Práctica 2. TCP/IP

1 Introducción

La práctica pretende la familiarización con las capas 2, 3 y 4 de la arquitectura TCP/IP. Para ello se utilizará el entorno del laboratorio mostrado en la primera práctica.

1.1 Sinopsis teórica

- Las capas 2, 3 y 4 del modelo TCP/IP tienen sus propios sistemas de direccionamiento. En redes Ethernet [1], las direcciones de nivel 2 enlace, se llaman direcciones MAC; en la capa 3 IP, las direcciones son direcciones IP; en la capa 4 de transmporte las direcciones se llaman puertos y determinan las diferentes aplicaciones que los escuchan o los envían.

- El sistema de direccionamiento IP es un sistema universal y centralizado cuyas direcciones constan de 4 octetos y se representan en formato decimal separados por puntos. Las direcciones IP se agrupan en redes, cuyos miembros tienen en común un número contínuo de bits de sus direcciones IP. El número contínuo de bits del mismo valor en una red se denomina máscara de red y se representa por su valor decimal separado por octetos, o por el número decimal de unos. El uso de una máscara de red determina el número de equipos que componen esa red. Por defecto, si no se especifica máscara de red (ClassFull), existen cinco clases de direcciones, de las cuales las tres primeras (A, B y C) se utilizan para direccionamiento unicast y broadcast, una (D) para direccionamiento multicast y la última (E) están reservadas (aunque recientemente se está haciendo uso de ellas como direcciones globales):

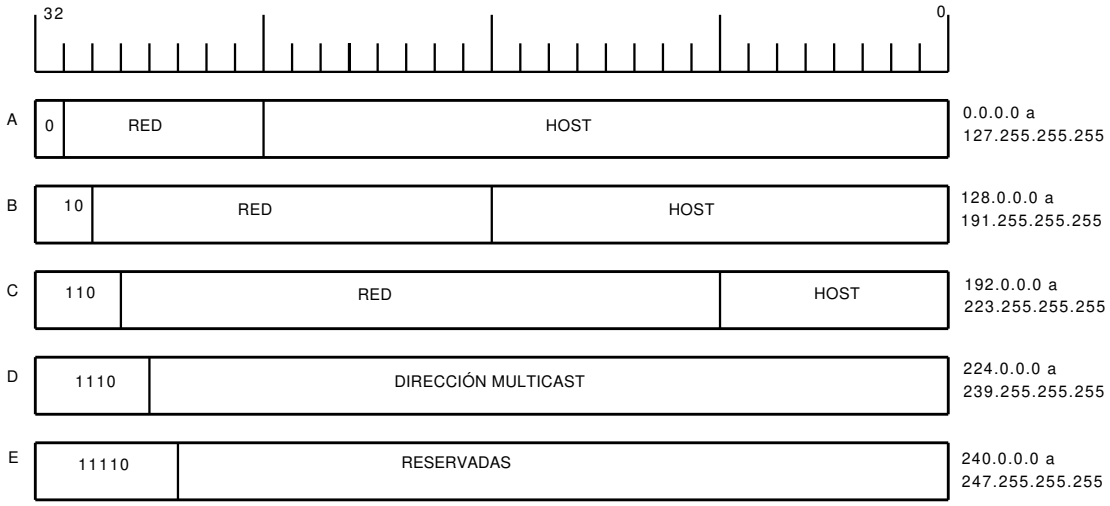

Figure 1: Clases de direcciones IP

- El sistema central de asignación de direcciones IANA contempló la existencia de rangos de direcciones de ámbito privado, que cualquiera puede utilizar a discreción, pero que no son accesibles desde las redes públicas como Internet. Los rangos privados son:

  - Clase A: 10.0.0.0 a 10.255.255.255 (8 bits red, 24 bits hosts). Uso en grandes entornos.

[1] Se utiliza Ethernet de forma genérica para definir las tecnologías 802.3 que incluyen Ehternet (10Mbps), FastEthernet (100Mbps) y GigabitEthernet (1Gbps).

- Clase B: 172.16.0.0 a 172.31.255.255 (16 bits red, 16 bits hosts). 16 redes clase B contiguas, uso en entornos medio/grande.
- Clase C: 192.168.0.0 a 192.168.255.255 (24 bits red, 8 bits hosts). 256 redes clase C contiguas; entornos de uso pequeño/medio.

Asímismo, existen unos rangos de uso específico:

- Clase A: *127.0.0.0* (8 bits red, 24 bits hosts). Es una red utilizada para direccionar dispositivos o servicios software en el propio equipo. Destaca la dirección *127.0.0.1* asociada a la interfaz "lo" o loopback. Se suele utilizar para comprobar el funcionamiento de la pila de protocolos TCP/IP sin tener que estar conectado físicamente a ninguna red.
- Clase B: *169.254.0.0* (16 bits red, 16 bits hosts). Definidas en el RFC 3927. Es una red con direcciones para autoconfiguración. Se suelen llamar direcciones autoconfiguradas, link-local addresses o APIPA. Se suelen autoconfigurar si no hay IP estática y si la consulta a un servidor DHCP ha sido fallida. Los routers no la reenvían, al igual que los rangos privados (RFC 1918).
- Dirección *0.0.0.0*. Es una dirección IP que denota una dirección por defecto y puede tener diferentes interpretaciones según el contexto. Habitualmente se utiliza como "red por defecto" en tablas de rutas para definir "Internet". También sirve para definir la IP por defecto de una interfaz en tablas de rutas.

- La cabecera IP dispone de los siguientes campos:

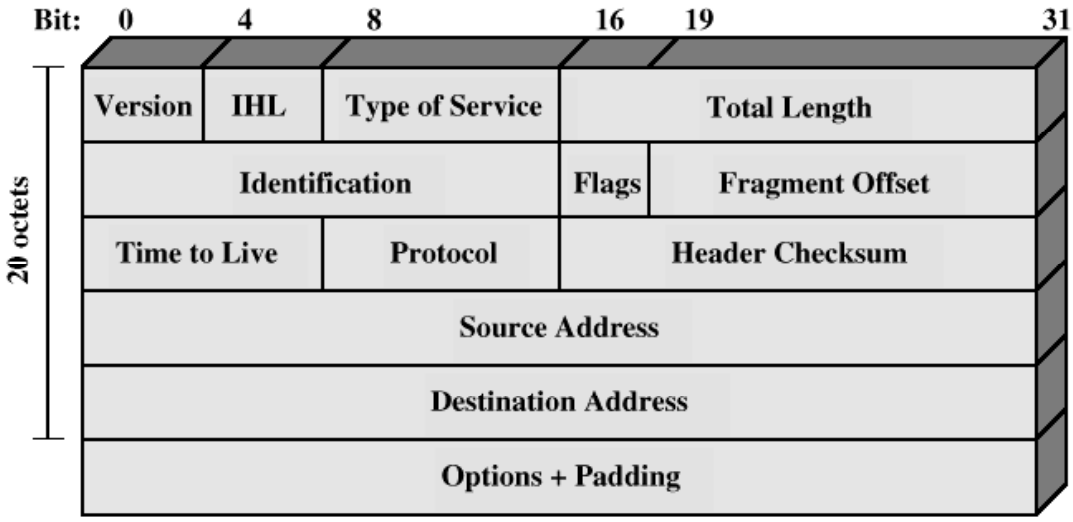

Figure 2: Cabecera IP

- Versión (4 bits): Nº de versión del protocolo IP. Actualmente en la 4.
- IHL, Longitud de cabecera de Internet (4 bits) expresada en bloques de 32 bits.
- Tipo de servicio (8 bits): parámetros de seguridad, prioridad, retardo y rendimiento.
- Longitud total (16 bits): longitud total del datagrama en octetos.
- Identificador (16 bits): nº de secuencia que identifica un datagrama (una "conexión") junto con los campos dirección origen, destino y protocolo.
- Etiquetas (3 bits): 3 bits, fragmentación (1 no fragmentar), segmentación (bit Mas). El tercero no se usa.
- Desplazamiento del fragmento (13 bits): posición de los datos dentro del datagrama completo antes de ser fragmentado. Se mide en bloques de 64 bits (8 octetos).
- TTL, tiempo de vida (8 bits): nº de saltos que puede permanecer un paquete en una red. Cada router hace disminuir el contador en 1.
- Protocolo: protocolo de la capa superior a IP (ICMP, TCP, UDP)

  - Suma de comprobación de la cabecera (16 bits): control de error mediante análisis de la cabecera. (complemento a 1 de las palabras de 16 bits de la cabecera).
  - Dirección origen/destino (32 bits): código de direccionamiento de 32 bits, dando un número de host (interfaz lógica IP) y/o un número de red.
  - Opciones (variable): contiene opciones del usuario a la red. Normalmente no se usa, salvo prioridades y QoS.
  - Relleno: obliga a la cabecera del datagrama IP a tener un tamaño múltiplo de 32 bits (4 octetos).
  - Datos: lleva información del protocolo y/o los datos del datagrama de la capa superior.

- El direccionamiento a nivel de aplicación se realiza mediante puertos TCP o UDP. Cuando una aplicación de un host se comunica con otra aplicación de otro host, la comunicación entre hosts se realiza por direcciones IP y entre aplicaciones mediante puertos. En TCP/UDP se permiten 2^{16} puertos. Los primeros 1024 son asignados por el IANA para aplicaciones conocidas. Algún ejemplo: puerto TCP 22 ssh, TCP 25 smtp, TCP 80 http, UDP 161 snmp, UDP 53 dns.

- Existen utilidades que no usan puertos para comunicarse entre sí. Un ejemplo muy conocido es el protocolo ICMP, cuya principal aplicación es el comando "ping". En esos casos se utiliza el campo "protocolo" de la cabecera IP, para distinguir unas utilidades de otras.

1.2 Objetivos

La práctica propuesta tiene el objetivo de conocer las herramientas básicas de configuración y visualización de los parámetros TCP/IP y de enlace de un host. Por extensión servirá para repasar el sistema de direccionamiento IP, tablas de enrutamiento básicas y parámetros de enlace.

Los objetivos concretos serían:

- Conocer y manejar herramientas básicas de gestión de interfaces Ethernet.
- Ser capaz de arrancar un equipo remotamente mediante WoL.
- Manejar el comando "ip addr".
- Manejar el comando "ip link"
- Visualizar el volumen de tráfico por las interfaces.
- Manejar el programa "nmap"
- Manejar el programa "tcpdump". Comprobación de las interfaces y el direccionamiento de cualquier dispositivo, en particular hosts linux.
- Conocer y manejar el sistema "netcat" de establecimiento de conexiones.
- Conocer algunos trucos útiles para activación remota de servicios: PortKnocking.
- Conocimiento básico del DNS.

2 Entorno

Cada grupo de prácticas se reagrupará por parejas, preferentemente con hosts enfrentados, es decir, PC10-PC11, PC12-PC13 etc. En los ejemplos desarrollados a continuación se usará PC10 y PC11 como equipos conectados entre sí. Alguno de los comandos requerirán una ejecución con el usuario root.

Los hosts del laboratorio arrancan por defecto en la red *192.168.29.0/24*, y se les pone como pasarela (gateway) por defecto la dirección *192.168.29.1*, para la salida a Internet. En adelante se representará a las pasarela como GW. Al final del texto se presenta un esquema de la red del laboratorio, donde el GW está representado como F0. Es posible que algunos de los equipos presentados no estén operativos.

Cuando alguno de los equipos no dispone de conectividad a Internet se realizará el siguiente flujo de tareas:

1. Comprobar que la dirección IP del equipo existe y es correcta.

2. Conocer la dirección IP de la pasarela de salida (GW). ¿La dirección IP de GW está en la misma subred que la del equipo?. Si la respuesta es SI, pasar al siguiente paso, si es NO, encontrar una pasarela en la red del equipo, o cambiar la dirección IP del equipo.

3. Comprobar conectividad con GW. Se puede hacer con el comando "ping". Si hay conectividad pasar al paso siguiente, si no hay conectividad el problema está en GW, en la red local o en la tarjeta de red del equipo.

4. Comprobar la existencia de la ruta por defecto. La ruta por defecto (default route) debe tener como siguiente salto a GW. Si ya hay ruta por defecto correcta pasar al siguiente paso, si no hay ruta por defecto, hay que crearla.

5. Conectividad IP a Internet. Se puede comprobar con el comando "ping" hacia una dirección habilitada de Internet (p.ej. *8.8.8.8*). Si hay conectividad se pasa al siguiente paso. Si no hay conectividad, puede haber problema en GW o en la red externa, en cuyo caso habría que actuar sobre ella.

6. Navegación web. Si no se puede navegar por Internet, habiendo pasado los pasos anteriores, el problema debe de ser del DNS o del navegador.

7. Una vez que existe conexión a Internet, es posible averiguar la dirección IP pública con la que se conecta el equipo mediante el uso de algunos servicios en Internet como *http://www.whatsmyip.org/* o *http://ifconfig.me/*. Si se quiere hacer desde línea de comandos se puede hacer de varias formas, siempre conectando a uno de los servicios online disponibles:

```
PCX$ curl curlmyip.com
PCX$ curl -s icanhazip.com
PCX$ curl -s http://ifconfig.me
```

Se recomienda la visualización contínua del tráfico y estado de las interfaces. Esta información se encuentra en "/proc/net/dev" [2]. La visualización de forma contínua se puede hacer mediante la creación en línea de comandos de un script como el siguiente:

```
PCX# while true; do cat /proc/net/dev; sleep 1; clear; done
```

o si se dispone del programa *watch*:

```
PCX$ watch -n 1 "cat /proc/net/dev "
```

3 Ethernet 802.3

3.1 ARP

El protocolo de resolución de direcciones (ARP en sus siglas en inglés) relaciona las capas de enlace y de red, particularmente la tecnología Ethernet 802.3 y la tecnología IP a nivel de mapeo de direcciones de ambas capas.

Básicamente ARP busca la dirección Ethernet asociada a una dirección IP, para poder enviar el paquete en formato UNICAST al destino marcado por dicha IP. El dispositivo origen envía un ARP REQUEST como broadcast a todo el segmento Ethernet, y todos los dispositivos del segmento reciben ese paquete y lo suben a su capa de red IP donde consultan la dirección IP preguntada. El dispositivo destino que tenga la dirección IP solicitada responde con un ARP RESPONSE, de forma que desde ese momento todos los paquetes dirigidos a ese dispositivo desde ese origen tendrán como dirección MAC destino la del destino final del paquete.

ARP puede ser útil para detectar los diferentes dispositivos que hay en un segmento de red, así como para detectar duplicidad de direcciones IP.

[2] En este directorio se encuentra prácticamente toda la información en tiempo real de la red

3.1.1 Detección de los dispositivos de un segmento de red

A diferencia del protocolo ICMP muy útil para la detección de un dispositivo a nivel IP, el protocolo ARP no suele ser objetivo de los firewalls o filtros, pues su filtrado podría provocar un malfuncionamiento de toda la red. Por ello, dentro de un segmento de red, el método mas eficaz para conocer la disponibilidad de un equipo es comprobando si responde a una solicitud ARP REQUEST. Para ello la mejor forma es provocar una solicitud masiva de conexión (por ejemplo con el comando *fping* - sección 5.3) y despues visualizar la caché ARP (por ejemplo con el comando *ip neigh* - sección 8.2).

Para mostrar que la visualización de la tabla ARP es mas fiable que la respuesta al comando "ping" para comprobar la disponibilidad de un equipo en la red, se puede deshabilitar la respuesta ICMP (ICMP 0) en un equipo ejecutando (en PC11 por ejemplo):

```
PC11$ echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

Así, aunque PC11 no responderá a un "ping", su entrada ARP aparecerá activa.
Por ejemplo, en la red del laboratorio se ejecutaría el comando "fping" de la forma:

```
PC10$ fping -q -a -c 1 -g 192.168.29.0/24
```

donde PC11 aparecerá como no disponible. En cambio, visualizando la tabla ARP con el comando "ip neigh" aparecerá su dirección IP y su MAC en estado REACHABLE o STALE.

```
PCX#:/etc# ip neigh show
192.168.29.1 dev eth0 lladdr dc:0e:a1:16:6b:06 DELAY
192.168.29.111 dev eth0 lladdr d8:50:e6:3b:a1:3e STALE
192.168.29.123 dev eth0 lladdr 88:30:8a:43:c9:4e REACHABLE
192.168.29.119 dev eth0 lladdr b8:70:f4:a1:02:07 STALE
192.168.29.124 dev eth0  FAILED
192.168.29.71 dev eth0 lladdr 00:55:66:77:aa:bb PERMANENT
.......
```

Las salidas de *ip neigh show* que no sean FAILED serán dispositivos detectados.

3.1.2 Detección de direcciones IP duplicadas

El comando *arping* genera solicitudes ARP REQUEST utilizando como argumento la dirección IP de la que se solicita su dirección MAC. Cuando se genera un ARP REQUEST broadcast hacia una dirección IP y responde mas de un equipo, es que existen direcciones IP duplicadas.

```
PCX:~# arping -I eth0 -c 2 192.168.29.123
ARPING 192.168.29.123
60 bytes from 00:03:0d:73:2a:98 (192.168.29.123): index=0 time=269.989 usec
60 bytes from f0:bf:97:de:54:86 (192.168.29.123): index=1 time=1.229 msec
60 bytes from 00:03:0d:73:2a:98 (192.168.29.123): index=2 time=497.980 usec
60 bytes from f0:bf:97:de:54:86 (192.168.29.123): index=3 time=1.339 msec
.......
```

Puede verse como responden dispositivos con dos direcciones MAC diferentes, por lo que existe una duplicidad en las direcciones IP. Incluso la diferencia de retardos entre ambos equipos puede dar una pista de donde están ubicados.

3.2 Wake on Lan (WoL)

Wake on Lan es una tecnología que permite arrancar equipos de forma remota. Para ello se envía un determinado paquete llamado MagicPacket desde el equipo de control (PC_B) hacia el equipo a arrancar (PC_A) . Este paquete contiene la dirección MAC del equipo a arrancar (PC_A). El MagicPacket se envía a nivel de capa 2 hacia todas las interfaces mediante la dirección broadcast. La dirección IP no es utilizada por PC_B.

En la topología del laboratorio supondremos que se arrancará PC22 desde PC24. Para ello deberá conocerse la dirección MAC del equipo PC22 (MAC_PC22). Una buena forma de conocer las direcciones MAC de equipos en una red local, es generar un broadcast IP sobre esa red y consultar posteriormente la tabla ARP (Ver 5.2 y 8.2). A veces el broadcast no está habilitado en los equipos, en cuyo caso podría ser útil generar un ICMP8 (ping) hacia la red IP local mediante *nmap* o similar (ver 10.

El MagicPacket es una trama Ethernet, con cabecera IP y cabecera UDP, y que contiene en su campo de datos un total de 102 octetos, de los cuales:

- los 6 primeros octetos están puestos a FF.
- 16 conjuntos de 6 octetos conteniendo cada uno de ellos la dirección MAC de PC_B.

Una captura de un MagicPacket se muestra en la figura 3

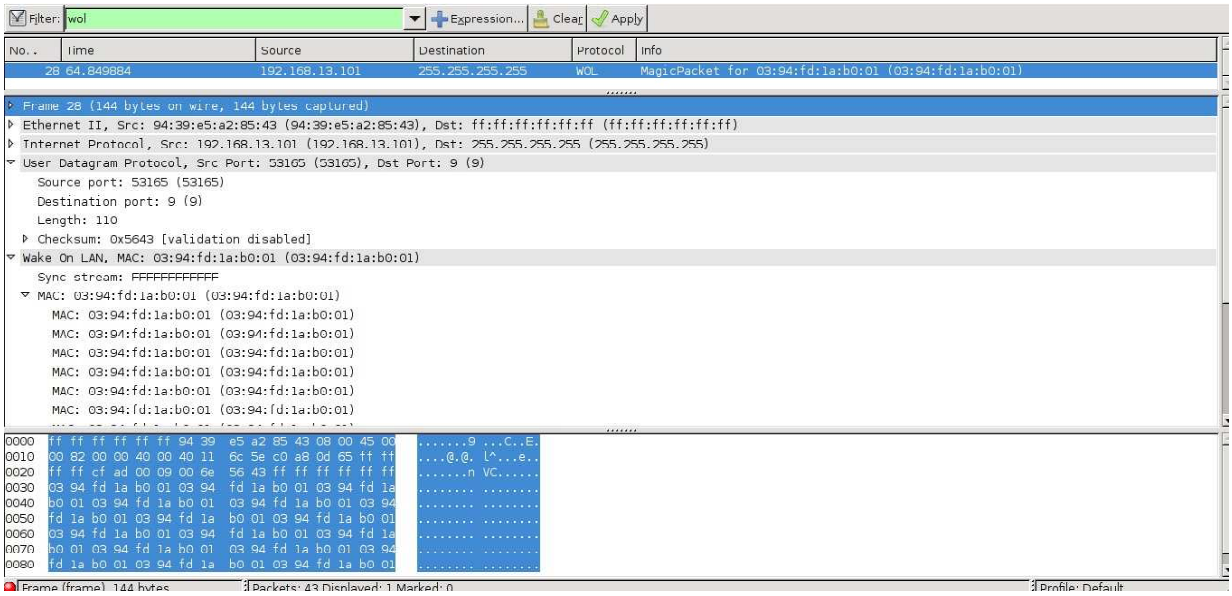

Figure 3: Captura de un MagicPacket

3.2.1 Configuración inicial de PC_A

Inicialmente será necesario activar en la BIOS del equipo a arrancar (PC_A = PC22) la opción Wake On LAN. Esta opción permite mantener encendida la interfaz de red cuando el equipo esté apagado. En algunas BIOS su activación puede ser evidente y en otras será necesario leer el manual.

Una vez activada la opción WoL en la BIOS, será necesario configurar la tarjeta de red para que acepte el MagicPacket y lo interprete correctamente.

Esto se puede hacer con la herramienta *ethtool*, que permite entre otras cosas habilitar la comprensión del MagicPacket.

```
PC22# apt-get install ethtool
PC22# ethtool -s eth0 wol g
```

El proceso puede ejecutarse en arranque, editando el fichero "/etc/network/interfaces":

```
iface eth0 inet dhcp
        post-up /sbin/ethtool -s $IFACE wol g
        post-down /sbin/ethtool -s $IFACE wol g
```

Esto ejecuta *ethtool* tras cada arranque de la interfaz eth0.

A partir de la distribución Squeeze, puede ponerse de la forma:

```
iface eth0 inet dhcp
        ethernet-wol g
```

En ese caso, el script */etc/network/if-up.d/ethtool* llamará a *ethtool* en el arranque de la interfaz (ifup).

3.2.2 Envío de mensajes WoL desde PC_B

Existen diversos productos de software para la generación y envío del MagicPacket.

Uno de ellos es el *etherwake*. Para ejecutarlo es necesario ser superusuario.

```
PC24# apt-get install etherwake
PC24# etherwake <MAC_PC22>
```

Otro de los productos es "wakeonlan", un programa que utiliza paquetes UDP. No necesita ejecutarse como root, y es mas configurable.

```
PC24# apt-get install wakeonlan && exit
PC24$ wakeonlan <MAC_PC22>
```

4 Bridging

Un "bridge" es un proceso implementado mediante hardware o software que cambia el encapsulado de capa 2 entre dos o mas interfaces. Las interfaces involucradas pueden pertenecer a la misma tecnología de enlace o a tecnologías diferentes. En el primer caso (misma tecnología de enlace) se puede incluir la creación de un conmutador o hub con varias interfaces ethernet, y en el segundo caso se pueden incluir los puntos de acceso Wifi (AP).

Cuando se crea un bridge, las interfaces que lo componen dejan de estar operativas a nivel IP, incluso aunque tengan direccionamiento establecido. Esto hace que sea necesario efectuar con mucho cuidado el proceso de creación del bridge si alguna de las interfaces que lo componen está operativa e incluso si tiene conexiones activas, particularmente conexiones IP. Es particularmente importante considerar la realización de bridges de forma remota como una práctica de riesgo, ya que si la conexión remota está establecida sobre alguna de las interfaces que se van a puentear, se perderá la conexión. En esos casos, es muy recomendable la configuración del bridge mediante scripting, restaurando las rutas por defecto usando como interfaz el bridge.

El proceso de creación de un bridge es:

- Identificación de las interfaces que se integrarán en él.
- Deshabilitación y borrado del direccionadmiento de las interfaces.
- Creación del bridge
  - Con brctl

    ```
    PCX# brctl addbr br0
    PCX# brctl addif br0 enp3s0
    ```

  - Con iproute2

    ```
    PCX$ sudo ip l add name bridge0 type bridge
    PCX$ ip a show dev bridge0
    3: bridge0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default ↩
        qlen 1000
    link/ether ba:f0:20:42:64:5f brd ff:ff:ff:ff:ff:ff
    PCX$ sudo ip l set enp3s0 address ba:f0:20:42:64:5f
    PCX$ sudo ip l set bridge0 address ac:22:0b:29:e6:0c
    PCX$ sudo ip l set enp3s0 master bridge0
    ```

- Configuración del bridge

  ```
  PCX# ip addr add 192.168.49.1X/24 dev br0
  ```

- Activación del bridge

  ```
  PCX# ip link set up dev br0
  ```

Un tipo de puente muy utilizdo es el que usan los AP para interconectar redes 802.11 con redes 802.3. Otro puente muy utilizado es el de los conmutadores.

5 Generación de tráfico

Cualquier aplicación de red, puede generar tráfico en la misma. La aplicación mas utilizada para la generación controlada de tráfico es la utilidad "ping", aunque con objetivos mas generales se utiliza también la utilidad *iperf* .

5.1 ping

Esta utilidad es un front-end del protocolo ICMP, en particular de los tipos ICMP 8 para enviar paquetes e ICMP 0 para devolverlos. Además de las páginas del manual, se puede encontrar información muy interesante en *http://linux-ip.net/html/tools-ping.html*.

El formato del comando es

```
PCX# ping <OPCIONES> <DIRECCION IP DESTINO>
```

El retorno del comando es una información valiosa sobre la conectividad del enlace IP entre origen y destino. En particular, se recibe información sobre el tiempo de ida y vuelta de un paquete (llamado RTT) y el volumen de paquetes perdidos. Hay que tener en cuenta que una falta de respuesta del comando ping puede tener su origen en cualquiera de los sentidos de la comunicación, de forma que la ruta de los paquetes hacia el destino esté libre de problemas, y el retorno nó.

Algunos ejemplos de uso:

- Generar paquetes de un tamaño determinado, con un valor TTL determinado

  El siguiente comando genera paquetes ICMP con un campo de datos de tamaño 200 octetos y un valor del campo Time to Live (TTL) de 25 a PC11.

  ```
  PC10# ping -s 200 -t 25 192.168.29.111
  ```

  En sistemas Debian 9 (strech) o superior, la opción "-t" tiene otra función. En ese caso, habría que sustituirla por "–ttl=N". Quedaría:

  ```
  PC10# ping -s 200 --ttl=25 192.168.29.111
  ```

  La opción de enviar paquetes con valores diferentes en el campo TTL permite conocer los routers intermedios hacia un destino concreto. Esta característica es la utilizada por los programas "traceroute, tracert" o similares. Se invita al alumno a hacer pruebas de localización de los routers hacia cualquier destino, por ejemplo *1.1.1.1*.

  Nota: el valor del TTL devuelto dependerá del valor asignado por defecto en el kernel del equipo que devuelve el ping (ICMP 0). Ese valor puede cambiarse en */proc/sys/net/ipv4/ip_default_ttl* de la forma siguiente:

  ```
  PCX# cat /proc/sys/net/ipv4/ip_default_ttl
  64
  PCX# echo 19 > /proc/sys/net/ipv4/ip_default_ttl
  PCX# cat /proc/sys/net/ipv4/ip_default_ttl
  19
  ```

  Cambiar el tamaño del campo de datos permite averiguar el MTU (Maximal Transfer Unit, o tamaño máximo del paquete) del enlace de red entre el origen y el destino. Para averiguarlo será necesario poner la etiqueta DF (Dont Fragment) de la cabecera IP a uno (DF = 1) incrementando los tamaños hasta que se devuelva un error. El comando siguiente envía paquetes de un tamaño de datos de 2000 octetos con la etiqueta DF puesta a uno.

  ```
  PC10# ping -s 2000 -M do 192.168.29.111
  ```

- Enviar paquetes ICMP a una velocidad determinada.

  El siguiente comando envía un paquete ICMP 8 de 500 octetos cada 10 milisegundos. Para poder enviar a periodos superiores a 0.2 segundos son necesarios permisos de administrador.

```
PC10# ping -i 0.01 -s 500 192.168.29.111
```

  En sistemas Debian 9 (strech) o superior, la opción "-i" precisa de números decimales separados por coma (en lugar de separado por punto). En ese caso, habría que ejecutar:

```
PC10# ping -i 0,01 -s 500 192.168.29.111
```

  Se deja al alumno/a el cálculo de la velocidad de envío en bits por segundo, y se invita a generar tráfico a una velocidad de 1Mbps.

- Generar un número limitado de paquetes a la máxima velocidad posible.

  La opción "-f" envía a una velocidad máxima (rondando los 100 paquetes por segundo, pero dependiendo del procesador), y presenta un "." por cada paquete enviado y un caracter de borrado por cada paquete recibido, de forma que al final sólo quedan tantos puntos como paquetes perdidos. Cuando se incluye un intervalo, se aplica su valor al periodo de envío.

```
PC10# ping -f -i 0.05 -c 400 -s 800 192.168.29.111
```

  El comando anterior envia 400 paquetes de un tamaño del campo de datos de 800 con un periodo de 50 milisegundos hacia PC11.

  La opción "-W" indica el tiempo de espera hasta dar por perdido el paquete (timeout) en segundos. Acepta decimales. Así, el comando

```
PC10# ping -W 0.1 -i 0.05 -c 400 -s 800 192.168.29.111
```

  considerará que un ping se ha perdido cuando no ha sido recibido el ICMP 8 en 0.1 segundos.

  Existen algunas distribuciones en las que la opción "-W" no parece que haga efecto. En ese caso, se puede utilizar directamente el comando "timeout" de bash:

```
PC10# timeout 0.1 ping -i 0.05 -c 400 -s 800 192.168.29.111
```

- Generar un paquete ICMP8 a un grupo de direcciones.

  Esto suele ser útil para comprobar la disponibilidad de equipos dentro de un segmento de red. Básicamente será necesario crear un script que genere secuencialmente un paquete ICMP8 a todos los equipos deseados. La salida del ping puede ser tratada. En el ejemplo se ofrece una salida "binaria" del estilo "up/down".

```
PC10$ for i in $(seq 110 130); do ping -q -c 1 192.168.29.$i > /dev/null; if [ $? -eq 0 ]; ↩
    then echo $i UP; else echo "$i DOWN"; fi; done
```

  El comando anterior envia de forma secuencial un paquete ICMP8 a cada IP de los rangos marcados y espera respuesta antes de enviar el siguiente ICMP8. Por ello no es conveniente utilizarlo de forma masiva con muchos equipos. Para ello se recomienda alguna alternativa, como el comando *fping* (ver 5.3).

Una forma elegante y artesanal de conocer la velocidad a la que se genera el tráfico, es modificando el script del principio de esta sección de la forma siguiente:

```
PCX# c2=0; while true; do c1=`cat /proc/net/dev |grep eth0 |tr -s " " |cut -d " " -f 11`; resta=`↩
    expr $c1 - $c2`; bits=`expr $resta \* 8`; echo "$bits bits/seg"; c2=$c1; sleep 1; clear; ↩
    done
```

Se deja al alumno/a la comprensión y/o modificación del mismo. Se propone a su vez ampliar el script para la visualización también de la velocidad de tráfico recibido. Otra propuesta está en la adaptación del script al comando *watch*.

Es importante conocer previamente el formato del fichero */proc/net/dev*, para poder escoger adecuadamente el número de campo a seleccionar con el parámetro "-f" del comando de parseo *cut* utilizando como delimitador de campo el espacio vacío (-d " "). En el caso de un kernel 2.6.34 y 3.2.9 sería "-f 11".

5.2 Generación de tráfico broadcast IP

El tráfico broadcast es útil cuando se pretende conocer qué equipos están operativos en una red o símplemente enviar una información a todos ellos. La dirección de broadcast IP es aquella en la que todos los bits del campo de host está a uno, es decir, la última de las direcciones IP disponibles de una red o subred.

En la red del laboratorio, un envío broadcast de paquetes ICMP sería:

```
PC10# ping -b 192.168.29.255
```

Para que un equipo Linux responda a una petición broadcast deberá tener desactivada la etiqueta de "ignore broadcast" que por defecto viene activada en kernel. Para ello ejecutar:

```
PC10# echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Para que los cambios sean permanentes:

```
PC10# echo "net/ipv4/icmp_echo_ignore_broadcasts=0" >> /etc/sysctl.conf
```

Se pueden visualizar las respuestas con los comandos *tcpdump* (sección 6) o con *wireshark*, pero una buena forma de comprobar el número de equipos que respondieron al broadcast sería consultando la tabla ARP (ver 8.2).

5.3 fping

Para poder enviar paquetes ICMP8 a un grupo de direcciones se puede utilizar el comando *fping*. Básicamente, para enviar un ICMP8 una sola vez a una subred eliminando salidas estadísticas, se haría:

```
PC10$ fping -q -a -c 1 -g 192.168.29.0/24
```

Se invita a la interpretación de la salida.

5.4 iperf

iperf es una aplicación cliente-servidor para linux, windows y mac. Genera tráfico entre el cliente y el servidor y presenta estadísticas sobre su velocidad, pérdidas y retardos.

El ejemplo mas sencillo es el de averiguar la máxima velocidad de transferencia entre dos equipos en una misma LAN.

En el lado servidor, por ejemplo PC20:

```
PC20$ iperf -s
```

En el lado cliente, por ejemplo PC22:

```
PC22$ iperf -c 192.168.29.120
```

El resultado se presenta tras algunos segundos:

```
PC20$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
```

```
------------------------------------------------------------
[  4] local 192.168.29.120 port 5001 connected with 192.168.29.122 port 49342
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.3 sec   115 MBytes  94.1 Mbits/sec
```

```
    PC22$ iperf -c 192.168.29.120
------------------------------------------------------------
Client connecting to 192.168.13.2, TCP port 5001
TCP window size: 23.5 KByte (default)
------------------------------------------------------------
[  3] local 192.168.29.122 port 49342 connected with 192.168.29.120 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec    115 MBytes  96.2 Mbits/sec
```

En el caso anterior se midió el tráfico en una sola dirección (de cliente a servidor). Utilizando la opción "-d" en el cliente puede medirse en ambos sentidos.

Para medir conexiones UDP se utiliza la opción -u en ambos lados. En ese caso se tiene la posibilidad de generar tráfico con una cadencia concreta. Por ejemplo, para medir la conexión de tráfico UDP a una cadencia de 10Mbits/seg con resultados cada segundo:

```
PC20$ iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.29.120 port 5001 connected with 192.168.29.122 port 41195
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec  1.19 MBytes  10.0 Mbits/sec   0.068 ms    0/  850 (0%)
[  3]  1.0- 2.0 sec  1.19 MBytes  10.0 Mbits/sec   0.080 ms    0/  851 (0%)
[  3]  2.0- 3.0 sec  1.19 MBytes  10.0 Mbits/sec   0.084 ms    0/  850 (0%)
[  3]  3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec   0.099 ms    0/  850 (0%)
[  3]  4.0- 5.0 sec  1.19 MBytes  10.0 Mbits/sec   0.076 ms    0/  851 (0%)
[  3]  5.0- 6.0 sec  1.19 MBytes  10.0 Mbits/sec   0.129 ms    0/  850 (0%)
[  3]  6.0- 7.0 sec  1.19 MBytes  10.0 Mbits/sec   0.139 ms    0/  851 (0%)
[  3]  7.0- 8.0 sec  1.19 MBytes  10.0 Mbits/sec   0.137 ms    0/  850 (0%)
[  3]  8.0- 9.0 sec  1.19 MBytes  10.0 Mbits/sec   0.201 ms    0/  850 (0%)
[  3]  9.0-10.0 sec  1.19 MBytes  10.0 Mbits/sec   0.253 ms    0/  851 (0%)
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec   0.240 ms    0/ 8504 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

```
    PC22$ iperf -c 192.168.29.120 -u -b 10m
------------------------------------------------------------
Client connecting to 192.168.29.120, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:   224 KByte (default)
------------------------------------------------------------
[  3] local 192.168.29.122 port 41195 connected with 192.168.29.120 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec
[  3] Sent 8505 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec  0.239 ms    0/ 8504 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

6 Visualización del tráfico. TCPdump

TCPdump es una herramienta en línea de comandos que permite monitorizar todo el tráfico entrante o saliente de cualquiera de las interfaces de red donde se ejecuta. Es una alternativa por línea de comandos al analizador de tramas "wireshark", y es muy utilizada cuando se pretende monitorizar el tráfico en un equipo remoto.

TCPdump dispone de múltiples opciones en la línea de comandos, en general para filtrar los paquetes que se presentan y cambiar la visualización del contenido de los mismos. El programa tiene que ejecutarse con permisos de root. Algunos ejemplos se presentan a continuación:

- Ejemplo 1

```
PC10# tcpdump -w test.pcap -i eth0 tcp port 80
```

En este ejemplo se escucha en la interface eth0, se salva lo capturado en el fichero "test.pcap", y se presentan sólo los paquetes que tienen como origen o destino el puerto 80. La opción -w sólo guarda los datos en el fichero, sin presentar resultados en pantalla. El fichero "test.pcap" no es editable con editores estándar, y sólo se puede visualizar con "wireshark" o con el propio "tcpdump" haciendo

```
PC10# tcpdump -r test.pcap
```

- Ejemplo 2

```
PC10# tcpdump -i eth0 tcp port 80 or udp port \( 161 or 162 \)
```

Este ejemplo es igual que el anterior, pero en este caso no se guardan los resultados en ningún fichero y se capturan paquetes que tengan como origen o destino los puertos tcp 80 o udp 161 o 162.

- Ejemplo 3

```
tcpdump -A -s 1550 dst 192.168.10.14 or src 192.168.10.173 and tcp port 22
```

Este ejemplo captura 1550 octetos en cada paquete (en lugar de los 96 octetos que se capturan por defecto) que tengan como IP destino *192.168.10.14* o IP origen *192.168.10.173* y como origen o destino el puerto 22. La opción "-A" indica que se presenten en modo ASCII.

7 Emulación de red

Existen numerosas herramientas emuladoras y simuladoras de redes. Posiblemente la mas sencilla sea la herramienta *netem* basada en el sistema de control del tráfico de Linux *tc*. El uso y comprensión de esta herramienta se deja a voluntad del alumno.

8 Herramienta "iproute2"

8.1 ip link

El subcomando "ip link" permite la visualización y el control de las interfaces y los enlaces disponibles en el dispositivo. También puede cambiar parámetros avanzados, como la dirección de broadcast, el tipo y tamaño de la cola de la interfaz o la dirección MAC. Es por tanto un comando fundamental de la capa 2 de enlace.

La ejecución del comando sin parámetros muestra el estado de las interfaces.

```
PC10:/home/alumno# ip link
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast qlen 1000
   link/ether 00:13:77:35:b4:c1 brd ff:ff:ff:ff:ff:ff
3: wifi0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 199
    link/ieee802.11 00:16:e3:cf:4e:92 brd ff:ff:ff:ff:ff:ff
4: ath0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc noqueue
    link/ether 00:16:e3:cf:4e:92 brd ff:ff:ff:ff:ff:ff
5: sit0: <NOARP> mtu 1480 qdisc noop
   link/sit 0.0.0.0 brd 0.0.0.0
```

La salida presenta el nombre de cada interfaz reconocida por el sistema. Algunos de los campos de la salida son:

- BROADCAST,MULTICAST: el dispositivo puede enviar tráfico broadcast y tiene el tráfico multicast habilitado.
- UP: si el dispositivo tiene este campo es que está habilitado, si nó, está deshabilitado
- mtu: maximal transfer unit. Tamaño máximo del paquete antes de ser fragmentado.
- qdisc, qlen: tipo y longitud de la cola. En el caso de eth0 es la cola habitual qdisc tipo fifo con una longitud de 1000 paquetes.

- link/ether, brd: tecnología de enlace y dirección de broadcast. A continuación viene la dirección MAC de enlace y la dirección de broadcast

Algunos ejemplos de uso habitual:

- Activar/desactivar una interfaz.

```
PCX# ip link set up dev eth0
PCX# ip link set down dev eth0
```

- Activar/desactivar tráfico multicast

```
PCX# ip link set dev eth0 multicast off
PCX# ip link set dev eth0 multicast on
```

El tráfico multicast puede comprobarse enviando un "ping" a la dirección multicast de todos los dispositivos de la red local (*224.0.0.1*). Se invita a hacer pruebas de envío cuando esté habilitado y cuando esté inhabilitado el envío y recepción de tráfico multicast tanto en el equipo local como en el host contrario.

- Cambiar el nombre de la interfaz

```
PCX# ip link set dev eth0 name local0
```

Es muy conveniente incluir el cambio de nombre de la interfaz en la configuración inicial de cualquier sistema, para tener en cuenta el nombre en cualquier script de funcionamiento que haga uso de él.

NOTA SOBRE LOS NOMBRES DE INTERFAZ:

En versiones recientes los nombres de interfaz en Linux no siguen el criterio de "tecnología" y "números consecutivos" como "eth0" para la primera interfaz ethernet. El criterio actual de asignación de nombres de interfaz es el "Predictable Interface Names" que consiste en:

1. Nombres que incorporan índices para las interfaces integradas en placa base según lo detectado en BIOS. Por ejemplo: "en01"; "en" ethernet, "01" primera interfaz.
2. Nombres que incorporan índices para interfaces en el bus PCIExpress según detectado en BIOS. Por ejemplo: "ens1"; "en" ethernet, "s1" PCIExpress interfaz 1.
3. Nombres que incorporan la ubicación del conector del hardware. Por ejemplo: "enp2s0"; "en" ethernet, "p2" bus 2, "s0" slot 0.
4. Nombres que incorporan la dirección MAC de la interfaz. Por ejemplo: "enx89ae45be5601".
5. Método clásico de asignación "ethX" con números consecutivos.

Este proceso puede configurarse en */etc/udev/rules.d/80-net-setup-link.rules* o en */etc/udev/rules.d/80-net-name-slot.rules* dependiendo de la versión.

En el caso en que se siga prefiriendo la gestión de nombres clásica "ethX", conviene mirar algún método fijo de asignación de nombres en */etc/udev/rules.d/70-persistent-net.rules*, gestionado por el script */lib/udev/write_net_rules*.

Puede leerse la razón de este cambio en
https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/: "The classic naming scheme for network interfaces applied by the kernel is to simply assign names beginning with "eth" to all interfaces as they are probed by the drivers. As the driver probing is generally not predictable for modern technology this means that as soon as multiple network interfaces are available the assignment of the names is generally not fixed anymore and it might very well happen that"eth" on one boot ends up being "eth1" on the next. This can have serious security implications..."

- Cambiar dirección MAC y de broadcast.

```
PCX# ip link set dev eth0 address 00:80:c8:f8:be:ef
PCX# ip link set dev eth0 broadcast ff:00:ff:00:ff:00
```

Algunas precauciones:

- Para cambiar el nombre y la dirección MAC (así como otros parámetros) es necesario que la interfaz esté desactivada.
- Los protocolos de enlace no contemplan la posibilidad de direcciones MAC duplicadas, por lo que hay que tener en cuenta que la MAC que se ponga sea única en la red local en la que se encuentre el host.
- Si se cambia la dirección de broadcast, es muy posible que el protocolo ARP no funcione y el equipo tenga serios problemas en disponer conectividad a los equipos locales.

Una descripción mas detallada del comando y de sus posibilidades se encuentra en *http://linux-ip.net/html/tools-ip-link.html* y en la documentación general de "iproute2"

8.2 ip neigh

Permite gestionar la caché ARP de forma mas eficiente que el clásico comando *arp*. Básicamente permite añadir, eliminar y visualizar los dispositivos que el equipo que ejecuta el comando tiene o cree tener como vecinos de la LAN. La consideración de vecino puede deberse en este caso a que, además de estar en el mismo segmento LAN (dominio de broadcast), en algún momento han transferido datos entre ellos.

Visualización de vecinos (estado de la caché ARP) *ip neigh*, *ip neigh show*, *ip neigh ls*:

```
PCX#:/etc# ip neigh show
192.168.29.1 dev eth0 lladdr dc:0e:a1:16:6b:06 DELAY
192.168.29.123 dev eth0 lladdr 88:30:8a:43:c9:4e REACHABLE
192.168.29.119 dev eth0 lladdr b8:70:f4:a1:02:07 STALE
192.168.29.124 dev eth0  FAILED
192.168.29.71 dev eth0 lladdr 00:55:66:77:aa:bb PERMANENT
.......
```

La salida indica la dirección IP y la dirección MAC de los vecinos, además del estado de la comunicación. Algunos estado son:

- REACHABLE: el registro está activo. No se ejecutará el protocolo ARP.
- DELAY: el Kernel ha enviado un paquete para mantener la comunicación y está esperando por una confirmación.
- STALE: el registro está en espera. Considera que el registro es correcto hasta nueva comunicación. Básicamente indica que hace tiempo que no hay comunicación entre los equipos y que el Kernel enviará un paquete en breve.
- FAILED: el registro está perdido. Alguna vez existió pero ahora o se ha eliminado manualmente o ha fallado un intento de comunicación. Se ejecutará ARP cuando vuelvan a intentar comunicarse.

Para eliminar un registro de la caché ARP se ejecuta *ip neigh delete*

```
PCX#:/etc# ip neigh delete 192.168.29.123 dev eth0
```

Para añadir registros de forma permanente a la caché ARP se ejecuta *ip neigh add*

```
root@urano:/home/cacho/scripts# ip neigh add 192.168.29.71 lladdr 00:55:66:77:AA:BB dev eth0
root@urano:/home/cacho/scripts# ip neigh
192.168.29.71 dev eth0 lladdr 00:55:66:77:aa:bb PERMANENT
............
```

Para reemplazar los valores de un registro de la caché ARP (puede ser necesaio si no deja añadir al existir un registro aunque sea FAILED), se ejecuta *ip neigh replace*

```
root@urano:/home/cacho/scripts# ip neigh add 192.168.29.71 lladdr 00:55:66:77:AA:BB dev eth0
RTNETLINK answers: File exists
root@urano:/home/cacho/scripts# ip neigh replace 192.168.29.71 lladdr 00:55:66:77:AA:BB dev eth0
root@urano:/home/cacho/scripts# ip neigh
192.168.29.71 dev eth0 lladdr 00:55:66:77:aa:bb PERMANENT
............
```

Para añadirlos de forma temporal será necesario establecer comunicación con el vecino, por ejemplo ejecutando un *ping*.

8.3 ip addr

Este comando permite añadir, eliminar o cambiar direcciones IP asociadas a una interfaz.

La ejecución del comando sin opciones permite visualizar el direccionamiento IP de las interfaces del equipo.

```
PC10# ip addr
```

Se crearán diferentes redes IP por pares de equipos. La dirección de red será la clase privada tipo C 192.168.<PC MENOR>.0, es decir si se agrupan en una red IP los equipos PC10 y PC11, la red sería *192.168.10.0* con máscara *255.255.255.0* (/24).

- Alta de una nueva dirección

```
PC10# ip addr add 192.168.10.110/24 dev eth0
```

- Baja de una dirección

```
PC10# ip addr del 192.168.29.110/24 dev eth0
```

- Interfaces virtuales.

  Utilizando iproute2 no son necesarias interfaces virtuales para disponer de mas de una dirección IP por interfaz. Disponer de mas de una dirección IP por interfaz implica símplemente añadir la dirección IP utilizando "ip addr".

```
PC10# ip addr add 192.168.10.110/24 dev eth0
PC10# ip addr add 192.168.20.110/24 dev eth0
PC10# ip addr add 192.168.30.110/24 dev eth0
PC10# ip addr
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:13:77:35:b4:c1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.110/24 brd 192.168.10.255 scope global eth0
    inet 192.168.20.110/24 brd 192.168.20.255 scope global eth0
    inet 192.168.30.110/24 brd 192.168.30.255 scope global eth0
3: sit0: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
```

Una descripción mas detallada del comando y de sus posibilidades se encuentra en "http://linux-ip.net/html/tools-ip-address.html" y en la documentación general de "iproute2"

8.4 ip route

Este subcomando se utiliza para la gestión de las tablas de rutas del host, pudiendo convertirlo en un auténtico router. En esta práctica se utilizará exclusivamente para la gestión de la ruta por defecto, la que permite la salida a cualquier red que no esté especificada y por tanto a Internet. A esta red no especificada se la conoce como *red por defecto* y queda representada por la dirección IP *0.0.0.0/0*; a la ruta que lleva a ella se la conoce habitualmente como *ruta por defecto* o *default route*.

La salida a la red por defecto se realiza por lo que habitualmente se llama *gateway por defecto* o *default gateway* y suele ser un router, firewall u otro dispositivo que es dado por el gestor de la red.

Para dar de alta la ruta por defecto, será necesari conocer el gateway por defecto. En el caso de la red del laboratorio es el firewall F0 con dirección IP *192.168.29.1*.

El comando para hacerlo sería:

```
PC10# ip route add <RED POR DEFECTO> via <GATEWAY POR DEFECTO> dev <INTERFACE DE SALIDA>
```

Y mas concretamente:

```
PC10# ip route add default via 192.168.29.1 dev eth0
```

En el comando anterior puede verse el uso de la palabra clave "default" para representar la red por defecto, aunque puede utilizarse la dirección IP de esa red (*0.0.0.0/0*), quedando:

```
PC10# ip route add 0.0.0.0/0 via 192.168.29.1 dev eth0
```

Si ya existiese una red por defecto configurada, los anteriores comandos dan una salida del tipo:

```
PC10# ip route add default via 192.168.29.1 dev eth0
RTNETLINK answers: File exists
```

por lo que habrá que eliminar la ruta previamente configurada y después volver a dar de alta la ruta por defecto deseada.

```
PC10# ip route del default
PC10# ip route add default via 192.168.29.1 dev eth0
```

8.5 ip rule

El sistema de enrutamiento avanzado de Linux permite el manejo de varias tablas de enrutamiento, y por tanto se podrían seleccionar diferentes rutas para un mismo destino. Este sería el caso de una red con dos (o mas) salidas a Internet (o a cualquier otro destino), y por tanto con dos rutas por defecto. Una de esas rutas estaría en una tabla de enrutamiento y la otra ruta en otra tabla de enrutamiento. La práctica dedicada a routing utiliza esta característica.

Las tablas pueden visualizarse con el comando *ip route*

El kernel de Linux gestiona esas tablas con el comando *ip rule* .

8.6 Otros

El comando *ip* permite ejecutar otras funcinalidades, como estadísticas. Se invita al alumno/a a investigarlo.

9 netstat

El comando *netstat* presenta diversa información de red, como las conexiones, tablas de rutas, estadísticas o miembros multicast entre otras.

Una de las informaciones mas interesantes y exclusivas es la identificación de los procesos que hacen uso de los puertos actualmente abiertos. Esta información está disponible utilizando el parámetro "-p", que se puede combinar con otras opciones. Esta opción añade el nombre de programa o PID del proceso a la salida por pantalla, muy util para identificar qué programa está usando un determinado puerto.

Para visualizar los puertos TCP utilizados con el nombre del proceso que los utiliza se puede ejecutar el comando siguiente:

```
alumno@PC10:~$ netstat -pt
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 PC10.local:50543        lis01s06-in-f21.1:https TIME_WAIT   -
tcp        0      0 PC10.local:50569        lis01s06-in-f21.1:https ESTABLISHED 1816/firefox
tcp        0      0 PC10.local:36428        78.136.107.219:https    ESTABLISHED 1816/firefox
```

Para visualizar los puertos UDP utilizados con el nombre del proceso que los utiliza se puede ejecutar el comando siguiente:

```
alumno@PC10:~$ netstat -aup
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 *:46582                 *:*                                 1036/rpc.statd
udp        0      0 *:54828                 *:*                                 1443/avahi-daemon:
udp        0      0 *:39461                 *:*                                 1338/snmpd
udp        0      0 *:domain                *:*                                 3717/dnsmasq
udp        0      0 *:bootps                *:*                                 3717/dnsmasq
udp        0      0 *:tftp                  *:*                                 3717/dnsmasq
udp        0      0 *:sunrpc                *:*                                 1024/portmap
udp        0      0 localhost:snmp          *:*                                 1338/snmpd
udp        0      0 *:ipp                   *:*                                 1497/cupsd
udp        0      0 *:788                   *:*                                 1036/rpc.statd
udp        0      0 *:mdns                  *:*                                 1443/avahi-daemon:
udp6       0      0 [::]:60173              [::]:*                              1443/avahi-daemon:
udp6       0      0 [::]:domain             [::]:*                              3717/dnsmasq
udp6       0      0 [::]:mdns               [::]:*                              1443/avahi-daemon:
```

10 NMAP

Nmap es un programa de escaneo de dispositivos y puertos para Linux. Se ejecuta desde línea de comandos y es utilizado por numerosos front-ends para el descubrimiento de dispositivos y servicios en redes alcanzables. La pagina web del producto es "http://nmap.org".

Su instalación se puede realizar desde los repositorios habituales con el comando *apt-get install* .

```
    PC10# apt-get install nmap
```

Nmap dispone de numerosas opciones, y se puede ejecutar como usuario normal o como root. Algunos de los escaneos mas utilizados son:

10.1 Descubrimiento de equipos

```
    PC10# nmap -sP 192.168.10.0/24

Starting Nmap 4.62 ( http://nmap.org ) at 2011-09-06 18:23 CEST
Host 192.168.10.1 appears to be up.
Host PC18 (192.168.10.118) appears to be up.
Host 192.168.10.19 appears to be up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 13.845 seconds
```

10.2 Descubrimiento de equipos y servicios

Si al comando nmap no se le dan opciones, este descubre equipos y servicios TCP (de puertos menores que el 1024) en el rango de IP's que se le pasa como argumento.

```
    PC10# nmap 192.168.10.0/24

Starting Nmap 4.62 ( http://nmap.org ) at 2011-09-06 18:25 CEST
Interesting ports on 192.168.10.1:
Not shown: 1712 filtered ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
515/tcp open  printer

Interesting ports on PC18 (192.168.10.11):
Not shown: 1709 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
111/tcp open  rpcbind
113/tcp open  auth
631/tcp open  ipp

Interesting ports on 192.168.10.12:
Not shown: 1712 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
```

```
111/tcp open  rpcbind

Nmap done: 256 IP addresses (3 hosts up) scanned in 20.733 seconds
```

10.3 Descubrimiento de todos los servicios (TCP) de un equipo

Existen varios métodos de escaneo de servicios en un equipo. Será necesario visualizar el manual para conocer todas las opciones. Por defecto, el método utilizado es el TCP SYN, pues es muy poco intrusivo y permite realizar escaneos muy rápidos.

El método TCP SYN consiste en enviar una solicitud de conexión TCP mediante el envío de un paquete SYN y esperar por la respuesta. Si se recibe un SYN/ACK es que el puerto (y por tanto el servicio asociado) está disponible; si se recibe un RST significa que el puerto está cerrado (y por tanto el servicio no está disponible), si no se recibe respuesta, es que el puerto (y/o posiblemente el equipo) está filtrado.

```
PC10:# nmap -p 1-65535 192.168.10.11

Starting Nmap 4.62 ( http://nmap.org ) at 2011-09-07 13:42 CEST
Interesting ports on 192.168.10.11:
Not shown: 65528 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
113/tcp   open  auth
631/tcp   open  ipp
56944/tcp open  unknown
MAC Address: 00:13:77:35:B4:C1 (Samsung Electronics CO.)

Nmap done: 1 IP address (1 host up) scanned in 3.239 seconds
```

10.4 Descubrimiento de todos los equipos con un servicio

Para conocer todos los equipos de una red que tienen servicio web, se procedería como sigue:

```
PC10:# nmap -p 80 192.168.10.0/24

Starting Nmap 4.62 ( http://nmap.org ) at 2011-09-07 13:42 CEST
Interesting ports on 192.168.10.11:
PORT   STATE    SERVICE
80/tcp open http
MAC Address: 00:1E:8C:16:E2:BF (Asustek Computer)

Interesting ports on 192.168.10.12:
PORT   STATE  SERVICE
80/tcp closed http
MAC Address: 00:13:77:35:B4:C1 (Samsung Electronics CO.)

Interesting ports on 192.168.10.13:
PORT   STATE  SERVICE
80/tcp open http
MAC Address: 00:13:8F:C1:8E:78 (Asiarock Incorporation)

Nmap done: 256 IP addresses (3 hosts up) scanned in 1.092 seconds
```

10.5 Descubrimiento de servicios UDP

Para el descubrimiento de servicios UDP son necesarios permisos de root. Descubrir puertos UDP en un rango de equipos o una subred suele tardar mucho tiempo, por lo que es recomendable especificar el equipo y un rango pequeño de puertos. Un ejemplo sería:

```
    PCX# nmap -sU -p U:1-200 192.168.10.11/24

tarting Nmap 4.62 ( http://nmap.org ) at 2011-09-06 19:21 CEST
Interesting ports on 192.168.10.11:
Not shown: 197 closed ports
PORT    STATE         SERVICE
111/udp open|filtered rpcbind
161/udp open|filtered snmp
162/udp open|filtered snmptrap
```

```
MAC Address: 00:02:A5:22:1E:1F (Compaq Computer)

Nmap done: 1 IP address (1 host up) scanned in 199.471 seconds
```

Pueden verse los puertos "SNMP", además de que este escaneo duró la nada despreciable cifra de casi 200 segundos.

11 Netcat

Netcat (habitualmente `nc`) es una utilidad avanzada que permite la creación de servicios de red y la comunicación mediante arquitectura cliente-servidor. Las posibilidades de uso son múltiples, y es utilizada habitualmente por administradores de redes para abrir puertos traseros de forma provisional o auditorías. Se invita al alumno a conocer diferentes opciones del comando visualizando la página del manual `man nc`.

En esta práctica `nc` se utilizará para crear un servicio de envío de datos hacia un host de monitorización. Como ejemplo se servirán como datos por parte del host del usuario el número de octetos transmitidos y recibidos de la interfaz "eth0", y se visualizará en el host remoto. En los ejemplos se utilizan los hosts PC10 y PC11.

11.1 Captura de datos

Los datos de consumo de la interfaz eth0 pueden visualizarse en el fichero "/proc/net/dev", apareciendo los datos transmitidos, recibidos, errores y otros por cada una de las interfaces.

```
PCD10:/home/alumno# cat /proc/net/dev
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo ↩
      colls carrier compressed
    lo:193490188  523038    0    0    0     0          0         0 193490188  523038    0    0    0 ↩
             0       0          0
  eth0:470288778 2635921    0    0    0     0          0         0 284249522  697356    0    0    0 ↩
           0       0          0
```

Por tanto será necesario parsear en esa salida los datos que se necesitan. Para ello se creará un script que los presenta, llamado "consumo_eth0.sh", y cuyo contenido sería (dependiendo de cada sistema)

```
#! /bin/sh

eth0_in=`cat /proc/net/dev |grep eth0 |tr -s " " |cut -d " " -f 2 |cut -d ":" -f 2`
eth0_out=`cat /proc/net/dev |grep eth0 |tr -s " " |cut -d " " -f 10`

echo $eth0_in $eth0_out"
```

La ejecución de este script daría

```
PC10:/home/alumno# chmod 755 consumo_eth0.sh
PC10:/home/alumno# ./consumo_eth0.sh
470630820 284356538
```

Es decir, el primer dato presentado sería el número de octetos recibido y el seguno el número de octetos transmitido.

Los datos se pueden servir de forma puntual o de forma permanente.

11.2 Servicio TCP o UDP puntual

El uso de "netcat" para servir datos a través de un puerto TCP podría realizarse de la siguiente forma:

```
    PC10# consumo_eth0.sh | nc -l -p 6660
```

En este caso, los datos se sirven en el puerto tcp 6660, que debería estar libre antes de la ejecución de `nc`.

Si se quisiera servir por el puerto libre udp 6660, se haría:

```
    PC10# consumo_eth0.sh | nc -l -u -p 6660
```

11.3 Servicio TCP o UDP permanente

En este caso, deberá servirse el dato por parte del sistema remoto (en este caso PC10), en un puerto libre de forma permanente como servicio. Esto se puede hacer directamente con el comando "netcat" utilizando el fichero "/etc/inetd.conf". Si se desea disponer de un servidor de esos datos de forma contínua, habrá que realizar dos pasos:

1. En el fichero "/etc/inetd.conf" habrá que añadir el servicio de entrega de los datos del script, de la forma:

```
.............
#:OTHER: Other services
nc_monit        stream  tcp     nowait  root    /home/alumno/consumo_eth0.sh
nc_monit        dgram   udp     nowait  root    /home/alumno/consumo_eth0.sh
.............
```

2. Al final del fichero "/etc/services" deberán ponerse las líneas que relacionan al servicio con un puerto, de la forma:

```
.............
nc_monit    6660/tcp    # monitorizacion de consumo de interfaces
nc_monit    6660/udp    # monitorizacion de consumo de interfaces
.............
```

11.4 Recepción de datos

Si PC10 sirve los datos, para que PC11 reciba los datos se utilizará también "nc" pero con otra sintaxis.
Si los datos se sirven en PC10 en el puerto tcp 6660

```
PC11# nc 192.168.10.110 6660
```

Si los datos se sirven en PC10 en el puerto udp 6660

```
PC11# nc 192.168.10.110 -u 6660
```

11.5 Otros usos

Un ejemplo habitual del uso de netcat es para la transmisión de ficheros entre dos equipos. Se presenta el ejemplo anterior de transmisión de datos de consumo de interfaces pero en forma de transmisión de ficheros. Es bueno resaltar que para transmitir ficheros es muy conveniente utilizar puertos tipo TCP, pues estos permiten la retransmisión de paquetes en casos de pérdidas, y los puertos UDP no, con lo que los ficheros podrían no recibirse correctamente.
Se crea como ejemplo la transmisión del fichero de consumos de interfaces de PC10 a PC11.
En PC10 se ejecuta:

```
PC10# nc -l -p 6660 < /proc/net/dev
```

En PC11 se ejecuta:

```
PC11# nc 192.168.10.110 6660 > monit_PC11
```

12 socat

Es un programa de manipulación de sockets mas avanzado que netcat. Crea sockets para la conexión de flujos de datos entre diferentes equipos (o el mismo). Puede conectar:

- ficheros

- colas
- dispositivos
- sockets (Unix, IPv4, IPv6, RAW, UDP, TCP)
- sockets SSL
- conexiones Proxy CONNECT
- descriptores de ficheros (stdin, etc)
- el editor de linea GNU
- programas
- combinaciones de dos de las anteriores

Algunos de sus principales usos podrían ser:

- TCP port forwarder
- Seguridad y auditoría
- interface shell a sockets unix.
- conectar puertos serie de diferentes equipos entre sí.
- relay IPv6
- enviar flujos TCP a puertos serie
- crear entornos seguros (su y chroot) para shell scripts de servidor o cliente con conexiones de red.

El comando socat tiene una relativamente sencilla sintaxis, una vez se tiene claro lo que se pretende hacer:

```
PCX# socat [options] <address> <address>
```

donde <address> es una terna "protocolo:ip:puerto"
Ejemplos:
1. Conectar con el puerto 80 en un equipo local o remoto

```
PCX# socat - TCP4:www.example.com:80
```

donde el símbolo "-" significa STDIO (es decir, la entrada de teclado en este caso). Este ejemplo sería similar a hacer "telnet sobre "www.example.com:80"
2. socat como TCP port forwarder

```
PCX# socat TCP4-LISTEN:81 TCP4:192.168.29.110:80
```

En este ejemplo, se rederige el puerto de escucha local 81 (TCP4-LISTEN:81) hacia TCP4:192.168.29.110:80.
Para conexiones múltiples se usa la opción "fork". Por ejemplo:

```
PCX# socat TCP4-LISTEN:81,fork,reuseaddr TCP4:TCP4:192.168.1.10:80
```

Este ejemplo escucha en el puerto 81, acepta conexiones, y redirige las conexiones al puerto 80 del equipo remoto.

```
PCX# socat TCP-LISTEN:3307,reuseaddr,fork UNIX-CONNECT:/var/lib/mysql/mysql.sock
```

El ejemplo anterior escucha en el puerto 3307, acepta conexiones y las redirige a un socket Unix del equipo remoto.
3. Colector de mensajes de red:

```
PCX# socat -u TCP4-LISTEN:3334,reuseaddr,fork OPEN:/tmp/test.log,creat,append
```

En ese ejemplo, si un cliente se conecta al puerto 3334, se crea un nuevo proceso. Todos los datos enviados por los clientes se registran en el fichero /tmp/test.log. Si el fichero no existe, lo crea. La opción reuseaddr permite el reinicio del proceso del servidor.

4. Envía un broadcast a la red local:

```
PCX# socat - UDP4-DATAGRAM:224.255.0.1:6666,bind=:6666,ip-add-membership=224.255.0.1:eth0
```

En este caso, socat tranfiere los datos del STDIN (teclado) a la dirección multicast utilizando UDP sobre el puerto 6666 tanto para la conexión local como remota. El comando también habilita a la interfaz eth0 a aceptar paquetes del grupo multicast especificado.

Otros ejemplos se pueden encontrar en: *http://www.dest-unreach.org/socat/doc/socat.html#EXAMPLES*

13 Port Knocking

El "PortKnocking" es una fórmula para escuchar si se está intentando acceder a un puerto aunque este esté cerrado, y abrirlo bajo demanda.

En realidad el sistema "Port Knocking" es mucho mas genérico, resumiéndose en un demonio que espera una secuencia determinada de paquetes IP y que realiza una acción programada cuando llegan.

Suele utilizarse como medida adicional de seguridad, y es común encontrarla para permitir la apertura del puerto *ssh* 22 y evitar intentos de intrusión por fuerza bruta cuando un puerto está abierto.

Como ejemplo, se muestra cómo abrir el puerto *ssh* 22 en Debian, bajo demanda, utilizando el demonio *knockd*. Este demonio se instala desde el repositorio de Debian mediante *apt-get install knockd*, aunque ya está instalado en los equipos del laboratorio.

Lo que hace el demonio *knockd* es esperar que llegue una secuencia de paquetes TCP identificados de alguna forma (por ejemplo con el flag de SYN) a unos puertos previamente definidos (pero que evidentemente no tienen que estar abiertos) antes de un tiempo determinado (también predefinido). Cuando llega esa secuencia, se ejecuta una acción, que podría ser añadir una regla de iptables para permitir el acceso al servicio SSH a la IP que ha hecho el Port Knocking. Y con otra secuencia, se puede cerrar el puerto posteriormente.

Para habilitar *knockd* hay que editar el fichero */etc/default/knockd* y cambiar sólo algunas opciones (p.ej. la interfaz de escucha o si se quiere que se ejecute en el arranque):

```
# PLEASE EDIT /etc/knockd.conf BEFORE ENABLING
START_KNOCKD=1
.............
# command line options
# KNOCKD_OPTS="-i eth1"
.............
```

El fichero de configuración de *knockd* es */etc/knockd.conf*, debería tener un aspecto mas o menos como:

```
.............
[openSSH]
        sequence    = 7000,8000,9000
        seq_timeout = 5
        command     = /sbin/iptables -A INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn

[closeSSH]
        sequence    = 9000,8000,7000
        seq_timeout = 5
        command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn
.............
```

Esta configuración muestra dos secuencias; la primera, llamada [openSSH]: al recibir un intento de conexión (paquete SYN, *tcpglags=syn*) a los puertos 7000, 8000 y 9000 de forma consecutiva (*sequence=7000,8000,9000*) y en menos de 5 segundos (*seq_timeout=5*) se ejecuta el comando definido en *command*, que en este caso es la creación de una regla de *iptables* en la tabla *INPUT* para aceptar paquetes con puerto destino

tcp 22 (*/sbin/iptables -A INPUT -s %IP% -p tcp –dport 22 -j ACCEPT*). La segunda secuencia, llamada [closeSSH] es la secuencia inversa de la anterior y consiste en eliminar la anterior regla.

Para generar desde el cliente la secuencia de paquetes, se puede utilizar el comando *knock* que viene con el propio paquete "knockd" o cualquier comando generador de paquetes, por ejemplo *telnet* o *netcat*

```
PCX$ knock 192.168.29.123 7000 8000 9000
```

o con *telnet*

```
PCX$ telnet 192.168.29.123 7000
PCX$ telnet 192.168.29.123 8000
PCX$ telnet 192.168.29.123 9000
```

Es posible que con *telnet* o *netcat* no funcione a la primera, por tener que repetir paquetes si no llega respuesta rápido, pero después de 3 o 4 intentos debería funcionar.

14 Sistema DNS

El sistema Domain Name Server DNS traduce las direcciones IP en nombres con caracteres alfanuméricos. Tiene una organización centralizada y jerárquica, y está fundamentado en la existencia de unos servidores centrales (dominios raíz) que derivan las consultas del protocolo DNS a los servidores locales específicos (dominios primarios), y estos a su vez servidores con dominios secundarios.

En Linux, los servidores DNS especificados a nivel de usuario se guardan en el fichero "/etc/resolv.conf".

Para hacer consultas tipo DNS se dispone de los programas *dig* y *nslookup*. El programa *dig* hace las consultas por defecto a los servidores dados de alta en "/etc/resolv.conf", aunque puede especificarse cualquier servidor de nombres. Si se ejecuta sin argumentos, devuelve los servidores raíz.

```
PCX$ dig

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39270
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       21502   IN      NS      g.root-servers.net.
.                       21502   IN      NS      k.root-servers.net.
.                       21502   IN      NS      h.root-servers.net.
.                       21502   IN      NS      j.root-servers.net.
.                       21502   IN      NS      a.root-servers.net.
.                       21502   IN      NS      f.root-servers.net.
.                       21502   IN      NS      b.root-servers.net.
.                       21502   IN      NS      m.root-servers.net.
.                       21502   IN      NS      c.root-servers.net.
.                       21502   IN      NS      e.root-servers.net.
.                       21502   IN      NS      d.root-servers.net.
.                       21502   IN      NS      l.root-servers.net.
.                       21502   IN      NS      i.root-servers.net.

;; Query time: 54 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Mar 25 14:31:56 2013
;; MSG SIZE  rcvd: 228
```

La consulta mas sencilla tiene el formato:

```
PCX$ dig @193.147.87.2  aisa.ei.uvigo.es
```

donde "@193.147.87.2" es un servidor DNS cualquiera y "aisa.ei.uvigo.es" es un dominio del cual quiero conocer dónde está alojado.

Se propone probar con diferentes opciones del comando "dig".

El programa *nslookup* es mas sencillo y se utiliza para consultas básicas.

15 IPv6

En la mayoría de los hosts y equipos de red está instalada la pila de protocolos IPv4 y la pila de protocolos IPv6, como una estrategia de migración paulatina. Algunas aplicaciones incluso usan IPv6 como protocolo por defecto, y si la dirección destino es IPv4, cambian.

Se recomienda una familiarización con las herramientas disponibles y con el protocolo . Existe una página web dedicada a Linux/Debian en *http://wiki.debian.org/DebianIPv6* Algunos ejercicios:

15.1 Desactivar IPv6

Inicialmente se comprueba que IPv6 está activo en las interfaces.

```
PCX# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b8:27:ea:f4:03:a2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.29.120/24 brd 192.168.29.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::ba27:eaff:fef4:3a2/64 scope link
       valid_lft forever preferred_lft forever
```

```
PCX# sudo sysctl -a |grep disable_ipv6
net.ipv6.conf.all.disable_ipv6 = 0
net.ipv6.conf.default.disable_ipv6 = 0
net.ipv6.conf.eth0.disable_ipv6 = 0
net.ipv6.conf.lo.disable_ipv6 = 0
```

```
PCX# sudo sysctl -w net.ipv6.conf.eth0.disable_ipv6=1
net.ipv6.conf.eth0.disable_ipv6 = 1
```

Comprobación de que IPv6 está desactivado en eth0

```
PCX# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b8:27:ea:f4:03:a2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.29.120/24 brd 192.168.29.255 scope global eth0
       valid_lft forever preferred_lft forever
```

15.1.1 Versiones aneriores de kernel

Está referido a la desactivación de IPv6 en el Kernel. La desactivación de IPv6 en algunas aplicaciones es específico de ellas.

```
PCX# echo "net.ipv6.conf.all.disable_ipv6=1" > /etc/sysctl.d/ipv6disable.conf
PCX# echo "net.ipv6.conf.default.disable_ipv6=1" >> /etc/sysctl.d/ipv6disable.conf
```

Reiniciar el equipo. Se podría probar a desactivarlo en tiempo de ejecución, haciendo:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/disable_ipv6
# cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

En teoría con esto debería bastar. No obstante podría ser necesario ejecutar también:

```
# echo "net.ipv6.conf.all.disable_ipv6=1" > /etc/sysctl.d/ipv6disable.conf
# echo "net.ipv6.conf.default.disable_ipv6=1" >> /etc/sysctl.d/ipv6disable.conf
```

```
# echo "net.ipv6.conf.lo.disable_ipv6=1" >> /etc/sysctl.d/ipv6disable.conf
```

15.2 Utilizar un tunel IPv6

Existe un ISP que provee gratuitamente de servicios de interconexión a través de IPv6 en Internet. Se plantea como trabajo futuro y voluntario. La dirección web es *http://tunnelbroker.net/*

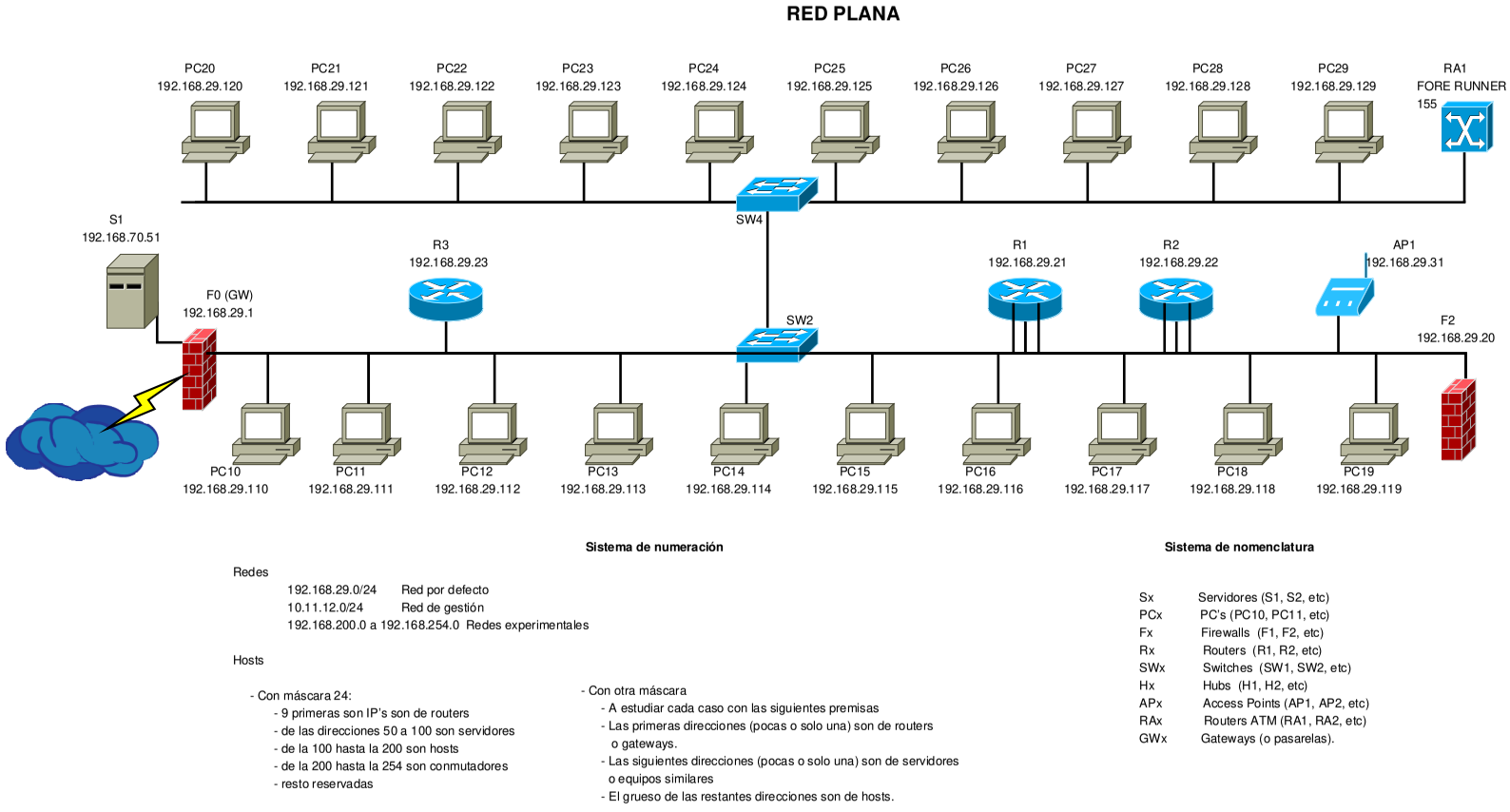

Figure 4: Red plana del laboratorio