## Configuración básica

**Información una interfaz:**

# ethtool eth0

# ifconfig eth0

# ifconfig -a

# ip link show

# ip link show dev eth0

# ip -s link show dev eth0

**Desactivación de una interfaz de red:**

# ifconfig eth0 down

# ifdown eth0

# ip link set dev eth0 down

**Activación de una interfaz de red:**

# ifconfig eth0 up

# ifup eth0

# ip link set dev eth0 up

(!) La diferencia entre ifconfig e ifup/ifdown es que estas útlimas leen el archivo de configuración /etc/network/interfaces

**Asignación de direcciones IP:**

# ifconfig eth0 192.168.0.1/24 up

**Asignación de direcciones MAC:**

# ifconfig eth0 hw ether 00:01:02:03:04:04

(!) Aconsejable desactivar y activar la interfaz de red antes/después del cambio.

**Generar tráfico ICPM:**

# ping <*IP_destino*>

Generar tráfico ICMP elevado:

# ping -f -b <dir_bcast>

Activar aceptación de tráfico elevado:

# sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0

**Modificar la MTU de una red:**

# ifconfig eth0 mtu 1000

# ip link set dev eth0 mtu 1000

**Modificar la dirección MAC broadcast:**

# ip link set eth0 broadcast <*dirMAC*>

**Habilitar/deshabilitar el flag Multicast de la interfaz de red:**

# ifconfig eth0 multicast

# ifconfig eth0 -multicast

# ip link set dev eth0 multicast on

# ip link set dev eth0 multicast off

## Configuración de IP

**Configuración de parámetros IP:**

# ifconfig eth0 10.1.1.1

# ifconfig eth0 netmask 255.255.255.0

# ifconfig eth0 broadcast 10.1.1.255

# ifconfig eth0 10.1.1.1 netmask 255.255.0.0 broadcast 10.1.1.255

# ifconfig eth0 10.1.1.1/24

**Borrar la configuración de una interfaz:**

# ifconfig eth0 **0.0.0.0**

**Configuración de parámetros IP (ip adress)**

Para modificar la configuración IP de la interfaz de red mediante la orden ip, previamente es necesario borrar (delete) la configuración anterior y añadir (add) la nueva configuración. En caso de no borrar la anterior, la interfaz mantiene ambas direcciones simultáneamente.

# ip address show dev eth0

# ip address delete 192.168.1.1/24 dev eth0

# ip address add 10.1.1.1 dev eth0

# ip address add 10.1.1.1/24 broadcast 10.1.1.255 dev eth0

Al contrario que la orden ifconfig, ip con comprueba la clase de la dirección para asignar una netmask y dirección broadcast. Por ello es necesario especificar la máscara en formato CDIR. Podemos forzar a ip a calcular la dirección broadcast con **'+'**:

# ip address add 10.1.1.1/24 broadcast + dev eth0

A diferencia de ifconfig, ip no activa de manera automática la interfaz, por lo que hay que activarla explícitamente.

**Asignación de varias IPs a la misma interfaz**

Es posible asignar varias direcciones IP a una misma interfaz de red. La interfaz será capaz de comunicarse a través de todas las IP's que tenga asociadas. Esto puede realizarse con ifconfig (alias obligatorio) o mediante ip address (alias opcional).

# ifconfig eth0:1 192.168.1.1/24

# ifconfig eth0:1

Cuando se desactiva un alias (down), éste desaparece y no puede activarse simplemente con 'up'. Hay que volver a crearlo.

# ifconfig eth0:1 down

Con ip address:

# ip address add 192.168.1.1/24 broadcast 192.168.1.255 dev eth0

# ip address delete 192.168.1.1/24 dev eth0

# ip address add 192.168.1.1/24 broadcast 192.168.1.255 label eth0:1 dev eth0

**Archivo de configuración de parámetros IP**

En las distribuciones basadas en Debian, las configuraciones hechas mediante ifconfig o ip address se pierden al reiniciar. Las configuraciones permanentes deben reflejarse en el archivo **/etc/network/interfaces**

## Protocolo ARP

Las tablas ARP almacenan de forma temporal las asociaciones IP-MAC de los dispositivos con los que nos hemos comunicado recientemente.

**Visualización de tablas ARP:**

# arp -an

# ip neighbour show

Para una interfaz concreta:

# arp -an -i eth0

# ip neighbour show dev eth0

**Borrar una entrada de la tabla:**

# arp -d 192.168.1.2 -i eth0

# ip neighbour delete *<dirIP>* dev eth0

**Añadir una entrada permanente:**

# arp -s *<dirIP>* *<dirMAC>* -i eth0

# ip neighbour add *<dirIP>* lladdr *<dirMAC>* dev eth0

**Añadir una entrada temporal:**

# arp -s *<dirIP>* *<dirMAC>* -i eth0 temp

# ip neighbour add *<dirIP>* lladdr *<dirMAC>* dev eth0 nud reachable

**Deshabilitar protocolo ARP en una interfaz:**

Si se hace, es necesario introducir manualmente las direcciones IP y MAC en la tabla ARP de las máquinas con las que vamos a comunicarnos.

# ifconfig eth0 arp  // Habilitar

# ip link set dev eth0 arp on

# ifconfig eth0 -arp  // Deshabilitar

# ip link set dev eth0 arp off

## Encaminamiento IP

**Tabla cache de rutas**

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 encuentra aquí la dirección, no busca más.

Para visualizar su contenido:

# route -C -n

# ip route show cache

# netstat -rnC

**Tabla de rutas**

Visualización de la tabla de rutas:

# route -n

# netstat -rn / # netstat -nr

# ip route show

- Gateway = 0.0.0.0 → Misma red
- Destination = 0.0.0.0 → Default router
- Genmask = 255.255.255.255 → El destino es una máquina específica, no una red.
- Genmask = 0.0.0.0 → Default router

**Añadir/eliminar default router a la tabla:**

# route add default gw *<dirIPRouter>*

# route delete default gw *<dirIPRouter>*

# ip route add default via *<dirIPRouter>*

# ip route delete default

**Activar reenvío de datagramas en una máquina (router?)**

```
# sysctl -w net.ipv4.conf.all.forwarding=1
```

## Protocolo ICMP

**Comprobación de equipo alcanzable (ping)**

```
# ping -c 3 -s 30 192.168.0.21
```

El parámetro **-c** indica el número de paquetes a enviar y **-s** indica el tamaño de su campo de datos.

Se puede realizar una solicitud de Echo a toda una subred con el parámetro **-b**. Para activar la respuesta de una máquina a estos mensajes:

```
# net.ipv4.icmp_echo_ignore_all=0
```

```
# net.ipv4.icmp_echo_ignore_broadcasts=1
```

La primera indica si deben ignorarse o no todas las peticiones de echo. La segunda determina si si deben ignorarse sólo las solicitudes de echo dirigidas a direcciones de broadcast.

## Fragmentación y reensamblado

Cuando la longitud de un datagrama excede la MTU de la red que quiere atravesar, debe fragmentarse.

La fragmentación puede realizarse en el origen o en alguno de los encaminadores intermedios. Cuando se produce en el origen, la máquina debe conocer la mínima MTU hasta el destino, lo que se conoce como *descubrimiento de MTU*.

Descubrimiento de MTU

1. La máquina origen envía un datagrama ajustándose a la MTU de su red local, con el bit DF = 1.
2. Si este datagrama encontrase una red con MTU menor, el origen recibiría un mensaje ICMP de host unreachable, subtipo **fragmentación necesaria.** Este mensaje contiene la MTU de la red conflictiva.
3. El proceso se repite hasta que el datagrama llegue al destino.

Se puede controlar el uso de este procedimiento mediante el parámetro:

```
# net.ipv4.ip_no_pmtu_disc=0
```

Una vez que se ha descubierto la MTU de una red, se añade este dato a la tabla de encaminamiento *cache*.

Si queremos repetir el proceso, previamente hay que vaciar la tabla con la orden:

```
# ip route flush table cache
```

## Simulación de un cortafuegos

Puede incluirse una regla de cortafuegos en un encaminador para que solo deje pasar el primero de los fragmentos de un datagrama:

```
# iptables -I FORWARD -f -j DROP
```

Puede utilizarse para simular la pérdida de un fragmento. Si se hace se recibirá un mensaje ICMP *time-to-live exceeded* cuando expire el temporizador de reensamblado.

Para modificar este temporizador:

```
# net.ipv4.ipfrag_time=30 (por defecto)
```

## Netcat: aplicaciones TCP

**Crear una aplicación TCP servidor:**

```
# nc -l -p <server_port>
```

```
# nc6 -l -p <server_port>
```

-l: '*listen*' y -p: *port*

**Crear una aplicación TCP cliente:**

```
# nc <server_IP_address> <server_port>
```

*Nota:* los servidores abiertos con netcat sólo aceptan un cliente. Además, salgamos primero del servidor o del cliente, el programa siempre finaliza.

**Trucos para ejecuciones en terminal:**

- Ejecutar en background → &
- Salir de background → Ctrl+Z
- Detener? → Ctrl+C
- Traer el proceso en bg a fg → fg
- Pasar un proceso a bg → bg

## Netcat: aplicaciones UDP

**Crear una aplicación UDP servidor:**

```
# nc -l -u -p <server_port>
```

**Crear una aplicación UDP cliente:**

```
# nc -u <server_IP_address> <server_port>
```

## Manejo de puertos

*Nota 2:* Listado de puertos en /etc/services

**Visualizar puertos UDP abiertos:**

# netstat -ua

# netstat -uan

**Visualizar los puertos TCP abiertos:**

$ netstat -ta

$ netstat -tan

## La aplicación *'Echo'*

La aplicación **echo** es un sencillo programa servidor que devuelve cualquier cadena de texto que se le envía. Tiene asociado el puerto bien conocido n°**7**.

El servidor echo se arranca a través del super-demonio de red **inetd**. En caso de que esta aplicación servidora no esté arrancada, es necesario incluir o descomentar la siguiente línea del archivo **/etc/inetd.conf**:

- echo dgram udp wait root internal

A continuación sería necesario reiniciar el demonio inetd mediante:

# /etc/init.d/openbsd-inetd restart

Una aplicación cliente que se conecte al servidor echo:

# nc -u *<IP_servidor>* 7

## La aplicación *'Telnet'*

La aplicación **telnet** es una aplicación cliente-servidor basada en **TCP** que permite abrir desde la máquina cliente una sesión o shell remota de la máquina servidora. Tiene asociado el puerto **23**.

Las aplicaciones cliente y servidor Telnet son las siguientes:

- Servidor: **in.telnetd**
- Cliente: **telnet**

El servidor in.telnetd se arranca a través del súper-demonio de red inetd. En caso de que esta aplicación servidora no esté arrancada, es necesario incluir o descomentar de **/etc/inetd.conf**:

- telnet stream tcp nowait telnetd /usr/sbin/tcpd /usr/sbin/in.telnet

A continuación sería necesario reiniciar el demonio inetd mediante la siguiente orden:

# /etc/init.d/openbsd-inetd restart

Ejecutar la aplicación telnet cliente:

$ telnet *<IP_servidor>*

*Nota:* telnet pide un usuario y contraseña. Como la información no va cifrada, no se permite la conexión con el usuario *root*.

Para activar las conexiones telnet en una determinada máquina, se debe añadir al fichero **/etc/inetd.conf**:

- telnet stream tcp6 nowait root /usr/sbin/tcpd /usr/sbin/in.teletd

## La aplicación *'Secure Shell'*

**Ssh** es una aplicación cliente-servidor basada en TCP, que permite abrir una sesión shell en la máquina remota de forma similar a telnet, pero cifrando la información. Tiene asociado el puerto **22**.

Las aplicaciones cliente y servidor ssh son:

- Servidor: **sshd**
- Clientes: **ssh**, **scp** y **sftp**

El servidor de Secure Shell tiene asociado su propio demonio (**sshd**) y su propio *script* de arranque y parada del servicio.

- Para arrancar el servidor sshd:

  # /etc/init.d/ssh start

- Para parar el servidor sshd:

  # /etc/init.d/ssh stop

- Para reiniciarlo:

  # /etc/init.d/ssh restart

Con *'netstat -tan'* podemos comprobar que el servidor sshd (puerto TCP 22) está abierto (LISTEN).

**Ejecutar el cliente ssh:**

$ ssh *<usuario>*@*<IP_servidor>*

## Modificar números de puerto

Es posible modificar el número de puerto asociado a cualquier aplicación servidora (/etc/services). No obstante, habría que notificar a los potenciales clientes del cambio, para que puedan conectarse.

**Para aplicaciones gestionadas por inetd:**

1. Editar el archivo /etc/services
2. Modificar el número de puerto del servicio concreto (ej: telnet).
3. Reiniciar el proceso **inetd** mediante:

   ```
   # /etc/init.d/openbsd-inetd restart
   ```

4. Para que el cliente pueda conectarse a telnet (por ejemplo), si tiene el puerto cambiado:

   ```
   $ telnet <IP_servidor> 10000
   ```

**Aplicaciones gestionadas por su propio proceso servidor:**

Cada una de ellas suele tener su propio archivo de configuración, donde normalmente se puede modificar el puerto de servicio.

Por ejemplo, en el caso de Secure Shell, el archivo de configuración es **/etc/ssh/sshd_config** y el número de puerto figura en la línea:

```
#What ports, IPs and protocols we listen for

port 22 #Esto se cambia
```

Después de cambiar el número de puerto, habrá que reiniciar el proceso sshd:

```
# /etc/init.d/ssh restart
```

**Para conectarse, el cliente deberá hacer:**

```
$ ssh <usuario>@<IP_servidor> -p <puerto>
```