Prácticas de Redes

Juan Carlos Fabero Jiménez
Rubén Santiago Montero
Rafael Moreno Vozmediano

Índice general

Práctica 1. Introducción al Laboratorio. LANs de cable (Ethernet)

1.1 Descripción del entorno

Las prácticas se realizarán sobre una máquina virtual con distribución Debian GNU/Linx. Dentro de esta máquina virtual se encuentran las herramientas necesarias para la realización de cada una de las prácticas, en particular las máquinas virtuales user-mode-linux (uml) que actuarán como hosts o encaminadores, así como los scripts necesarios para conectar estas máquinas entre sí, visualización de tráfico, etc.

1.1.1 Gestión del laboratorio virtual

En el laboratorio debemos iniciar sesión en GNU/Linux y con el usuario usuario_vms, que es el que tiene acceso a las máquinas virtuales de las distintas asignaturas.

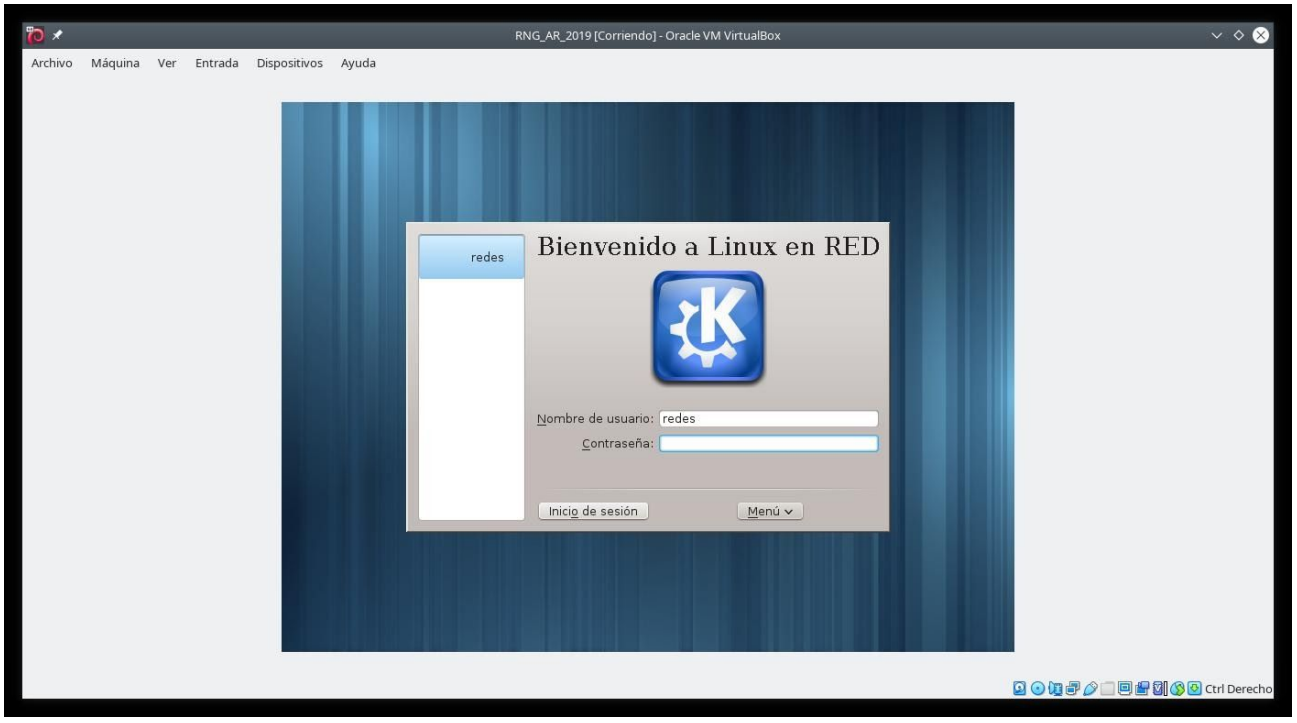
Una vez iniciada la sesión, desde la consola, deberemos usar el script redeslab para configurar la máquina virtual principal:

$ redeslab del
$ redeslab regenerate
$ redeslab start

La primera orden, redeslab del, sirve para borrar cualquier configuración que hubiera quedado de una práctica anterior en el mismo puesto de laboratorio.

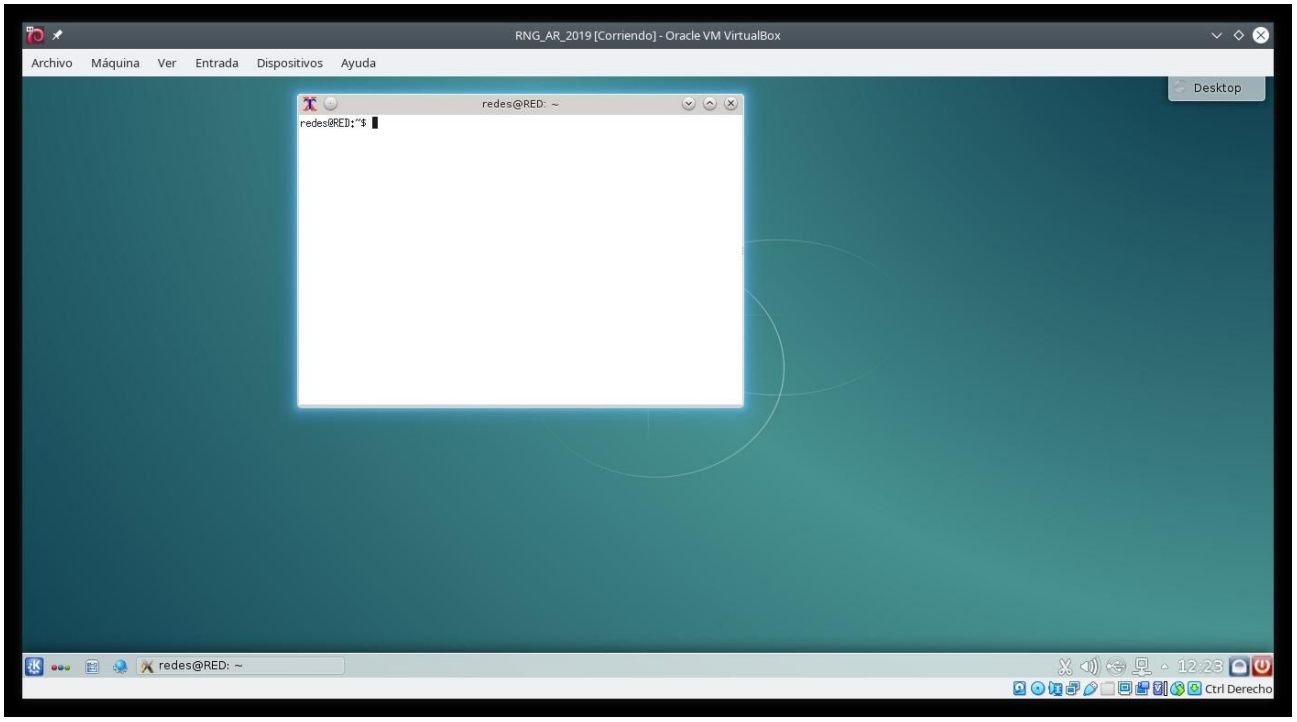
Con redeslab regenerate inicializamos el entorno de VirtualBox con la definición de la máquina principal.

Por último, redeslab start nos permite inicia dicha máquina principal, que será con la que trabajemos a partir de este momento. Nos debería aparecer la pantalla de bienvenida de la máquina principal:



El nombre de usuario es `redes` y la contraseña es `cursoredes`.

Cuando iniciamos sesión, se nos abre el entorno gráfico de la máquina principal con una consola. En esta consola será donde teclearemos las órdenes para configurar la topología virtual y lanzar las máquinas `uml`:



1.1.2 Definición de la topología virtual

Las máquinas `uml` tienen varios interfaces de red definidos, desde `eth0` hasta `eth9`, además del `loopback`. Para que se puedan comunicar entre ellas, es necesario crear una topología virtual que conecte las distintas interfaces. Para ello, crearemos, antes de cada práctica, un archivo de topología al que llamaremos `net.conf`. Este archivo consta de una o varias líneas de definición de *switches* virtuales. En esta asignatura usaremos un solo *switch* para conectar todas nuestras máquinas.

La sintaxis de este archivo es la siguiente:

```
defsw <nombre> <interfaz>*
```

donde **`defsw`** es una palabra clave (***def**ine **sw**itch*), <`nombre`> es el nombre de ese *switch* (usaremos siempre el nombre `vswitch`), e <`interfaz`>* es una lista con las interfaces de las máquinas, siguiendo la notación: `uml<número de máquina>.<número de interfaz>`. Por ejemplo:

```
defsw vswitch uml1.0 uml1.1 uml2.0 uml2.1 uml3.0 uml3.1 uml4.0
```

Con la línea anterior hemos definido un *switch* virtual llamado `vswitch` y al que se conectan las interfaces `eth0` de la máquina `uml1` (`uml1.0`), `eth1` de la máquina uml1 (`uml1.1`), etc., hasta la interfaz `eth0` de la máquina `uml4` (`uml4.0`)

Este archivo lo podemos crear con cualquier editor de texto, como `vim` o `nano`:

```
$ nano net.conf
```

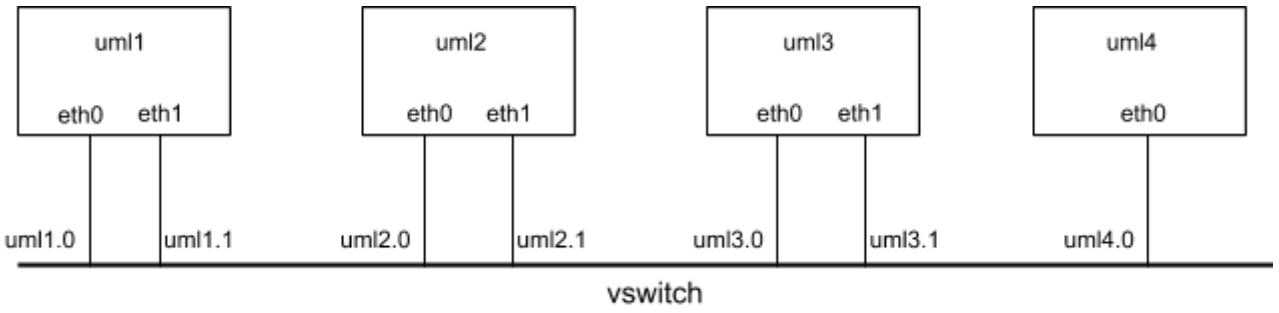
Una vez creado el archivo `net.conf` con el contenido anterior, debemos hacer efectiva esa configuración topológica con los *scripts* `ifovsdel` e `ifovsparse`:

```
$ sudo ifovsdel
$ sudo ifovsparse net.conf
```

Ambas órdenes deben ejecutarse con permiso de superusuario, por lo que utilizamos el comando `sudo` y la contraseña del usuario `redes`.

La orden `ifovsdel` (***inter**face **open**virtual**s**witch **del**ete*) elimina cualquier resto de topología de alguna práctica anterior. Debemos emplearla al comienzo de la práctica, y también al final antes de apagar la máquina principal para no dejar definiciones de topologías antiguas.

Con la orden `ifovsparse` (***inter**face **open**virtual**s**witch **parse***) examinamos el fichero `net.conf` que contiene la definición de la topología que queremos usar a partir de este momento. En este ejemplo, la topología que hemos creado es ésta:



En esta asignatura, puesto que la topología va a ser siempre la misma, podemos simplificar el proceso ejecutando simplemente el comando siguiente (hay que escribirlo tal y como aparece, prestando especial atención al uso de las llaves y los puntos):

```
$ echo defsw vswitch uml{1..5}.{0..4} > net.conf
```

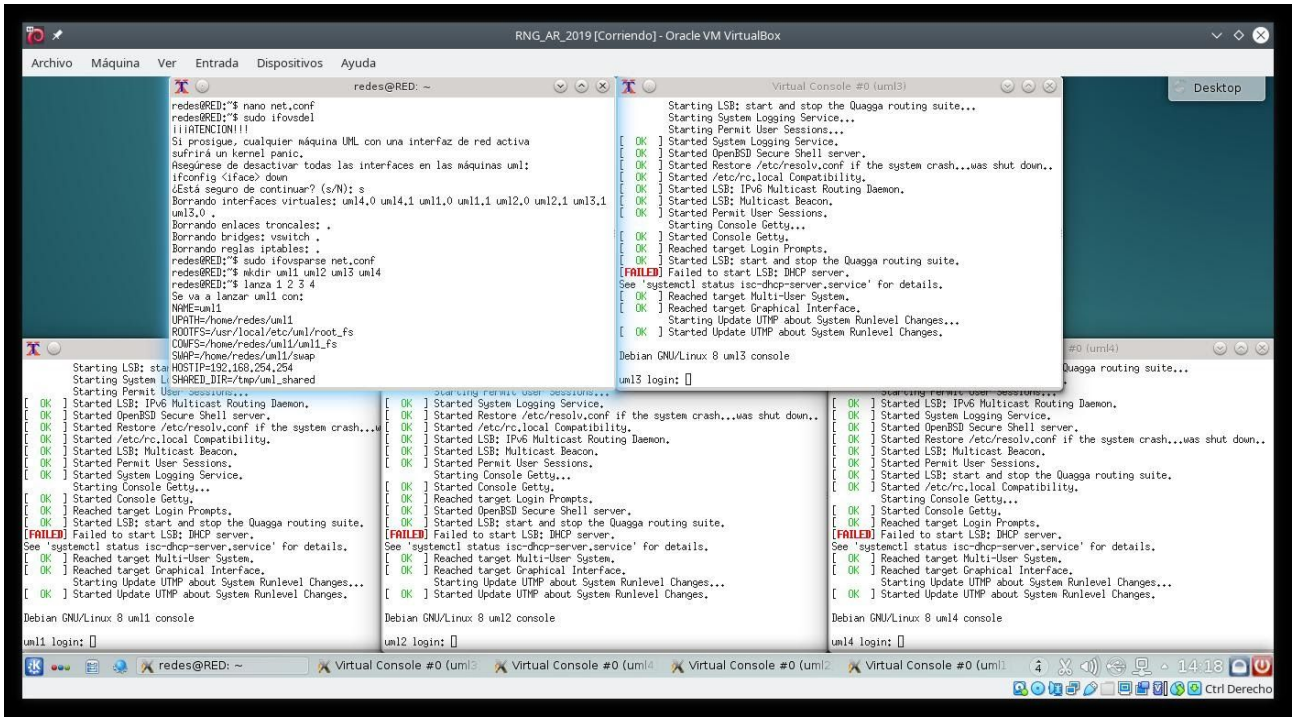
Una vez que hemos creado la topología virtual, deberemos crear un directorio para cada una de las máquinas virtuales, con el mismo nombre que éstas. Por ejemplo, para crear 4 máquinas virtuales, deberemos hacer:

```
$ mkdir uml1 uml2 uml3 uml4
```

Ahora ya podemos iniciar las máquinas con la orden `lanza` seguida de los números de las máquinas:

```
$ lanza 1 2 3 4
```

Aparecerán cuatro consolas, una para cada máquina `uml`:



Ahora ya podemos abrir sesión en cada una de las máquinas `uml`, con usuario `root` y sin necesidad de contraseña.

Cuando acabemos la práctica, deberemos apagar las máquinas `uml` con la orden `shutdown` en cada una:

```
root@uml1:~# shutdown -h now
```

y, por último, apagar la máquina virtual principal haciendo uso del botón rojo de apagado situado en la esquina inferior derecha del escritorio.

1.1.3 Resumen

Estos son los pasos que deberemos realizar al comienzo de cada práctica:

En la **máquina física**:

```
$ redeslab del
$ redeslab regenerate
$ redeslab start
```

En la **máquina virtual principal**:

```
$ echo defsw vswitch uml{1..5}.{0..4} > net.conf
$ sudo ifovsdel
$ sudo ifovsparse net.conf
$ mkdir uml1 uml2 uml3 uml4 uml5
```

(según el número de máquinas necesarias para la prácticas)

```
$ lanza 1 2 3 4 5
```

(según en número de máquinas necesarias para la práctica)

1.2 Configuración de Ethernet

1.2.1 Configuración básica de una interfaz de red Ethernet

Para la configuración de las interfaces Ethernet usaremos la orden genérica de configuración `ifconfig`, que nos permitirá la configuración de algunos detalles relacionados con IP que usaremos de modo básico en esta práctica.

Podemos consultar las principales opciones de la orden a través del manual:

```
$ man ifconfig
```

O bien, con la opción "`-h`" (o su forma larga "`--help`"):

```
# ifconfig --help
```

Si ejecutamos ifconfig con la opción "–a" se muestran los datos de configuración básica de todas las interfaces de red en el equipo:

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:00:00:00:01:f0
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

```
eth1      Link encap:Ethernet  HWaddr 02:00:00:00:01:f1
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

eth2      Link encap:Ethernet  HWaddr 02:00:00:00:01:f2
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

...

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:138 errors:0 dropped:0 overruns:0 frame:0
          TX packets:138 errors:0 dropped:0 overruns:0
          carrier:0 collisions:0 txqueuelen:0
          RX bytes:8696 (8.4 KiB)  TX bytes:8696 (8.4 KiB)
```

En este caso pueden verse, además del bucle local (`lo`), las interfaces Ethernet disponibles en cada puesto: `eth0` hasta `eth9` con los datos correspondientes (muchos de ellos están todavía sin configurar y se irán explicando más adelante).

Para cada interfaz se especifican el protocolo de encapsulamiento (Ethernet II), la dirección HW, la dirección IP, la dirección de broadcast y la máscara de red que le corresponde por defecto. Se incluyen además estadísticas sobre el número de paquetes recibidos, enviados, erróneos, descartados...

En nuestro caso, como las interfaces de red son virtuales, no tienen asignada una dirección MAC global, sino que su gestión se realiza de modo local y la define el administrador de la red. Por eso todas ellas empiezan por 02:xx:xx:xx:xx:xx. El convenio que se ha seguido es el siguiente: el primer byte de la dirección MAC vale 0x02 (gestión local, unicast). El penúltimo byte corresponde con el número de máquina virtual (0x01 para la máquina uml1, 0x02 para la máquina uml2, y así sucesivamente). El último byte identifica la interfaz dentro de la máquina (0xf0 para eth0, 0xf1 para eth1...). De esta manera es fácil identificar el origen y el destino de una trama cuando analizamos el tráfico mediante wireshark. Por ejemplo, la dirección MAC 02:00:00:00:01:f3 pertenece a la interfaz eth3 de la máquina virtual uml1.

1.2.2 Desactivación/activación de una interfaz de red

En algunas ocasiones, después de modificar la configuración de una interfaz de red, para que se haga efectiva hay que activar dicha modificación. La activación o desactivación de un interfaz se hace con la orden `ifconfig`, por ejemplo:

```
# ifconfig eth0 down
```

```
# ifconfig eth0 up
```

También puede hacerse con:

```
# ifdown eth0
# ifup eth0
```

La diferencia entre las dos versiones de cada orden es que ifdown e ifup leen el archivo de configuración /etc/network/interfaces, cuyo uso se explicará en una práctica posterior.

1.2.3 Asignación de direcciones IP

Para poder usar aplicaciones IP, necesitamos asignar a cada interfaz una dirección IP. A lo largo de estas prácticas usaremos direcciones privadas de clases B y C.

Podemos configurar las interfaces Ethernet mediante ifconfig. Lo haremos configurando en el laboratorio dos redes IP diferentes, la 192.168.0.0 y la 192.168.1.0. Para ello configuramos dos de las interfaces de cada máquina uml como se indica a continuación.

Primero configuramos la interfaz eth0 con una dirección de la red 192.168.0.0 (usaremos como campo host_id el número de máquina):

```
# ifconfig eth0 192.168.0.<máquina>/24 up
```

Luego configuramos la interfaz eth1 con una dirección de la red 192.168.1.0:

```
# ifconfig eth1 192.168.1.<máquina>/24 up
```

Comprobar que se ha realizado la configuración de ambos interfaces correctamente haciendo uso de la orden ifconfig para visualizar la configuración de red de cada interfaz. Probar con las siguientes órdenes:

```
# ifconfig -a
# ifconfig lo
# ifconfig eth0
# ifconfig eth1
```

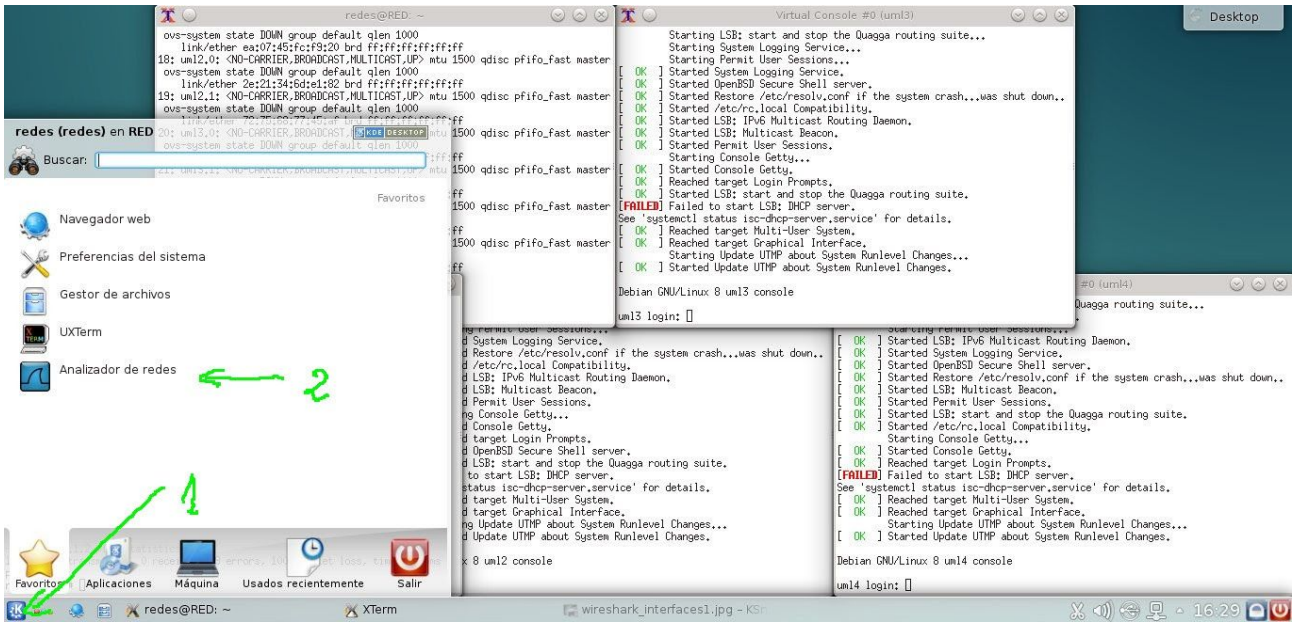
1.3 Visualización del tráfico con Wireshark

Para poner de relieve el funcionamiento de la red, necesitamos:

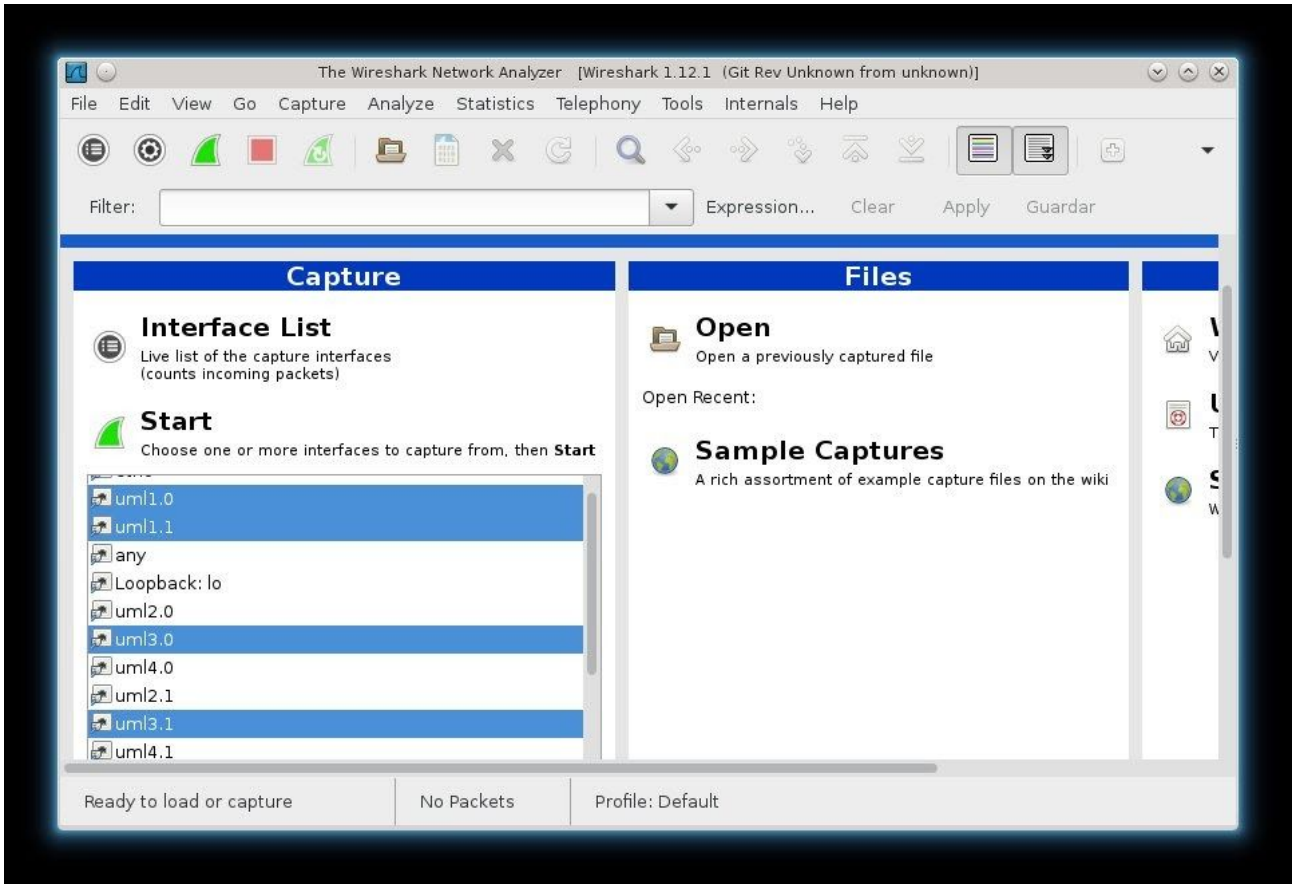
1. Poder generar tráfico a medida, y
2. poder capturarlo y analizarlo.

Para la segunda de estas funciones usamos la herramienta libre **`Wireshark`**. Es un analizador de tráfico que permite capturar las tramas que pasan por una interfaz, y analizar su contenido asociando en cada capa de la arquitectura de red el protocolo al que corresponde, e identificando los campos de cada paquete o trama. La emplearemos a lo largo de todas las prácticas, por lo que conviene familiarizarse con su uso.

Como es una herramienta gráfica, sólo se encuentra instalada en la máquina virtual principal. Para iniciarla, siempre dentro de la máquina principal, pulsar en el icono de escritorio correspondiente:



Aparecerá una ventana similar a la de la siguiente figura:

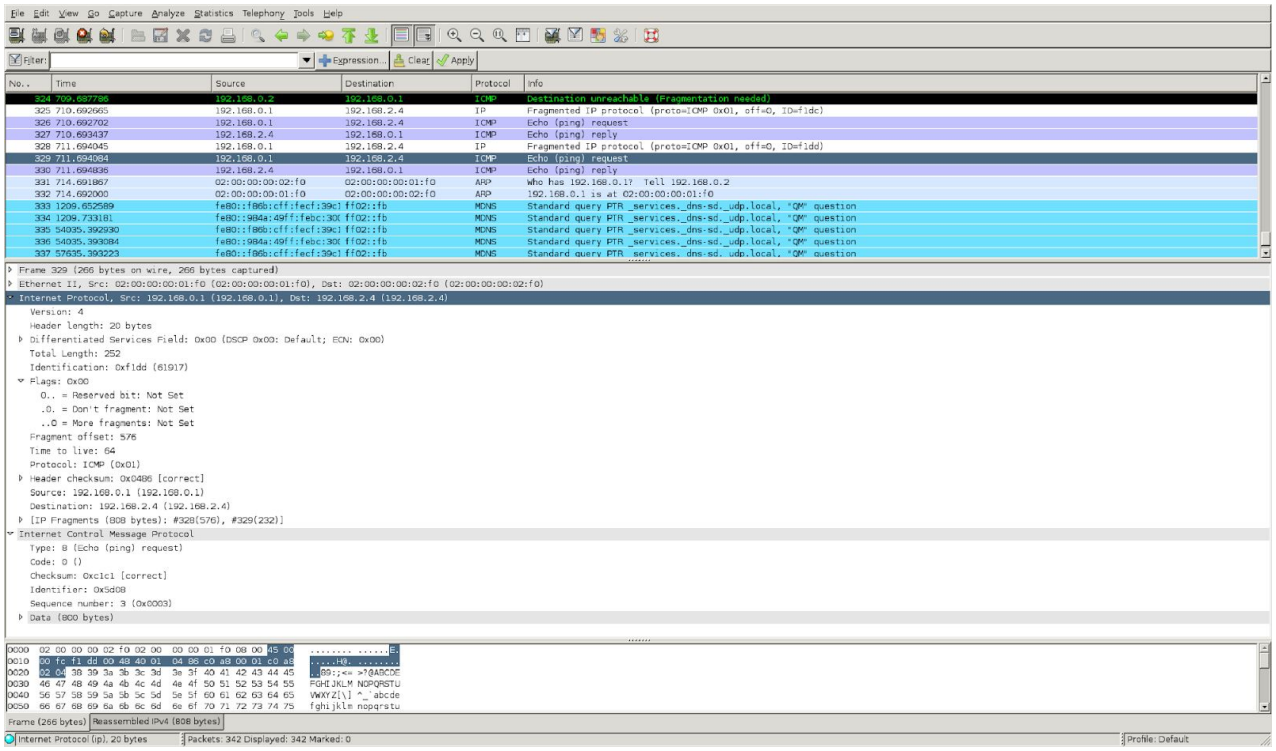


Para visualizar el tráfico, hay que especificar la interfaz o interfaces por la que va a escucharse (por ejemplo, uml1.0 y uml1.1). Para seleccionar varias interfaces, seleccionamos la primera y luego las demás manteniendo pulsada la tecla Ctrl, como puede verse en la figura.

Cuando hemos seleccionado las interfaces sobre las que realizaremos la captura, pulsamos en `Start` para empezar la misma.

Puede configurarse para capturar un número determinado de paquetes, o durante un intervalo de tiempo, o de forma indefinida y detenerse manualmente. Es interesante configurarlo de modo que la ventana principal se actualice en tiempo real.

Al iniciar la captura de paquetes, el aspecto de wireshark será el que aparece en la figura:



Además de la barra de menús desplegables y de botones de control para las acciones más básicas (selección de la interfaz, iniciar o detener captura, etc.), la ventana principal de `wireshark` muestra tres zonas:

- La principal refleja la secuencia de tramas capturadas a través de la interfaz que se haya seleccionado. Puede visualizarse un subconjunto de dichas tramas eligiendo como **Filtro** uno o varios criterios de selección que pueden incluir diversos protocolos.
- Debajo se muestra el contenido de la trama seleccionada, con la jerarquía de protocolos anidados y el correspondiente encapsulado (que puede desplegarse o no). Para cualquier nivel pueden analizarse los campos de la trama o del paquete.
- Finalmente, en una tercera se muestra el contenido de la trama capturada en formato hexadecimal.

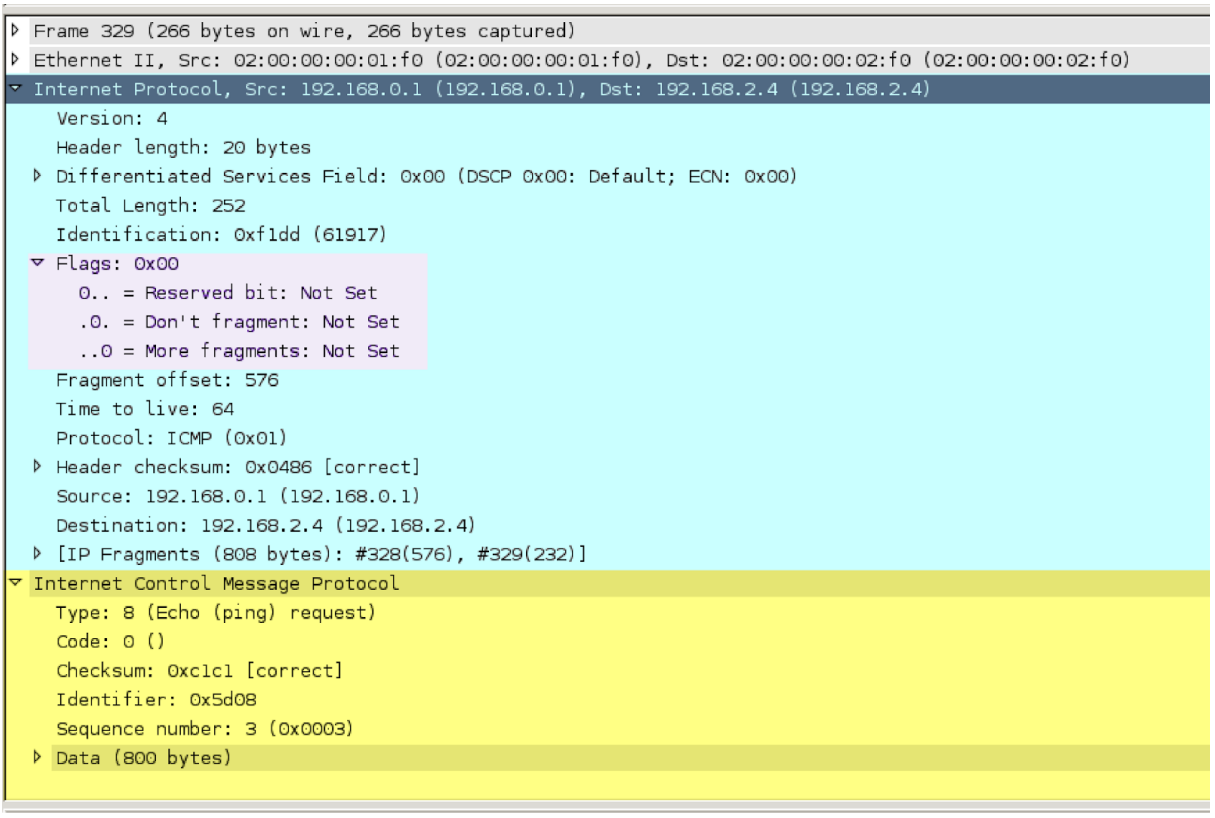
Para generar el tráfico usaremos la orden `ping` dentro de las máquinas uml. Esta orden permite generar un mensaje `ICMP` del tipo `Echo request` (`Type=8`) y enviarlo a una dirección IP de destino. Este, a su vez, produce una respuesta `ICMP` del tipo `Echo reply` (`Type=0`). El mensaje `ICMP` se encapsula en un datagrama IP, y éste a su vez en una trama Ethernet.

A continuación analizaremos mediante `wireshark` el tráfico que se intercambian las máquinas virtuales. Para generar tráfico usar la siguiente orden en alguna de las máquinas `uml` (iniciar la captura **ANTES** de usar la orden):

```
ping  <dir_destino>
```

Poner primero como destino la dirección de la red `192.168.0.0` de la otra `uml`, y luego otra de la red `192.168.1.0`. En cada caso, el datagrama IP se transmitirá a través de la interfaz apropiada, hecho que podemos comprobar examinando las direcciones MAC de las tramas. Capturar algunos

paquetes con `wireshark`. En la ventana superior de la aplicación se ven todos los paquetes capturados, por orden. El paquete seleccionado y su contenido pueden analizarse con detalle en el área situada debajo. En la siguiente figura, vemos un ejemplo de la captura y análisis de un mensaje ICMP de tipo `Echo request`:



En color azul se han resaltado los campos pertenecientes a IP. Podemos ver, en primer lugar, el campo de versión, 4 en este caso. A continuación aparece la longitud de la cabecera, 20 bytes, y los campos TOS, longitud total e identificación. Se ha desplegado el campo de indicadores (*flags*) para mostrar el estado de cada uno de los tres bits: el primero está reservado, por lo que debe valer 0; el bit DF (no fragmentar) está desactivado, por lo que se permite la fragmentación del datagrama en los encaminadores intermedios; y el bit MF (más fragmentos) también vale 0, lo que indica que éste es el único o último fragmento de un datagrama. Puesto que el campo de desplazamiento (*offset*) que aparece a continuación vale 576, sabemos que en este caso sí hubo fragmentación, y además éste es el último fragmento del datagrama original. Por último, en color amarillo se ha resaltado la información correspondiente al mensaje ICMP que viajaba dentro del datagrama: tipo 8 (solicitud de eco), código 0 (no hay subtipos), suma de comprobación, identificador del mensaje ICMP, número de secuencia y datos.

Para generar carga de tráfico elevada podemos usar la siguiente orden:

```
# ping -f -b <dir_broadcast>
```

Veremos que ninguna máquina responde al `ping`. Esto es porque, por defecto, están configuradas para no responder a este tipo de mensajes dirigidos a la dirección de difusión con el fin de evitar un tipo de ataque de denegación de servicio. Para activar las respuestas debe usarse la orden sysctl, que se estudiará con más detalle en prácticas sucesivas. Repetir el ping a la dirección de difusión de la red tras ejecutar la siguiente orden en cada máquina uml:

```
# sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0
```

Estudiar el encapsulado de los paquetes capturados mediante wireshark, analizando las tramas Ethernet, los datagramas IP y los mensajes ICMP, comprobando los distintos campos representados en las figuras, en cada caso.

1.4 Configuración de la capa MAC

Las órdenes `ifconfig` e `ip` nos permiten, también, modificar ciertos parámetros pertenecientes a la subcapa MAC de las interfaces.

1.4.1 Modificar la dirección MAC de la interfaz de red

Para modificar la dirección MAC podemos utilizar la orden ifconfig:

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 02:11:22:33:44:55
# ifconfig eth0 up
```

También se puede modificar la dirección MAC mediante la orden ip link:

```
# ip link set dev eth0 down
# ip link set dev eth0 address 02:11:22:33:44:55
# ip link set dev eth0 up
```

En ambos casos hemos fijado a 1 el valor del segundo bit menos significativo del byte más significativo de la dirección MAC, para indicar que es una dirección administrada localmente.

Ejercicios

1. En la máquina virtual `uml1`, obtener la dirección MAC de la interfaz `eth0` mediante la orden `ifconfig` y anotarla.
2. Modificar la dirección MAC de la interfaz `eth0` mediante la orden `ifconfig`, poniendo el siguiente valor `02:01:02:03:04:05`.
3. Comprobar la configuración mediante las órdenes `ifconfig` e `ip link`.
4. Restablecer la dirección MAC original de la interfaz `eth0` mediante la orden `ip link`.
5. Comprobar la configuración mediante las órdenes `ifconfig` e `ip link`.

1.4.2 Modificar la MTU de la interfaz de red

Para Ethernet, la MTU se fija a un valor máximo de 1500 bytes. Sin embargo, podemos modificarlo y fijarlo a valores menores mediante las órdenes `ifconfig` e `ip`.

Para modificar la MTU mediante la orden ifconfig:

```
# ifconfig eth0 mtu 1000
```

Para modificar la MTU mediante la orden ip link:

```
# ip link set dev eth0 mtu 1000
```

Ejercicios

1. Modificar la MTU de la interfaz `eth0` mediante la orden `ip`, poniendo el valor 1200.
2. Comprobar la configuración mediante las órdenes `ifconfig` e `ip link`.
3. Restablecer la MTU original (1500) de la interfaz `eth0` mediante la orden `ifconfig`.
4. Comprobar la configuración mediante las órdenes `ifconfig` e `ip link`.

1.4.3 Modificar la dirección MAC de difusión (*broadcast*) de la interfaz de red

Aunque el efecto en un medio de difusión como Ethernet será limitado, podemos modificar la dirección MAC de difusión mediante la orden ip link:

```
# ip link set dev eth0 broadcast f1:f2:f3:f4:f5:f6
```

Ejercicios

1. Modificar la dirección MAC de difusión de la interfaz `eth0` mediante la orden `ip link`, asignando el siguiente valor: `f1:f2:f3:f4:f5:f6`.
2. Comprobar la configuración mediante la orden `ip link`.
3. Ejecutando `wireshark` en la máquina *frontend*, realizar un `ping` desde la máquina `uml1` a la máquina `uml2`.
4. Comprobar que las preguntas `arp` van dirigidas a la nueva dirección de difusión.
5. Restaurar la dirección MAC de difusión original (`ff:ff:ff:ff:ff:ff`).
6. Comprobar la configuración mediante la orden `ip link`.

1.4.4 Habilitar/deshabilitar el flag *multicast* de la interfaz de red

Podemos habilitar o deshabilitar el indicador multicast mediante la orden ifconfig:

```
# ifconfig eth0 multicast
# ifconfig eth0 -multicast
```

También se puede conseguir mediante la orden ip link:

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

Además, con ip maddr podemos ver a qué grupos de multidifusión está asociada la interfaz:

```
# ip maddr show dev eth0
2:      eth0
        link  33:33:00:00:00:fb
        link  33:33:ff:00:50:1a
        link  01:00:5e:00:00:fb
        link  33:33:ff:8a:b9:c5
        link  01:00:5e:00:00:01
        link  33:33:00:00:00:01
```

Fijémonos en que todas las direcciones comienzan por un byte impar, esto es, tienen activado el bit menos significativo.

Ejercicios

1. Deshabilitar el *multicast* de la interfaz `eth0` mediante la orden `ifconfig`.
2. Comprobar la configuración mediante las órdenes `ifconfig` e `ip link`.
3. Habilitar el *multicast* de la interfaz `eth0` mediante la orden `ip link`.
4. Comprobar la configuración mediante las órdenes `ifconfig` e `ip link`.
5. Listar los grupos asociados mediante la orden `ip maddr`.

Práctica 2. Configuración básica de IP

2.1 Configuración básica de la interfaz de red

En esta práctica aprenderemos a configurar los parámetros básicos de la red IP. La configuración de la interfaz de red se puede consultar o modificar mediante las siguientes órdenes:

- Orden `ifconfig`
- Orden `ip link`

Algunas de las operaciones básicas que se pueden realizar son las siguientes:

- Visualizar la configuración de la interfaz de red.
- Habilitar/deshabilitar la interfaz de red.
- Modificar la dirección MAC de la interfaz de red.
- Modificar la MTU de la interfaz de red.
- Modificar la dirección MAC de broadcast de la interfaz de red.
- Habilitar/deshabilitar algunos indicadores de configuración (p. ej. multicast).

2.1.1 Visualizar la configuración de la interfaz de red

Podemos visualizar la configuración de todas las interfaces de red mediante la orden `ifconfig`:

```
# ifconfig -a
```

También se puede visualizar la configuración de la interfaz `eth1` mediante la orden `ifconfig`:

```
# ifconfig eth1
```

Para visualizar la configuración de todas las interfaces de red mediante la orden `ip link`:

```
# ip link show
```

Para visualizar la configuración de la interfaz `eth1` mediante la orden `ip link`:

```
# ip link show dev eth1
```

Añadiendo la opción `-s` a la orden `ip` se muestran las estadísticas de funcionamiento:

```
# ip -s link show dev eth1
```

Se pueden emplear abreviaturas para la orden `ip link`:

```
show = sh, list, lst, ls, l
```

Ejercicios

1. Probar las órdenes anteriores.

2.1.2 Habilitar/deshabilitar la interfaz de red

Podemos habilitar o deshabilitar la interfaz de red mediante la orden `ifconfig`:

```
# ifconfig eth1 up
# ifconfig eth1 down
```

También puede hacerse mediante la orden `ip link`:

```
# ip link set dev eth1 up
# ip link set dev eth1 down
```

Ejercicios

1. Deshabilitar la interfaz `eth1` mediante la orden `ifconfig`.
2. Comprobar su configuración (indicador `UP`) mediante las órdenes `ifconfig` e `ip link`.
3. Habilitar de nuevo la interfaz `eth1` mediante la orden `ip link`.
4. Comprobar su configuración (indicador `UP`) mediante las órdenes `ifconfig` e `ip link`.

2.2 Configuración de los parámetros IP

Para esta parte de la práctica emplearemos dos máquinas virtuales, `uml1` y `uml2`.

La configuración de los parámetros básicos de IP (dirección IP, máscara de red y dirección de difusión) se puede ver o modificar mediante las siguientes órdenes:

- Orden `ifconfig`
- Orden `ip address`

2.2.1 Configuración básica de los parámetros IP con la orden ifconfig

Mediante la orden `ifconfig` se pueden configurar la dirección IP, la máscara de red y la dirección de difusión:

```
# ifconfig eth1 10.1.1.1
# ifconfig eth1 netmask 255.255.255.0
# ifconfig eth1 broadcast 10.1.1.255
```

También se pueden configurar los tres parámetros en la misma orden:

```
# ifconfig eth1 10.1.1.1 netmask 255.255.255.0 broadcast
10.1.1.255
```

Como alternativa, se puede usar el formato **CIDR** para especificar la máscara:

```
# ifconfig eth1 10.1.1.1/24 broadcast 10.1.1.255
```

En este caso, no sería necesario especificar la dirección de difusión, pues la orden `ifconfig` ya la calcula a partir de la dirección IP y de la máscara de red, por lo que bastaría con escribir:

```
# ifconfig eth1 10.1.1.1/24
```

Una vez que se ha configurado una interfaz, se activa automáticamente. Sin embargo, esto no ocurre en todas las variantes de Unix, por lo que es conveniente acostumbrarse a escribir el parámetro `up` al final de la línea de `ifconfig`:

```
# ifconfig eth1 10.1.1.1/24 up
```

Una interfaz que tiene una dirección asignada puede desactivarse temporalmente mediante la orden:

```
# ifconfig eth1 down
```

Sin embargo, dicha interfaz no pierde su configuración, por lo que para volver a activarla se puede hacer, simplemente:

```
# ifconfig eth1 up
```

Para borrar la configuración de una interfaz debe asignársele la dirección `0.0.0.0`:

```
# ifconfig eth1 192.168.1.1
# ifconfig eth1
eth1   Link encap:Ethernet  HWaddr 02:00:00:00:01:01
       inet addr:192.168.1.1  Bcast:192.168.1.255
Mask:255.255.255.0
       inet6 addr: fe80::ff:fe00:101/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:66 errors:0 dropped:0 overruns:0 frame:0
       TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:5142 (5.0 KiB)  TX bytes:2016 (1.9 KiB)
       Interrupt:5

# ifconfig eth1 0.0.0.0
# ifconfig eth1
eth1   Link encap:Ethernet  HWaddr 02:00:00:00:01:01
       inet6 addr: fe80::ff:fe00:101/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:66 errors:0 dropped:0 overruns:0 frame:0
       TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:5142 (5.0 KiB)  TX bytes:2016 (1.9 KiB)
       Interrupt:5
```

Ejercicios

1. Asignar a la interfaz `eth1` de cada máquina la siguiente configuración IP usando la orden `ifconfig`:
   Dirección IP: `10.1.1.<máquina>`
   Máscara: `255.0.0.0`
   Difusión: `10.255.255.255`
   - Comprobar la configuración mediante la orden `ifconfig eth1`.
   - Comprobar mediante la orden `ping` desde una de las máquinas que es posible la comunicación con la otra máquina.
2. Asignar a la interfaz `eth1` la siguiente configuración IP usando la orden `ifconfig`:
   Dirección IP/Máscara: `10.1.1.<máquina>/16`
   Difusión: `10.1.255.255`
   - Comprobar la configuración mediante la orden `ifconfig eth1`.

- Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina.

3. Asignar a la interfaz `eth1` la siguiente configuración IP usando la orden `ifconfig`:
   Dirección IP/Máscara: `10.1.1.<máquina>/24`
   Difusión: `10.1.1.255`
   - Comprobar la configuración mediante la orden `ifconfig eth1`.
   - Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina.

2.2.2 Configuración de los parámetros IP con la orden `ip address`

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 configuración anterior, la interfaz mantiene ambas direcciones simultáneamente.

Esta situación es completamente normal, ya que, como se verá en el apartado siguiente, es posible configurar una misma interfaz de red con varias direcciones IP distintas, con sus correspondientes máscaras y direcciones de difusión.

Para visualizar los parámetros de configuración de IP mediante la orden `ip`:

```
# ip address show dev eth1
```

Para eliminar la dirección IP asociada a la interfaz de red mediante la orden `ip`, se emplea el parámetro `delete`:

```
# ip address delete 192.168.1.1/24 dev eth1
```

Se puede añadir una nueva dirección IP, máscara y dirección de difusión a la interfaz de red mediante la orden `ip address` y el parámetro `add`:

```
# ip address add 10.1.1.1/24 broadcast 10.1.1.255 dev eth1
```

Al contrario que la orden `ifconfig`, `ip` no comprueba la clase de la dirección para asignar de manera automática una máscara de red y una dirección de difusión en caso de que no las especifiquemos en la línea de órdenes. Por ejemplo, la orden `ip` asignará como máscara de red la `255.255.255.255` y como dirección de difusión la `0.0.0.0`:

```
# ip address add 10.1.1.1 dev eth1
```

Así, será imposible comunicarse con ninguna otra máquina conectada a ese enlace, pues la red que acabamos de crear consta de una sola dirección IP. Por ello es importante no olvidar nunca especificar la longitud de la máscara de red en notación CIDR. Podemos forzar a `ip` a calcular la dirección de difusión asociada especificando la etiqueta especial '`+`'. Por ejemplo:

```
# ip address add 10.1.1.1/24 broadcast + dev eth1
# ifconfig eth1
eth1    Link encap:Ethernet  HWaddr 02:00:00:00:01:f1
   inet addr:10.1.1.1  Bcast:10.1.1.255  Mask:255.255.255.0
        BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:5
```

La orden `ip` address admite el uso de las siguientes abreviaturas:

```
show    = sh, list, lst, ls, l
address = addr, a
add     = a
delete  = del, d
broadcast = brd, br
```

Por ejemplo:

```
# ip addr add 192.168.1.1/20 br + dev eth1
```

Es importante tener en cuenta que, a diferencia de la orden `ifconfig` sobre GNU/Linux, la orden `ip` no activa de manera automática la interfaz, por lo que hay que activarla explícitamente:

```
# ip link set dev eth1 up
```

Ejercicios

1. Comprobar la configuración IP de la interfaz `eth1` mediante la orden `ip address show dev eth1`.
2. Si la interfaz tiene una dirección IP asignada, eliminar dicha dirección/máscara mediante la orden `ip address`.
3. Asignar a la interfaz `eth1` la siguiente configuración IP usando la orden `ip address`:
   Dirección IP: `10.1.1.<máquina>/8`
   Difusión: `10.255.255.255`
   - Comprobar la configuración IP de la interfaz `eth1` mediante la orden `ip address show dev eth1`.
   - Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina.
4. Eliminar la anterior dirección IP/máscara mediante la orden `ip address` y asignar a la interfaz la siguiente configuración IP, usando la orden `ip address`:
   Dirección IP/Máscara: `10.1.1.<máquina>/16`
   Difusión: `10.1.255.255`
   - Comprobar la configuración mediante la orden `ip address show dev eth1`.
   - Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina.
5. Eliminar la anterior dirección IP/máscara mediante la orden `ip address` y asignar a la interfaz la siguiente configuración IP, usando la orden `ip address`:
   Dirección IP/Máscara: `10.1.1.<máquina>/24`
   Difusión: `10.1.1.255`
   - Comprobar la configuración mediante la orden `ifconfig eth1`.
   - Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina.

2.2.3 Asignación de varias direcciones IP a una misma interfaz de red

Como se mencionó anteriormente, es posible asignar varias direcciones IP distintas (con sus correspondientes máscaras y direcciones de difusión) a una misma interfaz de red. En esta situación, la interfaz de red es capaz de comunicarse a través de todas las direcciones IP que tenga asociadas. Esta operación se puede realizar usando la orden `ifconfig` o mediante la orden `ip address`.

La principal diferencia es que usando la orden `ifconfig` es obligatorio asociar una etiqueta o nombre alternativo a la nueva dirección de la interfaz de red. Por ejemplo, si asignamos dos nuevas direcciones IP a la interfaz de red `eth1`, las etiquetas asociadas a estas dos nuevas direcciones serían `eth1:1` y `eth1:2`. Estas etiquetas reciben también el nombre de alias.

En el caso de usar la orden `ip address`, la asignación de etiquetas a las distintas direcciones IP de una misma interfaz no es obligatorio, sino opcional.

Por ejemplo, para asignar una nueva dirección IP a la interfaz de red `eth1`, etiquetada con el nombre `eth1:1`, mediante la orden `ifconfig`:

```
# ifconfig eth1:1 192.168.1.1/24 broadcast 192.168.1.255
```

Para visualizar la configuración del alias etiquetado como `eth1:1` mediante la orden `ifconfig`:

```
# ifconfig eth1:1
```

Eliminar la dirección IP asociada al alias `eth1:1` mediante la orden `ifconfig`:

```
# ifconfig eth1:1 down
```

A diferencia de una interfaz física, cuando se desactiva (*down*) un alias, éste desaparece y ya no puede volver a activarse simplemente con

```
# ifconfig eth1:1 up
```

sino que hay que volver a crearlo asignándole de nuevo los parámetros de configuración.

Con la orden `ip address` también se pueden asignar varias direcciones IP a la misma interfaz simultáneamente. Por ejemplo, para añadir una nueva dirección IP a la interfaz de red `eth1` mediante la orden `ip address`, sin etiqueta:

```
# ip address add 192.168.1.1/24 broadcast 192.168.1.255 dev eth1
```

Para visualizar la configuración de todas las direcciones asociadas a la interfaz `eth1` mediante la orden `ip address`:

```
# ip address show dev eth1
```

Nota: Observar que esta nueva dirección IP, al no tener asociada una etiqueta, no se puede visualizar mediante la orden `ifconfig`.

Es posible eliminar una dirección IP asociada a la interfaz de red mediante la orden `ip address`:

```
# ip address delete 192.168.1.1/24 dev eth1
```

Añadir una nueva dirección IP a la interfaz de red `eth1` mediante la orden `ip address` con etiqueta `eth1:1` :

```
# ip address add 192.168.1.1/24 broadcast 192.168.1.255
label eth1:1 dev eth1
```

Visualizar la configuración de todas las direcciones asociadas a la interfaz `eth1` mediante la orden `ip address`:

```
# ip address show dev eth1
```

Nota: Observar que esta nueva dirección IP, al tener una etiqueta asociada, también se puede visualizar mediante la orden `ifconfig eth1:1`.

Para eliminar la última dirección IP asociada a la interfaz de red mediante la orden `ip address`:

```
# ip address del 192.168.1.1/24 dev eth1
```

Ejercicios

1. Usando la orden `ifconfig` en ambas máquinas virtuales, configurar tres direcciones IP asociadas a la interfaz `eth1`, con los siguientes valores:

`eth1`	IP/máscara: `192.168.1.<máquina>/24` Difusión: `192.168.1.255`
`eth1:1`	IP/máscara: `10.1.1.<máquina>/24` Difusión: `10.1.1.255`
`eth1:2`	IP/máscara: `172.16.1.<máquina>/24` Difusión: `172.16.1.255`

   - Comprobar la configuración de las tres direcciones mediante la orden `ifconfig -a`.
   - Comprobar la configuración de las tres direcciones mediante la orden `ip address show dev eth1`.
   - Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina a través de las tres direcciones.

2. Eliminar las direcciones asociadas a las interfaces `eth1:1` y `eth1:2` mediante la orden `ifconfig`.
3. Usando la orden `ip address`, añadir dos direcciones IP asociadas a la interfaz `eth1`, sin usar etiquetas, con los siguientes valores:

Dirección 1:	IP/máscara: `10.1.1.<máquina>/24`

	Difusión: `10.1.1.255`
Dirección 2:	IP/máscara: `172.16.1.<máquina>/24` Difusión: `172.16.1.255`

- Observar que las dos direcciones nuevas no se pueden visualizar mediante la orden `ifconfig -a`.
- Comprobar la configuración de las tres direcciones asociadas a la interfaz `eth1` mediante la orden `ip address show dev eth1`.
- Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina a través de las tres direcciones.

4. Eliminar las dos últimas direcciones asociadas a la interfaz `eth1` mediante la orden `ip address`.
5. Usando la orden `ip address add`, añadir de nuevo dos direcciones IP a la interfaz `eth1`, pero ahora usando etiquetas:

Dirección 1:	IP/máscara: `10.1.1.<máquina>/24` Difusión: `10.1.1.255` etiqueta: `eth1:1`
Dirección 2:	IP/máscara: `172.16.1.<máquina>/24` Difusión: `172.16.1.255` etiqueta: `eth1:2`

- Observar que ahora las dos direcciones nuevas sí se pueden visualizar mediante la orden `ifconfig -a`.
- Comprobar la configuración de las tres direcciones asociadas a la interfaz `eth1` mediante la orden `ip address show dev eth1`.
- Comprobar mediante la orden `ping` que es posible la comunicación con la otra máquina a través de las tres direcciones.

2.2.4 Archivo de configuración de los parámetros IP

Cuando la máquina se inicia, el sistema operativo en las distribuciones Linux basadas en Debian (Ubuntu, por ejemplo) lee los parámetros de configuración de la red del archivo de configuración `/etc/network/interfaces`.

Cualquier cambio que se realice en la configuración de la red mediante las órdenes `ifconfig` o `ip address` se perderá al apagar o reiniciar el sistema. Si queremos que la configuración sea permanente y se mantenga en el siguiente reinicio de la máquina, se debe establecer en el archivo `/etc/network/interfaces`.

A continuación se muestra un ejemplo del formato del archivo `/etc/network/interfaces`:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth1
```

```
iface eth1 inet static
      address 192.168.1.1
      netmask 255.255.255.0
      network 192.168.1.0
      broadcast 192.168.1.255
      gateway 192.168.1.100
```

Los campos que aparecen en este archivo son los siguientes:

- `auto lo`: Indica que la interfaz `lo` (*loopback*) se debe activar automáticamente durante el arranque del sistema.
- `iface lo inet loopback`: Indica que la interfaz `lo` se debe configurar en modo *loopback*.
- `auto eth1`: Indica que la interfaz `eth1` se debe activar automáticamente durante el arranque del sistema.
- `iface eth1 inet static`: Indica que la interfaz `eth1` se debe configurar en modo estático, es decir, que debe tomar los parámetros de configuración, referentes a IPv4, que se indican a continuación:
  - `address`: dirección IP de la interfaz.
  - `netmask`: máscara de red.
  - `network`: dirección de la red.
  - `broadcast`: dirección de difusión.
  - `gateway`: dirección del encaminador predeterminado (*default router*).

Ejercicios

1. En la máquina virtual `uml1`, escribir en el archivo `/etc/network/interfaces` el contenido mostrado anteriormente. Reiniciar la máquina mediante la orden `shutdown -r now` y comprobar, mediante la orden `ifconfig`, que se mantiene la configuración de la interfaz `eth1`.

2.3 El protocolo ARP

Linux considera internamente que las direcciones IP pertenecen al equipo, a pesar de que son asignadas a cada interfaz. Por tanto, responde a las peticiones ARP por todas las interfaces que la reciben, asociando la dirección IP a las distintas direcciones MAC de cada interfaz. Debido a este peculiar diseño, cuando varias interfaces de un equipo se conectan al mismo medio de transmisión, ocurre un problema conocido como *ARP flux*, donde todas las interfaces responden a la petición ARP con su IP. Este problema se puede solucionar de dos maneras (ambas funcionan en nuestro entorno):

- Ejecutar:

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

- O bien, ejecutar:

```
# sysctl -w net.ipv4.conf.all.arp_ignore=1
# sysctl -w net.ipv4.conf.all.arp_announce=2
```

2.3.1 Tabla ARP

La tabla ARP se puede ver o modificar mediante las siguientes órdenes:

- Orden `arp`
- Orden `ip neighbour`

Estas órdenes se estudian en los dos apartados siguientes.

Ejercicios

1. Configurar tres máquinas virtuales `uml1`, `uml2` y `uml3`, hacer `ping` entre ellas y utilizar un analizador de tráfico de red (`wireshark`) para visualizar los paquetes ARP.

2.3.2 Manipulación de la tabla ARP mediante la orden `arp`

La orden `arp` nos permite visualizar o modificar la tabla ARP. Para visualizar la tabla ARP completa:

```
# arp -an
```

Para visualizar la tabla ARP asociada a la interfaz de red `eth1`:

```
# arp -an -i eth1
```

Borrar una entrada de la tabla ARP (con dirección IP `192.168.1.2`), asociada a la interfaz `eth1`:

```
# arp -d 192.168.1.2 -i eth1
```

Añadir una entrada **permanente** a la tabla ARP (con dirección IP `192.168.1.2` y dirección MAC `00:40:F4:9A:63:FE`), asociada a la interfaz `eth1`:

```
# arp -s 192.168.1.2 00:40:F4:9A:63:FE -i eth1
```

Añadir una entrada **temporal** a la tabla ARP (con dirección IP `192.168.1.2` y dirección MAC `00:40:F4:9A:63:FE`), asociada a la interfaz `eth1`:

```
# arp -s 192.168.1.2 00:40:F4:9A:63:FE -i eth1 temp
```

Ejercicios

En las dos máquinas virtuales, `uml1` y `uml2`, configuradas para que estén en la misma subred `192.168.1.0/24`:

1. Hacer `ping` desde la máquina `uml1` a la máquina `uml2` y visualizar su entrada en la tabla ARP mediante la orden `arp`.
2. Eliminar de la tabla ARP la entrada correspondiente a la máquina `uml2` mediante la orden `arp`. Comprobar que se ha eliminado correctamente.
3. Añadir de nuevo a la tabla ARP la dirección IP y MAC de la máquina `uml2` mediante la orden `arp` de forma permanente. Comprobar que se ha añadido correctamente.
4. Eliminar de nuevo de la tabla ARP la entrada correspondiente a la máquina `uml2` mediante la orden `arp`. Comprobar que se ha eliminado correctamente.
5. Añadir de nuevo a la tabla ARP la dirección IP y MAC de la máquina `uml2` mediante la orden `arp`, pero ahora de forma temporal. Comprobar que se ha añadido correctamente.
6. Eliminar de nuevo de la tabla ARP la entrada correspondiente a la máquina `uml2` mediante la orden `arp`. Comprobar que se ha eliminado correctamente.
7. Añadir de nuevo a la tabla ARP la dirección IP de la máquina `uml2`, pero esta vez con una dirección MAC falsa (por ejemplo, `02:11:22:33:44:55`). Comprobar mediante `ping` que ahora la máquina `uml2` no es accesible. Esta técnica se denomina *ARP cache poisoning* o envenenamiento de la tabla ARP.

8. Eliminar de la tabla ARP la entrada errónea introducida en el ejercicio anterior, mediante `arp`.

2.3.3 Manipulación de la tabla ARP mediante la orden `ip neighbour`

La orden `ip neighbour` nos permite, también, visualizar y manipular la tabla ARP. Para visualizar la tabla ARP completa:

```
# ip neighbour show
```

Para visualizar la tabla ARP asociada a la interfaz de red `eth1`:

```
# ip neighbour show dev eth1
```

Para borrar una entrada de la tabla ARP (con dirección IP `192.168.1.2`), asociada a la interfaz `eth1`:

```
#  ip neighbour delete 192.168.1.2 dev eth1
```

Para añadir una entrada **permanente** a la tabla ARP (con dirección IP `192.168.1.2` y dirección MAC `00:40:F4:9A:63:FE`), asociada a la interfaz `eth1`:

```
 # ip neighbour add 192.168.1.2 lladdr 00:40:F4:9A:63:FE dev eth1
```

Para añadir una entrada **temporal** a la tabla ARP (con dirección IP `192.168.1.2` y dirección MAC `00:40:F4:9A:63:FE`), asociada a la interfaz `eth1`:

```
 # ip neighbour add 192.168.1.2 lladdr 00:40:F4:9A:63:FE dev eth1 nud reachable
```

Abreviaturas para la orden `ip neighbour`:

```
show      = sh, list, lst, ls, l
neighbour = neighbor, neigh, n
add       = a
delete    = del, d
```

Ejercicios

Repetir de nuevo todos los ejercicios del apartado anterior, pero esta vez usando la orden `ip neighbour`.

2.3.4 Deshabilitar el protocolo ARP en la interfaz de red

La interfaz de red se puede configurar con el protocolo ARP deshabilitado. En esta situación, para poder comunicarse con otros equipos de la red, es necesario introducir manualmente sus respectivas direcciones IP y MAC en la tabla ARP, salvo que la interfaz sea punto a punto.

El protocolo ARP se puede habilitar y deshabilitar mediante las órdenes `ifconfig` e `ip link`.

Para habilitar o deshabilitar el protocolo ARP en la interfaz `eth1` mediante la orden `ifconfig`:

```
# ifconfig eth1 arp
# ifconfig eth1 -arp
```

Para habilitar o deshabilitar el protocolo ARP en la interfaz `eth1` mediante la orden `ip link`:

```
# ip link set dev eth1 arp on
# ip link set dev eth1 arp off
```

Cuando el protocolo ARP está deshabilitado, al visualizar la configuración de la interfaz de red (usando la orden `ifconfig eth1` o `ip link dev eth1`), se muestra el indicador `NOARP` activado.

Ejercicios

1. En la máquina `uml1`, hacer `ping` a la máquina `uml2` y visualizar su entrada en la tabla ARP mediante cualquiera de las órdenes conocidas (`arp` o `ip neighbour`). Anotar la dirección MAC de dicho equipo.
2. Eliminar de la tabla ARP la entrada correspondiente a la máquina `uml2`, mediante la orden `arp` o `ip neighbour`.
3. Deshabilitar el protocolo ARP en la interfaz `eth1` mediante la orden `ifconfig` o `ip link`.
4. Comprobar mediante `ping` que ahora la máquina `uml2` no es alcanzable, ya que la interfaz no puede averiguar su dirección MAC.
5. Añadir manualmente a la tabla ARP la dirección IP y MAC correspondiente a la máquina `uml2`, mediante la orden `arp` o `ip neighbour`.
6. Comprobar mediante `ping` que ahora la máquina `uml2` sí es alcanzable.
7. Habilitar de nuevo el protocolo ARP en la interfaz `eth1` mediante la orden `ifconfig` o `ip link`.

Práctica 3: Encaminamiento IP

3.1 Configuración básica de tablas de rutas

En esta sección se estudiará el manejo básico de tablas de rutas:

- Visualizar e interpretar las tablas de rutas del *kernel*.
- Añadir y eliminar un encaminador predeterminado (*default router*).
- Añadir y eliminar una ruta a una red destino

Estas operaciones se pueden llevar a cabo mediante las siguientes órdenes:

- Orden `route`
- Orden `ip route`

NOTA: para realizar los distintos ejercicios de esta práctica se recomienda arrancar una configuración de red con 5 máquinas virtuales `uml`.

3.1.1 Visualizar e interpretar tablas de rutas

La tabla de rutas se puede visualizar mediante la orden `route`:

```
# route -n
Destination   Gateway          Genmask          Flags Metric Ref Use
Iface
192.168.0.0   0.0.0.0          255.255.255.0    U     0        0    0
eth1
192.168.1.0   0.0.0.0          255.255.255.0    U     0        0    0
eth2
0.0.0.0        192.168.1.100   0.0.0.0          UG    0        0    0
eth2
```

En esta tabla de rutas se muestran tres rutas distintas, que se interpretan de la siguiente forma:

- Las dos primeras rutas, con el campo `Gateway = 0.0.0.0`, hacen referencia a las dos redes locales a las que están conectadas respectivamente las interfaces de red `eth1` y `eth2`.
- La tercera ruta, con el campo `Destination = 0.0.0.0`, indica cuál es el encaminador predeterminado (*default router*). Este encaminador es accesible a través de la interfaz `eth2`.

La orden `netstat -rn` produce una salida similar.

La tabla de rutas también se puede visualizar mediante la orden `ip route`:

```
# ip route show
192.168.0.0/24 dev eth1  proto kernel  scope link  src 192.168.0.1
192.168.1.0/24 dev eth2  proto kernel  scope link  src 192.168.1.1
default via 192.168.1.100 dev eth2
```

En esta tabla de rutas se muestran tres rutas distintas, que se interpretan de la siguiente forma:

- Las dos primeras rutas de la tabla hacen referencia a las dos redes locales a las que están conectadas, respectivamente, las interfaces `eth1` y `eth2`. Por ejemplo, en la primera entrada se pueden distinguir los siguientes campos:

- `192.168.0.0/24`: indica la red/máscara destino de la ruta.
- `dev eth1`: interfaz de red asociada a la ruta (por la que se envían los datagramas para alcanzar el destino de la ruta).
- `proto kernel`: indica que la ruta fue añadida por el *kernel* durante la configuración de la interfaz de red.
- `scope link`: indica que la ruta sólo es válida para acceder al enlace o red local.
- `src 192.168.0.1`: indica la dirección fuente de los datagramas expedidos por la interfaz de red asociada a la ruta.

- La tercera ruta de la tabla indica cuál es el encaminador predeterminado. En esta entrada se pueden distinguir los siguientes campos:
  - `default`: indica el destino de la ruta, que puede ser cualquiera (ruta asociada al *default router*).
  - `via 192.168.1.100`: indica cuál es el encaminador que se utiliza para alcanzar el destino (*default router*).
  - `dev eth2`: interfaz de red asociada a la ruta (por la que se envían los datagramas para alcanzar el destino de la ruta).

Ejercicios

1. Configurar en las máquinas `uml1` y `uml2` las interfaces de red `eth1` y `eth2` con las siguientes direcciones:

`eth1`	Dirección IP: `192.168.1.<máquina>` Máscara: `255.255.255.0`
`eth2`	Dirección IP: `192.168.2.<máquina>` Máscara: `255.255.255.0`

2. Visualizar la tabla de rutas con las órdenes `route` e `ip route`.
3. Desactivar (*down*) la interfaz `eth1` y volver a visualizar la tabla de rutas con las órdenes `route` e `ip route`.
4. Desactivar (*down*) la interfaz `eth2` y volver a visualizar la tabla de rutas con las órdenes `route` e `ip route`.

3.1.2 Añadir/eliminar un encaminador predeterminado a la tabla de rutas

Para añadir o eliminar un encaminador predeterminado

- Mediante la orden `route`:

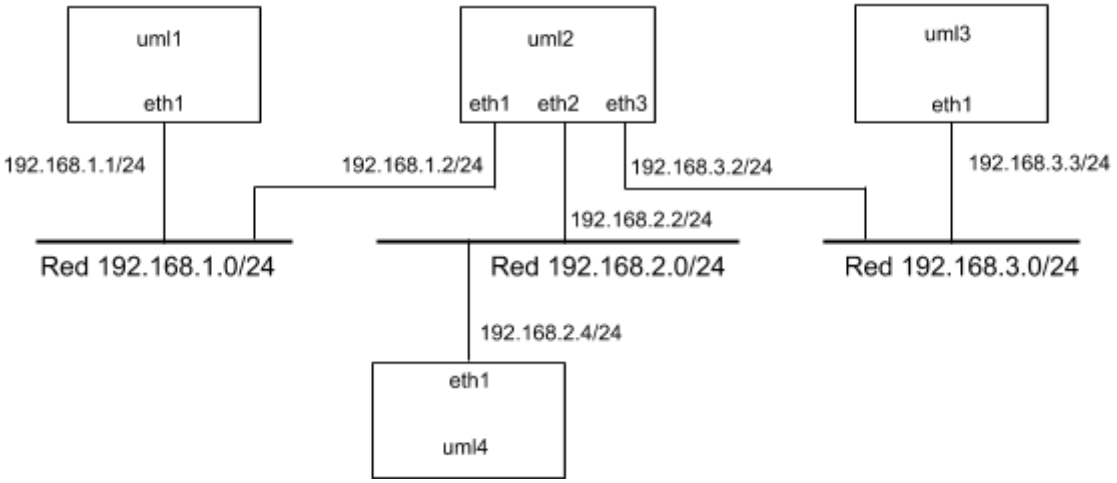
```
# route add default gw 192.168.1.100
# route delete default gw 192.168.1.100
```

- Mediante la orden `ip route`:

```
# ip route add default via 192.168.1.100
# ip route delete default
```

Ejercicios

1. Configurar cuatro máquinas virtuales tal y como se muestra en la siguiente figura:



2. Comprobar mediante `ping` que las máquinas `uml1`, `uml3 y uml4` no son accesibles entre sí.
3. En las máquinas `uml1`, `uml3 y uml4`, añadir como encaminador predeterminado a `uml2` usando la orden `route`. Cada máquina tendrá que añadir la dirección del encaminador asociada a la red en la que se encuentra.
4. Visualizar la tabla de rutas para ver la nueva ruta añadida. Comprobar que las máquinas de las otras redes siguen sin ser accesibles.
5. Ejecutar, en la máquina `uml2`, que actúa como encaminador, la siguiente orden:

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

   Esta orden activa el reenvío (*forwarding*) entre las interfaces del encaminador.
6. Comprobar que todas las máquinas son accesibles entre sí
7. Visualizar las tablas de rutas de todas las máquinas virtuales
8. Eliminar la ruta asociada al *default router* usando la orden `route` y volver a añadirla usando la orden `ip route`.

3.1.3 Añadir/eliminar una ruta a una red

Para añadir o eliminar una ruta a una red destino (p.ej. red 192.168.2.0/24) a través de un encaminador (p.ej. encaminador 192.168.1.100)

- Mediante la orden `route`:

```
# route add -net 192.168.2.0/24 gw 192.168.1.100
# route delete -net 192.168.2.0/24 gw 192.168.1.100
```

- Mediante la orden `ip route`:

```
# ip route add 192.168.2.0/24 via 192.168.1.100
# ip route delete 192.168.2.0/24 via 192.168.1.100
```

Ejercicios

1. Configurar cinco máquinas virtuales tal y como se muestra en la siguiente figura:

2. En las máquinas `uml1` y `uml5`, como sólo tienen un encaminador accesible (`uml2` y `uml4` respectivamente) podemos configurarlos, respectivamente, como su encaminador predeterminado usando la orden `ip route`
3. En la máquina `uml3` configurar manualmente una ruta para la red `192.168.1.0/24` a través del encaminador `uml2` y otra ruta para la red `192.168.3.0/24` a través del encaminador `uml4` usando la orden `ip route`
4. En las máquinas `uml2` y `uml4`, que actúan como encaminadores, activar el *forwarding* mediante la orden:

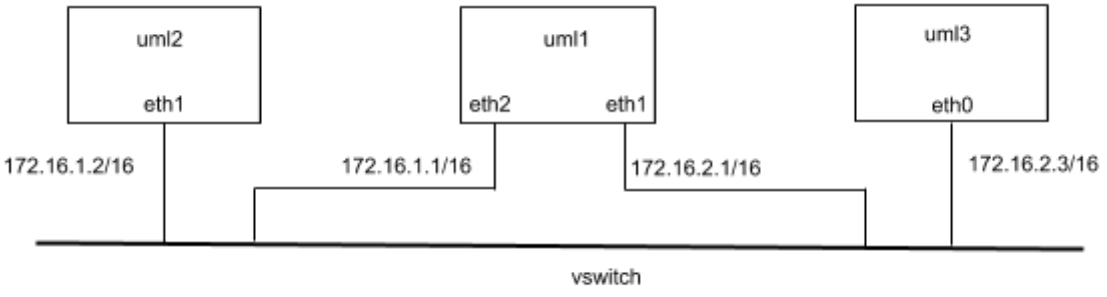
```
# sysctl -w net.ipv4.conf.all.forwarding=1
```

5. En el encaminador `uml2` configurar manualmente una ruta para la red `192.168.3.0/24` a través del encaminador `uml4` usando la orden `ip route`
6. En el encaminador `uml4` configurar manualmente una ruta para la red `192.168.1.0/24` a través del encaminador `uml2` usando la orden `ip route`
7. Comprobar que todas las máquinas son accesibles entre sí
8. Visualizar las tablas de rutas de todas las máquinas virtuales
9. Borrar todas las entradas de las tablas de rutas añadidas anteriormente con la orden `ip route` y volverlas a añadir usando la orden `route`

3.2 Creación de subredes

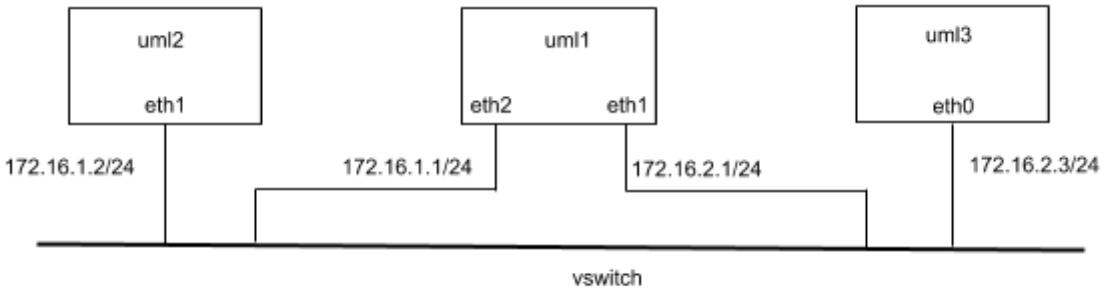
Ejercicios

1. Creación de subredes a partir de una red de clase B.
   a) Creación de una red de clase B: asignar a tres máquinas virtuales las direcciones IP `172.16.x.y`, con máscara `255.255.0.0` y dirección de difusión `172.16.255.255`, según la figura:



   - Visualizar la tabla de rutas y comprobar que todas las máquinas están en la misma red (identificador de red `172.16.0.0`).
   - Comprobar mediante `ping` que todas las máquinas de la red son alcanzables entre sí.

b) Organización de la red de clase B en subredes: asignar a las máquinas de la red anterior la máscara `255.255.255.0` y la dirección de difusión adecuada, según la figura:



- Visualizar la tabla de rutas y comprobar que `uml2` y `uml3` están en subredes distintas (subredes `172.16.1.0/24` y `172.16.2.0/24` respectivamente).
- Comprobar mediante `ping` que las máquinas de subredes distintas no son alcanzables entre sí.

c) Unión de la subredes mediante un encaminador: configurar la máquina `uml1` para que actúe de encaminador entre las dos subredes:

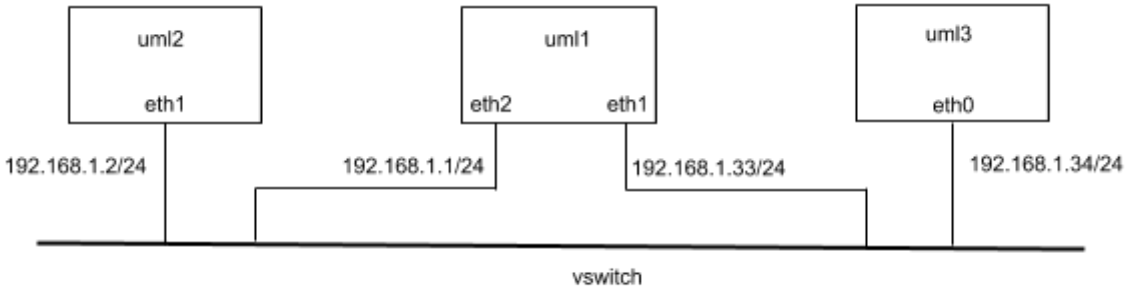
- En las máquinas `uml2` y `uml3`, añadir como encaminador predeterminado la dirección de `uml1` asociada a cada subred.
- Activar el reenvío de datagramas en `uml1` mediante la orden:

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

- Comprobar mediante `ping` que ahora las máquinas `uml2` y `uml3` sí son alcanzables entre sí.

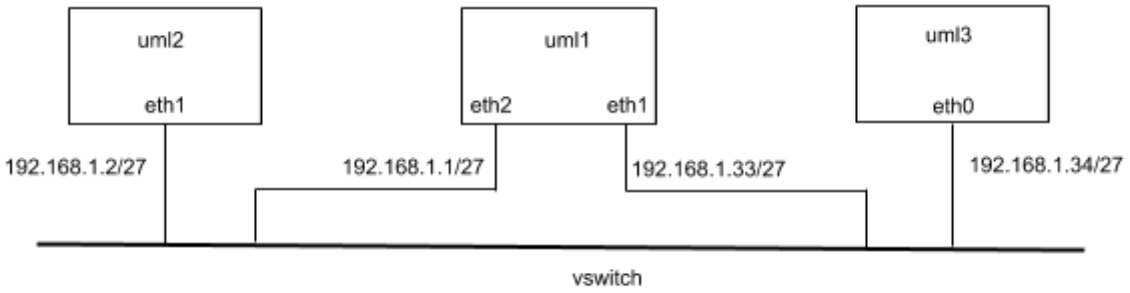
2. Creación de subredes a partir de una red de clase C

a) Creación de una red de clase C: asignar a las máquinas de la red la dirección IP `192.168.1.x`, con máscara `255.255.255.0` y dirección de difusión `192.168.1.255`, según la figura:



- Visualizar la tabla de rutas y comprobar que todas las máquinas están en la misma red (identificador de red `192.168.1.0`).
- Comprobar mediante `ping` que todas las máquinas de la red son alcanzables entre sí.

b) Organización de la red de clase C en subredes: asignar a las máquinas de la red anterior la máscara `255.255.255.224` y la dirección de difusión adecuada, según la figura:



- Visualizar la tabla de rutas y comprobar que `uml2` y `uml3` están en subredes distintas (subredes `192.168.1.0/27` y `192.168.1.32/27` respectivamente).
- Comprobar mediante `ping` que las máquinas de subredes distintas no son accesibles entre sí.

c) Unión de la subredes mediante un encaminador: configurar la máquina `uml1` para que actúe de encaminador entre las dos subredes:

- En las máquinas `uml2` y `uml3`, añadir como encaminador predeterminado la dirección de `uml1` asociada a cada subred.
- Activar el reenvío de datagramas en `uml1` mediante la orden:

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

- Comprobar mediante `ping` que las máquinas `uml2` y `uml3` ahora sí son alcanzables entre sí.

3.3 Fragmentación y reensamblado

La fragmentación puede realizarse en dos lugares: en el origen del datagrama o en alguno de los encaminadores intermedios. Cuando se usa **fragmentación en origen**, la máquina debe conocer la mínima MTU hasta el destino, lo que se conoce como **descubrimiento de MTU** (*path MTU discovery*).

La mayor parte de los sistemas modernos activan por defecto el bit DF (*Don't Fragment*) y utilizan el procedimiento de descubrimiento de MTU (descrito en el RFC 1191). El mecanismo es sencillo:

- La máquina origen envía el datagrama ajustándose a la MTU de su red local y con el bit DF activado.
- Si el datagrama debe atravesar una red con una MTU menor, el encaminador correspondiente se encuentra con que no puede fragmentarlo, al estar el bit DF activado, por lo que descarta el datagrama y envía un mensaje ICMP de tipo **destino inalcanzable**, subtipo **fragmentación necesaria**, al origen. Este mensaje ICMP contiene la MTU de la red a la que no se pudo llegar.
- El origen envía un nuevo datagrama, más corto, que se ajusta a la MTU de la red remota. El bit DF se activa de nuevo.
- El proceso se repite hasta que el datagrama llega al destino (no se recibe ningún ICMP de fragmentación).

Se puede controlar el uso o no del algoritmo de descubrimiento de MTU mediante el parámetro del kernel `net.ipv4.ip_no_pmtu_disc`. Cuando dicho parámetro vale 0, que es su valor predeterminado, está activado el descubrimiento de MTU.

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

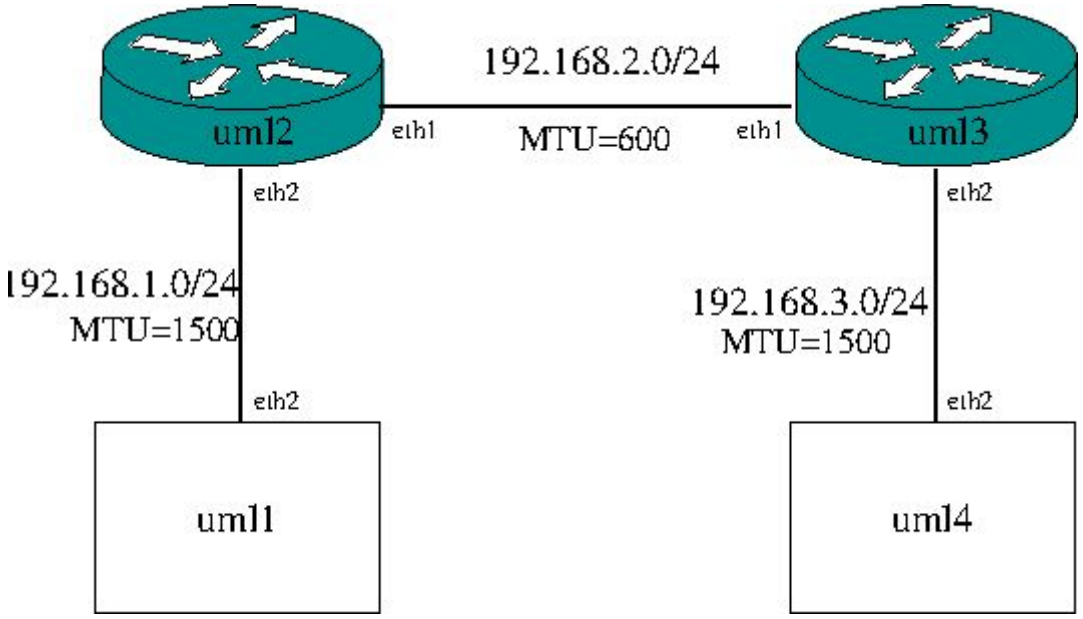
Ejercicios

Utilizar las órdenes `ip` e `ifconfig` para averiguar la MTU de las redes locales.

3.3.1. Fragmentación en origen con descubrimiento de MTU

Ejercicios

Activar 4 máquinas virtuales y configurarlas como indica la siguiente figura:



Las máquinas `uml2` y `uml3` son encaminadores. Por tanto, debemos configurar sus rutas manualmente y activar el reenvío de paquetes (*forwarding*). La red que une ambos encaminadores tiene una MTU de 600 bytes. Como se vio en la primera práctica, se deben utilizar las órdenes `ifconfig` o `ip link` para establecer la MTU correspondiente en las interfaces `eth1` de los encaminadores.

La máquina `uml1` debe usar `uml2` como encaminador predeterminado, y `uml4` usará `uml3`, de manera que si hacemos `ping` desde la máquina `uml1` hasta `uml4`, debería ser accesible.

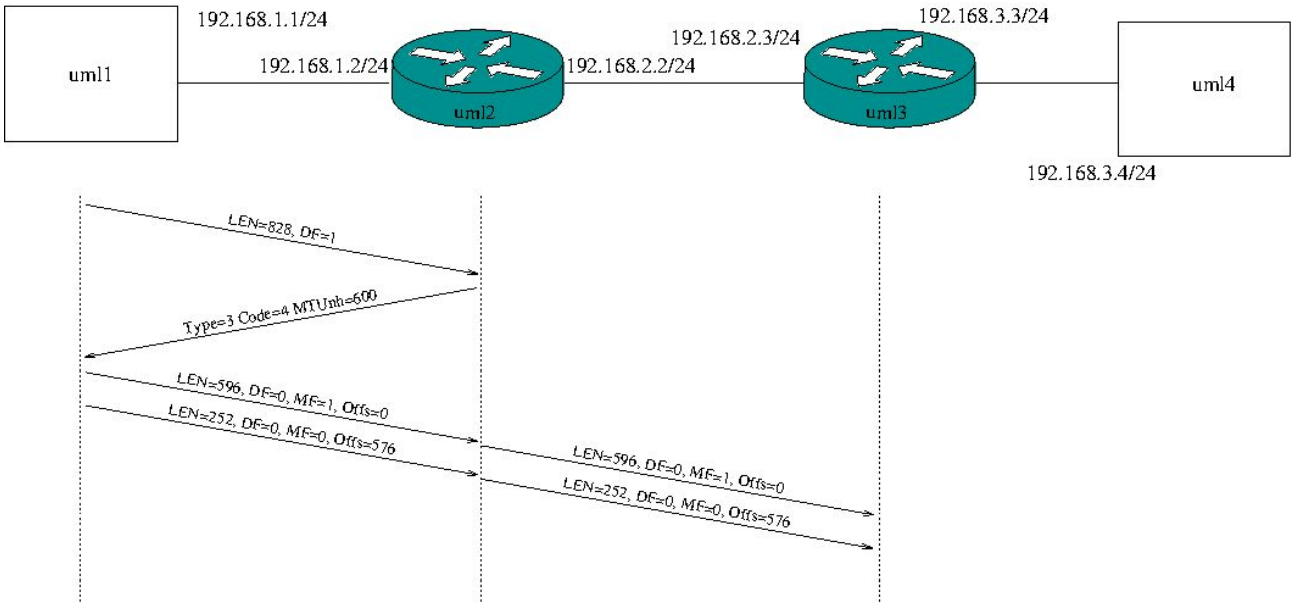
Iniciamos `wireshark` en la máquina anfitriona e iniciamos la captura del tráfico.

Suponiendo que hemos configurado la máquina `uml4` con la dirección IP `192.168.3.4`, estudiar el resultado de la siguiente orden desde `uml1`:

```
# ping -c 3 -s 800 192.168.3.4
```

Como `net.ipv4.ip_no_pmtu_disc` vale 0 (valor por defecto), se activa el descubrimiento de MTU. Cuando `uml1` hace `ping` con el parámetro `-s 800`, se envía un mensaje ICMP de tipo *Echo request* con un campo de datos de 800 bytes. El primer datagrama lleva el bit `DF` activado y una longitud de 828 bytes (800 de datos, 8 de cabecera ICMP y 20 de cabecera IP). La máquina `uml2` lo descarta y envía un ICMP con `Type=3` (*Destination unreachable*), `Code=4`

(*Fragmentation needed*) y `MTU of next hop=600`. Ahora `uml1` fragmentará el datagrama **en origen** como se muestra en la siguiente figura:



Una vez que se ha descubierto la MTU para esa ruta, se añade una entrada en la **tabla *cache* de encaminamiento**. Conocida como Tabla de Información de Reenvío (FIB, *Forwarding Information Base*), es una tabla dinámica que almacena las entradas de rutas usadas recientemente por el sistema. Es la primera que se consulta a la hora de tomar una decisión de encaminamiento. Si existe una entrada apropiada en esta tabla, el sistema reenviará el datagrama inmediatamente. En caso contrario, consultará la tabla de rutas. Se puede ver su contenido con la siguiente orden:

```
# ip route show table cache.
```

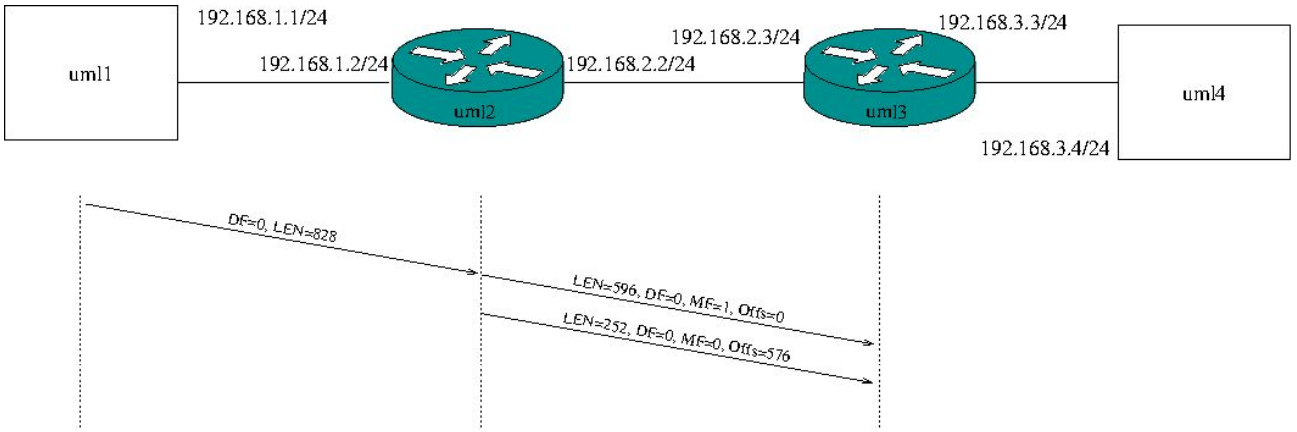
Por tanto, si queremos repetir el proceso de descubrimiento de MTU, previamente debemos borrar la entrada en la tabla. Se puede vaciar la tabla con la siguiente orden:

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

3.3.2. Desactivación del descubrimiento de MTU

Ejercicios

Usando la misma configuración del apartado anterior, estableceremos `net.ipv4.ip_no_pmtu_disc=1` en todas las máquinas, de manera que los datagramas que parten de `uml1` no llevan activado el bit `DF` y el encaminador `uml2` puede fragmentarlos sin problemas, como se ve en la siguiente figura:



Podemos comprobarlo inspeccionando el tráfico con `wireshark`.

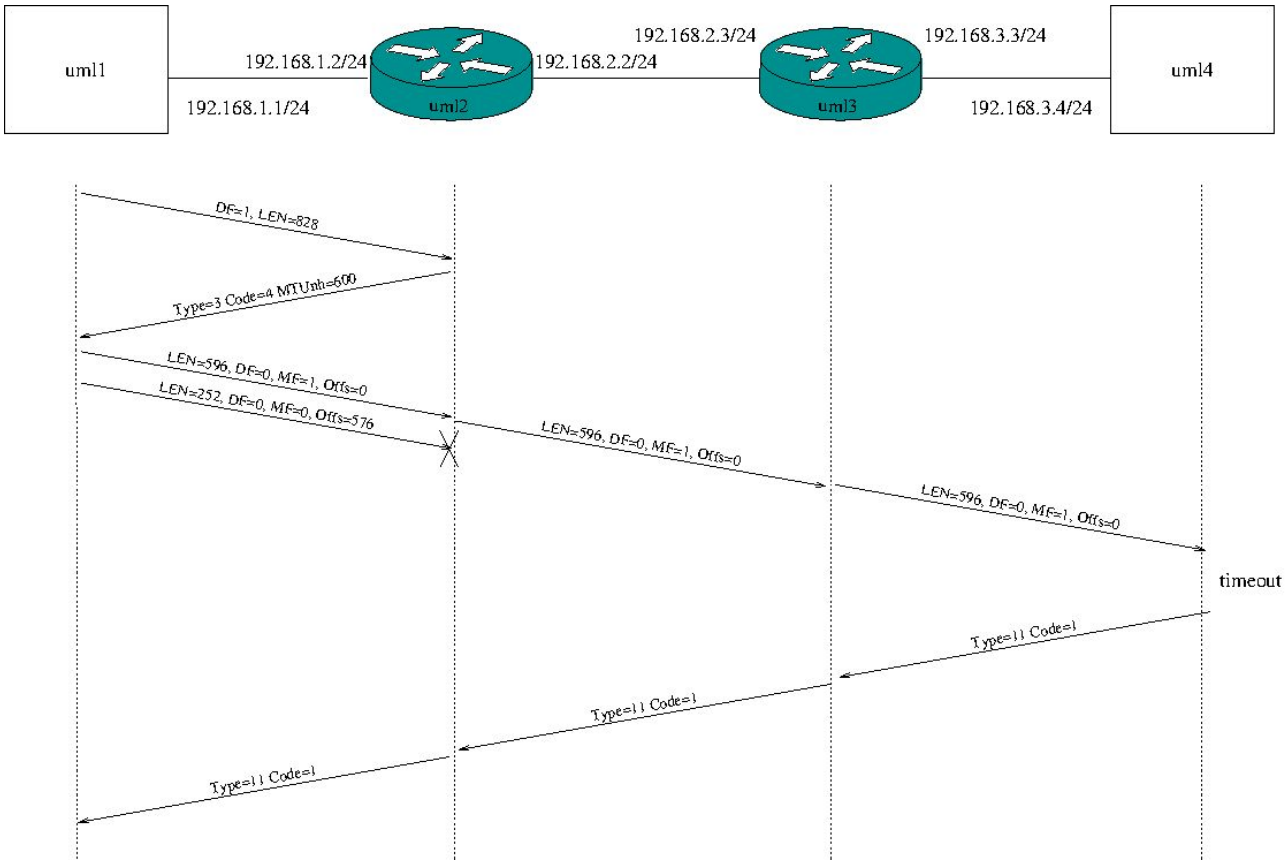
3.3.3. Agotamiento del tiempo de reensamblado

Ejercicios

Usando la misma configuración de los apartados anteriores, incluiremos en el encaminador `uml2` una regla de filtrado de paquetes para descartar el segundo fragmento y sucesivos (cuyo campo desplazamiento es distinto de cero):

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

De esta manera, solo dejará pasar el primer fragmento de cada datagrama, simulando de esta manera la pérdida de un fragmento. Si `uml1` no fragmenta en origen, no habrá problema, pero si lo hace, `uml2` no permitirá pasar el segundo fragmento. Cuando `uml4` recibe el primero, queda a la espera de los siguientes (el bit `MF` viene activado). Cuando se cumple el temporizador de ensamblado, descarta el datagrama y envía a `uml1` un mensaje ICMP con `Type=11` (*Time–to–live exceeded*) y `Code=1` (*Fragment reassembly time exceeded*), como se ve en la siguiente figura:



El temporizador de reensamblado se controla con la variable `net.ipv4.ipfrag_time` (por defecto, 30 segundos). Se puede repetir el proceso anterior modificando el valor de esta variable en la máquina `uml4`.

Práctica 4. Protocolos de transporte UDP y TCP

4.1 Puertos bien conocidos

El listado de puertos bien conocidos se encuentra en el archivo del sistema `/etc/services`.

```
# cat /etc/services
-----------------------------------------------------
tcpmux          1/tcp    # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp    sink null
discard         9/udp    sink null
systat          11/tcp   users
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
...
-----------------------------------------------------
```

Ejercicios

1. Listar el archivo `/etc/services` de una de las máquinas virtuales y buscar los puertos bien conocidos asociados a los siguientes servicios: FTP, SSH, SMTP, POP3, HTTP (www), HTTPS, DNS (domain).

4.2 La herramienta `netcat`

La herramienta `netcat` o, abreviadamente, `nc` es una aplicación versátil para redes TCP/IP con múltiples usos desde el punto de vista de administración. En esta sección se estudia el uso de esta herramienta para la creación de sencillas aplicaciones de tipo cliente-servidor, tanto TCP como UDP.

4.2.1 Cliente-servidor TCP con `netcat`

Servidor TCP con `netcat`

Para crear un servidor TCP con `netcat`, usar la siguiente orden:

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

Las opciones de esta orden son:

- `-l` Indica que debe actuar como servidor, abriendo un puerto TCP en modo escucha (listen)
- `-p` A continuación, se especifica el número de puerto del servidor (server_port)

Cliente TCP con `netcat`

Para crear un cliente TCP con `netcat`, usar la siguiente orden:

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

Ejemplo

Supongamos que el cliente se ejecuta en la máquina 192.168.1.1 y el servidor en la máquina 192.168.1.2 y que el puerto servidor es el número 60000.

```
192.168.1.2# nc -l -p 60000
192.168.1.1# nc 192.168.1.2 60000
```

A continuación, cualquier cadena de texto que se introduzca en el lado del cliente, se envía al servidor y viceversa.

Es importante notar que un servidor abierto con `netcat` sólo puede atender a un cliente. De este modo, cuando salgamos del programa cliente (mediante Ctrl-C), el programa servidor también finaliza su ejecución. Igualmente, si salimos primero del servidor (mediante Ctrl-C), el programa cliente también finaliza.

Ejercicios

1. Usando dos máquinas virtuales, crear un par de aplicaciones cliente-servidor TCP con `netcat` usando el puerto servidor número 60000.
2. Usando un analizador de red (`wireshark`) visualizar el tráfico que intercambian ambas aplicaciones.

4.2.2 Cliente-servidor UDP con `netcat`

Servidor UDP con `netcat`

Para crear un servidor UDP con `netcat`, usar la siguiente orden:

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

Las opciones de esta orden son:

- `-l` Indica a netcat que debe actuar como servidor
- `-u` Indica a netcat que se tiene que abrir un puerto UDP
- `-p` A continuación se especifica el número de puerto del servidor (server_port)

Cliente UDP con `netcat`

Para crear un cliente UDP con `netcat`, usar la siguiente orden:

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

Ejemplo

Supongamos que el cliente se ejecuta en la máquina 192.168.1.1 y el servidor en la máquina 192.168.1.2 y que el puerto del servidor es el número 20000.

```
192.168.1.2# nc  -l  -u  -p 20000
192.168.1.1# nc  -u  192.168.1.2  20000
```

A continuación, cualquier cadena de texto que se introduzca en el lado del cliente, se envía al servidor y viceversa.

Ejercicios

1. En dos máquinas virtuales, crear un par de aplicaciones cliente-servidor UDP con `netcat` usando el puerto servidor número 20000.
2. Usando un analizador de red (`wireshark`) visualizar el tráfico que intercambian ambas aplicaciones.

4.3 El protocolo UDP

4.3.1 Formato del datagrama UDP

Repasar el formato del datagrama UDP en los apuntes de teoría.

Ejercicios

1. En dos máquinas virtuales, crear un par de aplicaciones cliente-servidor UDP con `netcat` usando el puerto servidor número 20000.
2. Usando un analizador de red (`wireshark`) visualizar el tráfico que intercambian ambas aplicaciones e identificar los campos de la cabecera UDP.

4.3.2 Listado de puertos UDP con `netstat`

Para visualizar el listado de puertos UDP abiertos en un sistema se puede usar la orden `netstat`, con las siguientes opciones:

```
  # netstat -ua
  Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address     Foreign Address   State
  udp        0      0   *:32770           *:*
  udp        0      0   *:32771           *:*
  udp        0      0   *:echo            *:*
  udp        0      0   *:discard         *:*
  udp        0      0   *:812             *:*
  udp        0      0   *:mdns            *:*
  udp        0      0   *:sunrpc          *:*
  udp        0      0   *:ipp             *:*
```

Si añadimos la opción `-n` se muestra el número de puerto en lugar del nombre del servicio:

```
# netstat -uan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address     Foreign Address   State
```

```
udp        0      0   0.0.0.0:32770     0.0.0.0:*
udp        0      0   0.0.0.0:32771     0.0.0.0:*
udp        0      0   0.0.0.0:7         0.0.0.0:*
udp        0      0   0.0.0.0:9         0.0.0.0:*
udp        0      0   0.0.0.0:812       0.0.0.0:*
udp        0      0   0.0.0.0:5353      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:*
```

Cuando se establece una comunicación entre un cliente y un servidor UDP, la pareja **dirección_IP:puerto_UDP** del cliente y **dirección_IP:puerto_UDP** del servidor se mapean en las columnas `Local Address` y `Foreign Address` de la salida de la orden `netstat`.

Ejemplo

Ejecutamos con `netcat` un servidor UDP en la máquina 192.168.1.2 con número puerto 20000, mediante la siguiente orden:

```
# nc  -l  -u  -p 20000
```

Si visualizamos el listado de puertos UDP en la máquina del servidor, obtenemos la siguiente salida:

```
# netstat -uan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address     Foreign Address   State
...
udp        0      0   0.0.0.0:20000     0.0.0.0:*
...
```

A continuación, ejecutamos el correspondiente cliente UDP en la máquina 192.168.1.1, mediante la siguiente orden:

```
# nc  -u  192.168.1.2  20000
<Introducir cadena de texto por teclado>
```

Si visualizamos de nuevo el listado de puertos UDP en la máquina del servidor, obtenemos la siguiente salida:

```
  # netstat -uan
Active Internet connections (servers and established)
  Proto Recv-Q Send-Q Local Address     Foreign Address   State
  …

  udp        0      0 192.168.1.2:20000 192.168.1.1:34567 ESTABLISHED
  ...
```

Esta línea indica que hay una comunicación establecida entre el servidor UDP `192.168.1.2:20000` y el cliente UDP `192.168.1.1:34567`, siendo 34567 el número de puerto cliente asignado por el sistema operativo.

Ejercicios

1. En un par de máquinas virtuales (máquinas A y B), crear con `netcat` un servidor UDP en la máquina A con el puerto 5000.
2. Comprobar que el puerto UDP 5000 está abierto usando la orden `netstat -uan`.
3. Crear con `netcat` un cliente UDP en la máquina B que se conecte al servidor de la máquina A. A continuación introducir algún dato por teclado
4. Usando de nuevo la orden `netstat -uan`, comprobar que el puerto UDP 5000 tiene una comunicación establecida (ESTABLISHED).

4.3.3 Intento de comunicación con un puerto UDP cerrado

Si un cliente UDP intenta establecer una comunicación con un puerto servidor UDP que está cerrado, la máquina devuelve un mensaje ICMP de tipo **destino inalcanzable** (*Destination unreachable*) por causa de puerto UDP inalcanzable (*Port unreachable*).

Ejercicios

1. Crear con `netcat` un cliente UDP que intente comunicarse a un puerto servidor cerrado. Para ello, en el servidor, no se debe ejecutar el correspondiente servidor `netcat`.
2. Usando un analizador de red (`wireshark`) visualizar los mensajes ICMP de tipo *Destination unreachable* que devuelve el servidor.

4.3.4 Ejemplos de aplicaciones UDP

Ejemplo 1: la aplicación `echo`

La aplicación `echo` es un sencillo programa servidor que devuelve cualquier cadena de texto que le envía el cliente. Esta aplicación tiene asociado el puerto bien conocido número 7 (ver archivo `/etc/services`).

El servidor `echo` se arranca a través del super-demonio de red `inetd`. En caso de que este servicio no esté arrancado, es necesario incluir o descomentar la siguiente línea en el archivo de configuración `/etc/inetd.conf`

```
# cat /etc/inetd.conf
 ...
 echo    dgram   udp     wait    root    internal
 ...
```

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

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

Para comprobar que el puerto del servidor `echo` (puerto UDP número 7) está abierto, se puede ejecutar la siguiente orden:

```
# netstat -uan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address     Foreign Address   State
...
udp        0    0     0.0.0.0:7         0.0.0.0:*
...
```

Para ejecutar un cliente que se conecte al servidor `echo`, se puede utilizar la herramienta `netcat`:

```
# nc  -u  <IP_servidor>  7
```

A continuación, cualquier cadena de texto que se introduzca por teclado en el lado cliente, se transmite al servidor `echo` y éste devuelve la misma cadena al cliente.

4.4 El protocolo TCP

4.4.1 Formato del segmento TCP

Repasar el formato del segmento TCP en los apuntes de teoría.

Ejercicios

1. En dos máquinas virtuales, crear con `netcat` un par de aplicaciones cliente-servidor TCP usando el puerto servidor número 20000.
2. Usando un analizador de red (`wireshark`) visualizar el tráfico que intercambian ambas aplicaciones e identificar los campos de la cabecera TCP.

4.4.2 Listado de puertos TCP con `netstat`

Para visualizar el listado de puertos TCP abiertos en un sistema y el estado de los mismos, se puede usar la orden `netstat -at`.

```
# netstat -ta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address    State
tcp        0      0 *:ssh              *:*                LISTEN
tcp        0      0 localhost:2208     *:*                LISTEN
tcp        0      0 localhost:41602    *:*                LISTEN
tcp        0      0 *:time             *:*                LISTEN
tcp        0      0 *:discard          *:*                LISTEN
tcp        0      0 *:daytime          *:*                LISTEN
tcp        0      0 *:sunrpc           *:*                LISTEN
tcp        0      0 *:auth             *:*                LISTEN
tcp        0      0 localhost:ipp      *:*                LISTEN
tcp        0      0 localhost:smtp     *:*                LISTEN
tcp        0      0 *:44829            *:*                LISTEN
tcp6       0      0 *:ssh              [::]:*             LISTEN
tcp6       0      0 localhost:ipp      [::]:*             LISTEN
```

Si añadimos la opción `-n` se muestra el número de puerto en lugar del nombre del servicio:

```
# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address     Foreign Address    State
tcp        0      0 127.0.0.1:2208    0.0.0.0:*          LISTEN
tcp        0      0 127.0.0.1:41602   0.0.0.0:*          LISTEN
tcp        0      0 0.0.0.0:37        0.0.0.0:*          LISTEN
tcp        0      0 0.0.0.0:9         0.0.0.0:*          LISTEN
tcp        0      0 0.0.0.0:13        0.0.0.0:*          LISTEN
tcp        0      0 0.0.0.0:22        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      0 127.0.0.1:25      0.0.0.0:*           LISTEN
tcp        0      0 0.0.0.0:44829     0.0.0.0:*           LISTEN
tcp6       0      0 :::22             :::*                LISTEN
tcp6       0      0 ::1:25            :::*                LISTEN
tcp6       0      0 ::1:631           :::*                LISTEN
```

Las conexiones TCP se pueden encontrar en diversos estados (que se corresponden con el diagrama de estados de TCP), aunque los tres estados más comunes que aparecen en el listado de la orden `netstat` son los siguientes:

- `LISTEN`**:** significa que el puerto está abierto (en escucha), a la espera de recibir conexiones de clientes.
- `ESTABLISHED`**:** significa que el puerto TCP local (identificado en la columna `Local Address`) ha establecido una conexión con una entidad TCP remota (identificada en la columna `Foreign Address`). Este estado permite el intercambio de datos entre ambos extremos.
- `TIME_WAIT`**:** significa que la aplicación TCP local ha cerrado la conexión, y ésta se mantiene en estado `TIME_WAIT` durante un cierto periodo de tiempo (60 s) antes de liberarse.

Ejemplo de conexión en estado LISTEN

Por ejemplo, ejecutamos con `netcat` un servidor TCP en la máquina 192.168.1.2 con número puerto 60000, mediante la siguiente orden:

```
# nc  -l  -p 60000
```

Si visualizamos el listado de puertos TCP en la máquina del servidor, obtenemos la siguiente salida:

```
# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address    Foreign Address   State
...
tcp        0      0   0.0.0.0:60000    0.0.0.0:*         LISTEN
...
```

Ejemplo de conexión en estado ESTABLISHED

Continuando con el ejemplo anterior, ejecutamos el correspondiente cliente TCP en la máquina 192.168.1.1, mediante la siguiente orden:

```
# nc  192.168.1.2  60000
```

Si visualizamos de nuevo el listado de puertos TCP en la máquina del servidor, obtenemos la siguiente salida:

```
# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address    State
...
```

```
      tcp             0          0  192.168.1.2:60000   192.168.1.1:24516
ESTABLISHED
     ...
```

Esta línea indica que hay una conexión establecida entre un puerto TCP local (puerto servidor, identificado por `Local Address = 192.168.1.2:60000`) y un puerto TCP remoto (puerto cliente, identificado por `Foreign Address = 192.168.1.1:24516`, siendo 24516 el número de puerto cliente asignado por el sistema operativo de la máquina cliente).

Si visualizamos el listado de puertos TCP en la máquina cliente, obtenemos una salida similar, aunque las columnas `Local Address` y `Foreign Address` están intercambiadas con respecto al servidor:

```
# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address     Foreign Address   State
...
tcp        0      0 192.168.1.1:24516 192.168.1.2:60000 ESTABLISHED
...
```

Ejemplo de conexión en estado TIME_WAIT

Cuando uno de los dos extremos (cliente o servidor) cierra la conexión (mediante Ctrl-C), ésta pasa a estado `TIME_WAIT` (sólo en el extremo que solicita el cierre de la conexión). La conexión se mantiene en este estado durante un cierto periodo de tiempo (60 s) y luego se libera.

```
# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address     State
...
      tcp             0          0  192.168.1.2:60000   192.168.1.1:24516
TIME_WAIT
     ...
```

Ejercicios

1. En dos máquinas (máquinas A y B), crear con `netcat` un servidor TCP en la máquina A con el puerto 5000.
2. Comprobar que el puerto TCP 5000 está abierto usando la orden `netstat -tan`.
3. Crear con `netcat` un cliente TCP en la máquina B que se conecte al servidor de la máquina A.
4. Usando de nuevo la orden `netstat -tan`, comprobar que el puerto TCP 5000 tiene una conexión establecida (ESTABLISHED).
5. Cerrar la conexión en el servidor (mediante Ctrl-C) y comprobar que este extremo la conexión pasa a estado TIME_WAIT durante un cierto periodo de tiempo (60 s) y luego se libera.

4.4.3 Establecimiento de conexión TCP

Repasar en los apuntes de teoría el mecanismo de establecimiento de una conexión TCP (protocolo de 3 vías).

Ejercicios

1. En dos máquinas (máquinas A y B), crear con `netcat` un servidor TCP en la máquina A con el puerto 5000 y un cliente en la máquina B que se conecte al servidor de la máquina A.
2. Usando un analizador de red (`wireshark`) analizar los tres primeros segmentos que intercambian ambas aplicaciones durante la fase de establecimiento de conexión. Identificar la siguiente información en dichos segmentos:
    - Los números de puerto del cliente y del servidor.
    - El valor de los flags SYN y ACK de cada segmento.
    - Los números de secuencia y confirmación de cada segmento, identificando los números de secuencia iniciales del cliente (x) y del servidor (y).

Nota: Para ver los números de secuencia reales (no relativos) de los segmentos TCP en la aplicación wireshark, pinchar con el botón derecho del ratón sobre la ventana donde se muestran los números de secuencia, buscar en el menú desplegable la opción "*Protocol Preferences*" y desactivar la opción "*Relative sequence number and window scaling*"

4.4.4 Intento de conexión a un puerto TCP cerrado

Cuando un cliente TCP intenta establecer una conexión con un puerto servidor TCP que está cerrado, la máquina devolverá un segmento con el flag RST (*reset*) activado.

Ejercicios

1. Crear con `netcat` un cliente TCP que intente comunicarse a un puerto servidor cerrado. Para ello, en la otra máquina, no se debe ejecutar el correspondiente servidor `netcat`.
2. Usando un analizador de red (`wireshark`) analizar los segmentos que intercambian ambos extremos. Identificar la siguiente información en dichos segmentos:
    - Los números de puerto del cliente y del servidor.
    - El valor de los flags SYN, ACK y RST de cada segmento.
    - Los números de secuencia y confirmación de cada segmento.

4.4.5 Fin de conexión TCP

Repasar en los apuntes de teoría mecanismo de desconexión TCP, denominado protocolo de 3 vías.

Ejercicios

1. En dos máquinas (máquinas A y B), crear con `netcat` una servidor TCP en la máquina A con el puerto 5000 y un cliente en la máquina B que se conecte al servidor de la máquina A.
2. Una vez establecida la conexión entre cliente y servidor, cerrar la conexión en alguno de los dos extremos (mediante Ctrl-C)
3. Usando un analizador de red (`wireshark`) analizar los tres últimos segmentos que intercambian ambos extremos. Identificar la siguiente información en dichos segmentos:
    - Los números de puerto del cliente y del servidor.
    - El valor de los *flags* FIN y ACK de cada segmento.
    - Los números de secuencia y confirmación de cada segmento.

4.4.6 Ejemplo de aplicaciones TCP

Ejemplo 1: la aplicación Telnet

Telnet es una aplicación cliente-servidor basada en TCP que permite abrir una sesión o *shell* remota. El servidor Telnet tiene asociado el puerto bien conocido número 23 (ver archivo `/etc/services`).

Las aplicaciones cliente y servidor Telnet son las siguientes:

- **Servidor Telnet:** `in.telnetd` (gestionado por `inetd`)
- **Cliente Telnet:** `telnet`

El servidor `in.telnetd` se arranca a través del súper-demonio de red `inetd`. En caso de que este servicio no esté iniciado, es necesario incluir o descomentar la siguiente línea en el archivo de configuración `/etc/inetd.conf`

```
# cat /etc/inetd.conf
...
telnet  stream  tcp  nowait telnetd /usr/sbin/tcpd
/usr/sbin/in.telnetd
...
```

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

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

Para comprobar que el servicio Telnet (puerto TCP número 23) está abierto, se puede ejecutar la siguiente orden:

```
# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address   Foreign Address   State
...
tcp        0      0   0.0.0.0:23      0.0.0.0:*         LISTEN
...
```

El cliente `telnet`, se ejecuta desde línea de comando de la siguiente forma:

```
telnet  <IP_servidor>
```

Una vez establecida la conexión entre el cliente y servidor, se solicita al cliente un nombre de usuario y una contraseña válidas para acceder a la máquina remota (servidor):

```
# telnet 192.168.1.2
Trying 192.168.1.2...
Connected to 192.168.1.2.
Escape character is '^]'.
Debian GNU/Linux 5.0
debian login: nombre_de_usuario
Password: ***********
```

Una vez abierta la sesión o *shell* remota con Telnet, se puede ejecutar cualquier comando UNIX en la máquina remota (`ls`, `pwd`, `ps`, `netstat`, `ifconfig`, etc.). Para salir de una sesión Telnet, ejecutar la orden `exit`.

Uno de los principales inconvenientes de la aplicación Telnet es que toda la información se envía en texto claro (sin cifrar), incluyendo el nombre de usuario y la contraseña. Por tanto, si esta información viaja por la red, es posible que una tercera persona no autorizada pueda capturar esta información (mediante un analizador de red) y conseguir acceder, de forma ilícita, a la máquina remota. Esta es la razón por la que en el ejemplo anterior no permitía la conexión del usuario `root`. En cambio, si creamos un usuario sin privilegios en la máquina virtual, mediante la orden `adduser`, la orden `telnet` sí permitirá el acceso remoto a dicho usuario.

Ejemplo 2: la aplicación Secure Shell

Secure Shell es una aplicación cliente-servidor basada en TCP que permite abrir una sesión o *shell* remota (cliente `ssh`), de forma similar a Telnet, pero con una diferencia fundamental, ya que Secure Shell envía toda la información cifrada mediante un mecanismo de cifrado por clave pública. Además, Secure Shell incluye aplicaciones cliente para el intercambio de ficheros también de forma cifrada (clientes `scp` y `sftp`). El servidor Secure Shell tiene asociado el puerto bien conocido número 22.

Las aplicaciones cliente y servidor de Secure Shell son las siguientes:

- **Servidor Secure Shell:** `sshd`
- **Clientes Secure Shell:** `ssh`, `scp` y `sftp`

El servidor de Secure Shell no suele gestionarse mediante el súper-demonio `inetd` (o `xinetd`), sino que tiene asociado su propio proceso servidor o 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 reiniciar el servidor `sshd`

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

Para comprobar que el servidor `sshd` (puerto TCP número 22) está abierto, se puede ejecutar la siguiente orden:

```
# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q   Local Address    Foreign Address   State
...
 tcp             0      0        0.0.0.0:22           0.0.0.0:*
LISTEN
...
```

El cliente (`ssh`) se ejecuta desde línea de comando, de la siguiente forma:

```
ssh  <usuario>@<IP_servidor>
```

Ejercicios

1. En una máquina virtual deberá crearse un usuario sin privilegios, mediante la orden `adduser`.
2. Cliente-servidor Telnet:
   - En un par de máquinas (máquinas A y B), comprobar que en la máquina A está arrancado el servidor Telnet. Si no lo está, arrancarlo.
   - Arrancar en la máquina B un cliente Telnet que se conecte al servidor de la máquina A.
   - Desde el cliente Telnet, introducir algunas órdenes en la máquina remota (por ejemplo, `ls`, `pwd`, `ps`, `netstat -tan`, `ifconfig eth1`, etc.)
   - Cerrar la conexión Telnet con la orden `exit`.
   - Usando un analizador de red (`wireshark`) visualizar y analizar el tráfico que intercambian ambas aplicaciones, identificando los números de puerto cliente y servidor en las cabeceras TCP, las fases de establecimiento y fin de conexión.
   - En el analizador de red, pinchar sobre cualquiera de los paquetes de la conexión y usar la opción del menú `Analyze → Follow TCP stream`, para visualizar el texto intercambiado entre cliente y servidor. Comprobar que este texto es totalmente legible, incluido el nombre de usuario y la contraseña.
3. Cliente-servidor Secure Shell:
   - En un par de máquinas (máquinas A y B), comprobar que en la máquina A está arrancado el servidor `sshd`. Si no lo está, arrancarlo.
   - Arrancar en la máquina B un cliente `ssh` que se conecte al servidor de la máquina A.
   - Desde el cliente `ssh`, introducir algunas órdenes en la máquina remota (por ejemplo, `ls`, `pwd`, `ps`, `netstat -tan`, `ifconfig eth1`, etc.)
   - Cerrar la conexión Secure Shell con la orden `exit`.
   - Usando un analizador de red (`wireshark`) visualizar y analizar el tráfico que intercambian ambas aplicaciones, identificando los números de puerto cliente y servidor en las cabeceras TCP, las fases de establecimiento y fin de conexión.
   - En el analizador de red, pinchar sobre cualquiera de los paquetes de la conexión y usar la opción del menú `Analyze → Follow TCP stream`, para visualizar el texto intercambiado entre cliente y servidor. Comprobar que el texto mostrado no es legible, ya que toda la información intercambiada está cifrada.

4.4.7 Modificar el número de puerto de un servidor

Como ya sabemos, las aplicaciones más típicas de Internet tienen asociado, por defecto, un puerto servidor bien conocido, que se puede consultar en el archivo `/etc/services`. Sin embargo, es posible modificar el número de puerto asociado a cualquier servidor.

Si modificamos el número de puerto de un servidor, debemos notificar a los potenciales usuarios (clientes) de esa aplicación cuál es el nuevo número de puerto, ya que de lo contrario no podrán conectarse al servidor.

A continuación se explica cómo modificar el número de puerto en dos casos distintos:

- Para una aplicación gestionada por el súper-demonio `inetd` (por ejemplo Telnet)

- Para una aplicación gestionada por su propio proceso o demonio servidor (por ejemplo Secure Shell)

Aplicaciones gestionadas por `inetd`

En el caso de una aplicación gestionada por `inetd`, como por ejemplo, Telnet, para cambiar el número de puerto del servidor, es necesario realizar los siguientes pasos:

1. Editar el archivo `/etc/services`
2. Modificar el número de puerto del servicio Telnet. Por ejemplo:

```
# nano /etc/services
...
telnet          10000/tcp
...
```

3. Reiniciar el proceso `inetd` mediante la siguiente orden:

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

Para que el cliente Telnet se pueda conectar al nuevo número de puerto del servidor (en el ejemplo anterior, el puerto 10000), el cliente se debe iniciar mediante la siguiente orden:

```
telnet  <IP_servidor>  <puerto>
```

Aplicaciones gestionadas por su propio proceso servidor

En el caso de aplicaciones gestionadas por su propio proceso o demonio servidor, como por ejemplo, Secure Shell, cada una de ellas suele tener su propio archivo de configuración, donde normalmente se puede modificar el puerto de servicio.

En el caso de Secure Shell, el archivo de configuración del servidor es `/etc/ssh/sshd_config` y el número de puerto figura en la siguiente línea del archivo:

```
# cat /etc/ssh/sshd_config
...
# What ports, IPs and protocols we listen for
port 22
...
```

Por tanto, para cambiar el número de puerto del servidor `sshd` es necesario realizar los siguientes pasos:

1. Editar el archivo de configuración `/etc/ssh/sshd_config`
2. Modificar el número de puerto. Por ejemplo:

```
# nano /etc/ssh/sshd_config
-----------------------------
...
# What ports, IPs and protocols we listen for
port 20000
...
-----------------------------
```

3. Reiniciar el proceso `sshd` mediante la siguiente orden:

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

Para que el cliente `ssh` se pueda conectar al nuevo número de puerto del servidor (en el ejemplo anterior, el puerto 20000), el cliente se debe iniciar mediante la siguiente orden:

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

Ejercicios

1. Cambiar el número de puerto del servidor Telnet
   - En un par de máquinas (máquinas A y B), cambiar en la máquina A el número de puerto del servidor Telnet, al puerto 10000.
   - Comprobar en la máquina del servidor, mediante la orden `netstat -tan` que el puerto 10000 está abierto (estado LISTEN).
   - Desde la máquina B, conectarse al servidor Telnet (puerto 10000) de la máquina A.
   - Una vez comprobado que funciona el nuevo puerto, volver a configurar el servicio Telnet con su puerto original (el número 23).
2. Cambiar el número de puerto del servidor Secure Shell
   - En un par de máquinas (máquinas A y B), cambiar en la máquina A el número de puerto del servidor `sshd` al puerto 20000.
   - Comprobar en la máquina del servidor, mediante la orden `netstat -tan` que el puerto 20000 está abierto (estado LISTEN).
   - Desde un cliente `ssh` de la máquina B, conectarse al servidor `sshd` (puerto 20000) de la máquina A.
   - Una vez comprobado que funciona el nuevo puerto, volver a configurar el servicio `sshd` con su puerto original (el número 22).