L -n
Chain INPUT (policy REJECT)
target     prot opt source               destination
ACCEPT     all  --  192.168.10.100       0.0.0.0/0

Chain FORWARD (policy REJECT)
target     prot opt source               destination

Chain OUTPUT (policy REJECT)
target     prot opt source               destination
ACCEPT     all  --  192.168.10.100       0.0.0.0/0
```

Para limpiarlo todo y dejarlo como estaba tendremos que volver a poner las políticas de cada una de las reglas en ACCEPT y limpiar todas las reglas creadas. Esto se hace como sigue:

```
# iptables -P INPUT ACCEPT
# iptables -P OUTPUT ACCEPT
# iptables -P FORWARD ACCEPT
# iptables -F
# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Otra consideración a tener en cuenta es la diferencia entre REJECT y DROP. Cuando se quiere denegar el acceso a alguien podemos hacerlo con REJECT (rechazando el paquete) o con DROP (ignorándolo). Todos los expertos en sistemas aconsejan utilizar DROP porque da la incertidumbre al atacante de no saber si la dirección IP realmente existe o no, le demora mucho el realizar cualquier escaneo de puertos y no da información del sistema en absoluto.

Hasta aquí llega la introducción, la creación de nuevas reglas, utilización de NAT y DNAT la dejo para cada cual que quiera leerse el libro mencionado al principio de esta sección.

Capítulo 4. Nivel de Aplicación

En esta sección vamos a tratar el nivel de aplicación del modelo TCP/IP que se utiliza en Internet. Este consisten en los protocolos que todo el mundo conoce, tales son DNS, HTTP, SMTP, POP3, SIP... y un largo etcétera.

La mayoría de estos protocolos están recogidos en RFC (*Request For Comments*, son los documentos que genera la IETF (Internet Enginering Task Force) sobre protocolos para Internet). Su principal característica es que su emisión por Internet se realiza en texto plano, lo cual da la facilidad de poder comprobar que la comunicación es apropiada y adecuada y gracias a sus vertientes en mezcla con SSL o TLS, que encripta la transmisión, pueden ser aseguradas de forma relativamente fácil.

DNS

El Domain Name System (DNS) es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet. Aunque como base de datos el DNS es capaz de asociar distintos tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.

La asignación de nombres a direcciones IP es ciertamente la función más conocida de los protocolos DNS. Por ejemplo, si la dirección IP del sitio FTP de prox.ve es 200.64.128.4, la mayoría de la gente llega a este equipo especificando ftp.prox.ve y no la dirección IP. Además de ser más fácil de recordar, el nombre es más fiable. La dirección numérica podría cambiar por muchas razones, sin que tenga que cambiar el nombre.

Inicialmente, el DNS nació de la necesidad de recordar fácilmente los nombres de todos los servidores conectados a Internet. En un inicio, SRI (ahora SRI International) alojaba un archivo llamado HOSTS que contenía todos los nombres de dominio conocidos (técnicamente, este archivo aún existe - la mayoría de los sistemas operativos actuales todavía pueden ser configurados para revisar su archivo hosts).

El crecimiento explosivo de la red causó que el sistema de nombres centralizado en el archivo HOSTS no resultara práctico y en 1983, Paul Mockapetris publicó los RFCs 882 y 883 definiendo lo que hoy en día ha evolucionado al DNS moderno. (Estos RFCs han quedado obsoletos por la publicación en 1987 de los RFCs 1034 y 1035).

Para que el sistema GNU/Linux pueda resolver nombres a direcciones IP y viceversa necesita conectarse a un servidor DNS. La configuración del servidor DNS se hace a través del fichero `/etc/resolv.conf`, que tendrá este aspecto:

```
search xtremenetworks.dynalias.org
nameserver 192.168.10.100
nameserver 80.58.61.250
nameserver 80.58.61.254
```

Cuando se le dé al sistema un nombre que no corresponda a un dominio completo, es decir, un nombre de máquina como puede ser `xtreme1`, este supondrá que la máquina pertenece al dominio al que pertenece la máquina que hace la petición. El nombre de dominio de pertenencia se especifica en el archivo `resolv.conf` anteponiendo la palabra clave `search`.

Para los demás nombres de servidores de nombres se empleará la palabra clave `nameserver` y después la dirección IP para poder encontrarlo en Internet o en la red local.

Aunque los clientes como los navegadores, clientes de FTP y cualquier cliente de servicio de Internet realiza la conversión por DNS de un nombre dado a una dirección IP, nosotros también podemos emplear herramientas que nos permitan obtener información de registros DNS de dominios de Inter-

net. Estas herramientas son `nslookup` (ya obsoleta) y `dig`.

Aquí tienes un ejemplo de uso de `nslookup`:

```
# nslookup
> server xtremecore
Default server: xtremecore
Address: 192.168.10.100#53
> set type=mx
> telecomsolutions.es
Server:         xtremecore
Address:        192.168.10.100#53

Non-authoritative answer:
telecomsolutions.es     mail exchanger = 10 mail.telecomsolutions.es.
telecomsolutions.es     mail exchanger = 10 mail2.telecomsolutions.es.

Authoritative answers can be found from:
telecomsolutions.es     nameserver = ns2.interdomain.net.
telecomsolutions.es     nameserver = ns1.interdomain.net.
mail.telecomsolutions.es        internet address = 130.117.110.4
mail2.telecomsolutions.es       internet address = 130.117.110.4
ns1.interdomain.net     internet address = 212.170.240.180
ns2.interdomain.net     internet address = 212.170.240.181

> exit
```

A diferencia de `nslookup`, el comando `dig` es desde línea de comandos, no abre ninguna consola aparte para su uso, lo cual lo hace más útil para incluir en scripts del sistema o incluso para comprobaciones rápidas. Este se emplea de la siguiente forma:

```
# dig @xtremecore telecomsolutions.es mx

; < <>> DiG 9.3.2-P1 < <>> @xtremecore telecomsolutions.es mx
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER< <- opcode: QUERY, status: NOERROR, id: 65003
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;telecomsolutions.es.           IN      MX

;; ANSWER SECTION:
telecomsolutions.es.    44561   IN      MX      10 mail2.telecomsolutions.es.
telecomsolutions.es.    44561   IN      MX      10 mail.telecomsolutions.es.

;; AUTHORITY SECTION:
telecomsolutions.es.    348     IN      NS      ns2.interdomain.net.
telecomsolutions.es.    348     IN      NS      ns1.interdomain.net.

;; ADDITIONAL SECTION:
mail.telecomsolutions.es. 11553 IN      A       130.117.110.4
mail2.telecomsolutions.es. 44561 IN     A       130.117.110.4
ns1.interdomain.net.    172549  IN      A       212.170.240.180
ns2.interdomain.net.    172549  IN      A       212.170.240.181

;; Query time: 3 msec
;; SERVER: 192.168.10.100#53(192.168.10.100)
;; WHEN: Thu Sep 28 18:02:52 2006
;; MSG SIZE  rcvd: 195
```

Los registros dentro de un servidor DNS pueden ser de varios tipos:

- **A** = Address -- (Dirección) Este registro se usa para traducir nombres de hosts a direcciones IP.
- **CNAME** = Canonical Name -- (Nombre Canónico) Se usa para crear nombres de hosts adicionales, o alias, para los hosts de un dominio.
- **NS** = Name Server -- (Servidor de Nombres) Define la asociación que existe entre un nombre de dominio y los servidores de nombres que almacenan la información de dicho dominio. Cada dominio se puede asociar a una cantidad cualquiera de servidores de nombres.
- **MX** = Mail Exchange -- (Intercambiador de Correo) Define el lugar donde se aloja el correo que recibe el dominio.
- **PTR** = Pointer -- (Indicador) También conocido como 'registro inverso', funciona a la inversa del registro A, traduciendo IPs en nombres de dominio.