Redes de Computadoras
Guión de Prácticas

Vicente González Ruiz
Depto de Arquitectura de Computadores y Electrónica
vruiz@ual.es
http://www.ace.ual.es/~vruiz/docencia

14 de abril de 2009

Índice

Práctica 1

El laboratorio de redes

Las prácticas de redes de computadoras son realizadas en el laboratorio de redes. Dedicaremos esta primera práctica a conocer el hardware y el software del que disponemos.

1.1. Hardware

En el laboratorio existen 25 computadoras con las siguientes características:

```
* Procesador: AMD Athlon 64 bits 3500
* Memoria RAM: 1GB
* Disco Duro: 80GB ATA
* Adaptadores de red: DLINK DFE-528TX, Linksys NC100, ALi M5263
* Tarjeta de Video: ATI RADEON 9250
* Tarjeta de Sonido: Integrada en placa base Realtek AC´97
```

Como se deduce del tipo de los adaptadores de red que tienen instalados las computadoras (hosts), la red del laboratorio es una red Ethernet. Además, como utiliza conmutadores (switches) Ethernet, podemos decir que se trata de una Ethernet conmutada (hay diferentes tipos de redes Ethernet). Finalmente, si seccionáramos uno de los cables que unen los adaptadores de red de los hosts con los conmutadores, veríamos que se están utilizando enlaces fabricados con pares trenzados de cobre UTP, categoría 5. En Ethernet estos enlaces son full-duplex (hay pares para enviar datos y otros para recibir datos, que funcionan de forma simultánea).

La topología de la red se muestra en la Figura 1.1.

La tasa de transmisión entre un host y su conmutador directamente conectado es de 100 Mbps y la tasa de transmisión entre los conmutadores es de 100 Mbps. Por tanto, la tasa de transmisión entre un host del laboratorio y el resto de Internet es a lo sumo, 100 Mbps.

1.2. Software

Cada computadora tiene instalados (al menos) dos sistemas operativos: Microsoft Windows XP y Linux. Ambos permiten contectarse a Internet mediante la pila de protocolos TCP/IP. Esto significa que seremos capaces de comunicarnos con otros usuarios de la red si el software de red ha sido convenientemente configurado.

1.2.1. Windows

Utilizaremos Windows para las tareas típicas de navegar por Internet, descargar programas, etc. Para acceder a la computadora utilizaremos la cuenta:

Usuario: `redes`

Contraseña: `redes`

Taller 1.1: Si está consultando este documento en un ordenador del laboratorio usando Windows, ya habrá accedido a su cuenta de usuario. Si no es así, hágalo y si existe algún problema, comuníqueselo al profesor.

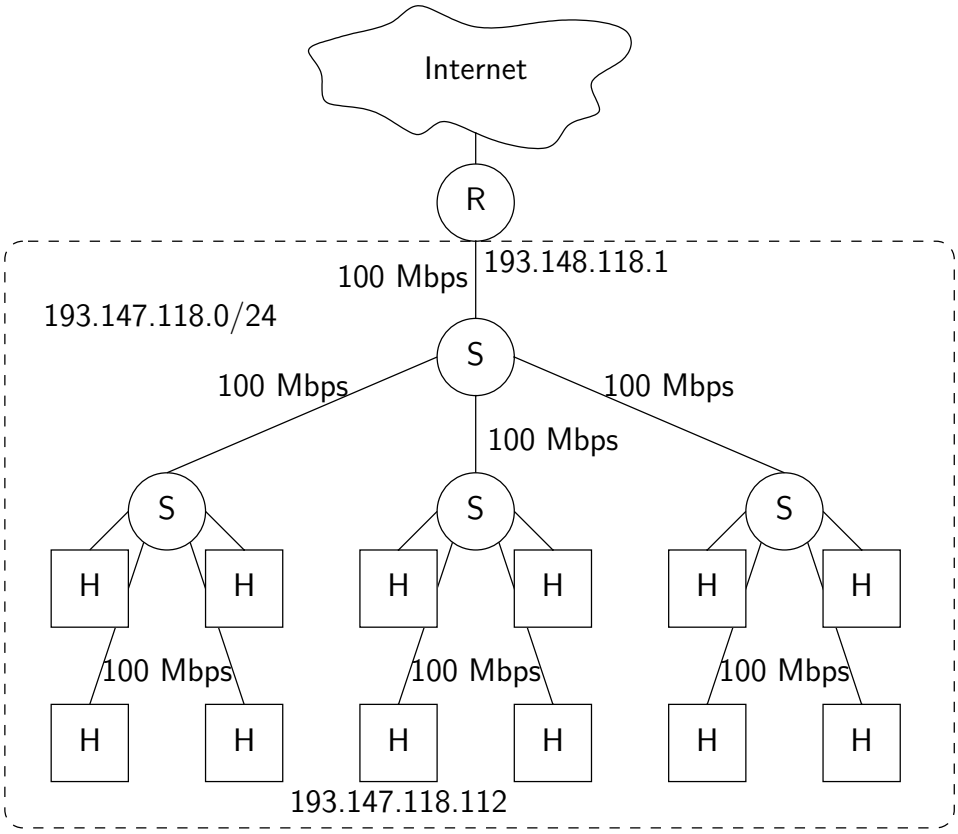


Figura 1.1: Topología física del Laboratorio de Redes.

1.2.2. Fedora Core Linux

Como una alternativa a Windows, podremos utilizar Linux para acceder a la computadora y realizar las prácticas. Los datos del usuario que deberemos utilizar son:

Usuario: `redes`

Contraseña: `redes`

> **Taller 1.2:** Si está consultando este documento en un ordenador del laboratorio usando Linux, ya habrá accedido a su cuenta de usuario. Si no es así, hágalo y si existe algún problema, comuníqueselo al profesor.

1.2.3. Debian Linux del PC virtual

Para realizar la mayoría de las prácticas de redes utilizaremos Linux, Debian Linux, en concreto.

Debido a que es necesario realizar muchas acciones como administrador del sistema, existe un peligro potencial de estropear el buen funcionamiento del sistema operativo. Por otra parte, en algunas prácticas es interesante poder utilizar más de una computadora. Aunque esto podría ser posible físicamente, lo más cómodo es utilizar un software de emulación de PC que permita ejecutar en una misma computadora cuántos PC's virtuales sean necesarios.

Estos dos problemas son resueltos utilizando VMware (`http://www.vmware.com`). Dicho software permite definir una computadora virtual e instalar en ella los programas que queramos. Nosotros vamos a utilizar una distribución Debian Linux (`http://www.debian.org`) instalada en un PC virtual.

1.3. Funcionamiento básico del VMware

VMware consta de muchos programas de los cuales, para realizar las prácticas, sólo necesitaremos el VMware Player (`http://www.vmware.com/download/player/`). Este programa es de libre distribución.

VMware Player permite ejecutar el PC virtual sin necesidad de usar ninguna otra utilidad extra. Para definir un PC virtual se utiliza el programa VMware Workstation (`http://www.vmware.com/download/ws/`) que sí es de pago.

1.3.1. El PC virtual

Con el objetivo de acelerar las prácticas lo máximo posible se ha creado un PC virtual y se le ha instalado Debian Linux. Dicho PC virtual puede descargarse de esta URL: `http://www.ace.ual.es/~{}vruiz/docencia/redes/debian`.

Como puede apreciarse se trata de una colección de ficheros que contienen el contenido del PC virtual. Dicho PC tiene las siguientes características:

```
* Procesador: El mismo que el host físico
* Memoria RAM: 256 MB
* Disco Duro: 4 GB BusLogic BT-958 SCSI
* Adaptador de red: Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
* Tarjeta de Video: VMware Inc [VMware SVGA II] PCI Display Adapter
* Tarjeta de Sonido: La misma que el host físico
```

Taller 1.3: Realice los siguientes pasos:

1. Descargue e instale el programa VMware Player (`http://www.vmware.com/download/player/`) (si no estuviera ya instalado). Seleccione el sistema operativo adecuado en su caso.
2. Descarge el PC virtual. Recuerde que tiene que descargarse todos los ficheros que parecen en la anterior URL.
3. Ejecute el PC virtual.

1.3.2. Usuarios

La instalación de Linux Debian que descargamos tiene creados dos usuarios:

El administrador del sistema: El administrador del sistema en los sistemas Linux (y Unix) se llama `root`. Nuestra instalación de Linux Debian viene con el password `toor` para este usuario.

Por defecto, en Debian Linux no se puede acceder como administrador al sistema X Window. Sin embargo, cambiar esto es sencillo. Cuando vea la pantalla de bienvenida, pinche en "Acciones", luego en "Configurar el gestor de entrada". A continuación teclee la clave del administador (`toor`). Aparecerá una ventana con varias carpetas. Pinche en "Seguridad". Allí active la opción "Permitir entrada local al administador del sistema". Finalmente piche en "Cerrar".

Un usuario: Utilizar el sistema como adminstrador es potencialmente peligroso si no se es un usuario experto. Por tanto, es recomendable utilizar una cuenta de usuario normal para realizar todas las tareas que sean posibles. Por este motivo se ha creado una cuenta con login `alumno` y password `alumno`.

Taller 1.4: Como acabamos de indicar, nos identificaremos en el sistema como `root` sólo cuando sea imprescindible. Una de estas situaciones se da durante la creación de una nueva cuenta de usuario (véase el Apéndice B).

Acceda al PC virtual como administrador y cree una cuenta de usuario. Anote el nombre de usuario (`login`) y clave de acceso (`password`) para poder usar esta cuenta más adelante. Utilize dicha cuenta, particular, para realizar el resto de las prácticas. Para saber cómo crear una cuenta de usuario, lea el Apéndice B.

1.3.3. Conectividad

El Player puede configurarse de tres formas diferentes dependiendo del nivel de conectividad que queremos dotar a los PC's virtuales. Las alternativas son:

1. *Bridged:* El PC virtual es dotado con una IP real, perteneciente a la red a la que pertenece el host donde se ejecuta el PC virtual (host huésped). Esto sólo es posible hacerlo si disponemos de dicha IP.
2. *NAT:* El PC virtual accede a la red a través de un *NAT box*. La IP usada es, por tanto, privada y asignada dinámicamente por un servidor de DHCP instalado en el host huésped (el que ejecuta Windows, por ejemplo).
3. *Host Only:* En este caso el PC virtual no puede conectarse a ningún otro host, independientemente de que sea real o virtual.

Taller 1.5: Para comprobar si hay acceso a Internet desde el PC virtual, actualice los paquetes instalados en él (véase el Apéndice A). Nótese que esta acción sólo puede realizarse como superusuario (usuario `root`).

Cuestión 1: ¿Qué tasa de transmisión a lo sumo vamos a obtener entre dos hosts del laboratorio? Razone su respuesta.

Cuestión 2: Descarge un archivo de Internet usando su navegador. ¿Cuál ha sido la tasa de descarga? Explique cómo y por qué ha realizado dicho cálculo.

Cuestión 3: ¿Cuántos conmutadores, a lo sumo, cruzará un paquete de datos transmitido desde un host del laboratorio hasta cualquier otro host del laboratorio? Razone su respuesta.

Cuestión 4: ¿Cuántos conmutadores, como mínimo, han atravesado el archivo que se ha descargado? Considere la NAT box del VMware como un conmutador más. Razone.

Práctica 2

Atenuación de las señales

2.1. ¿Qué es una señal?

Una señal [11] es cualquier perturbación medible de las características físicas de un medio (el que sirve de transmisión).

2.2. ¿Cuándo transporta información una señal?

Las señales suelen ser impredebibles para el receptor. Cuando una señal es recibida y además ésta es impredecible de alguna manera, la señal aporta información al receptor. También se dice, en este caso, que disminuye la incertidumbre del receptor [13].

2.3. ¿Atenuación?

Todos los medios de transmisión atenúan las señales que transportan. Esta atenuación produce un decremento en la energía que transporta la señal y por tanto, cuanto más alejados están el emisor y el receptor, menor sería dicha energía.

La cantidad de energía que se pierde durante la transmisión depende del tipo de medio[1] y de la frecuencia de la señal. Generalmente, las señales de mayor frecuencia se atenúan más que las señales de menor frecuencia [18]. Por este motivo, por ejemplo, cuando nos acercamos a un lugar donde hay un concierto de música escuchamos primero los sonidos de más baja frecuencia (el bombo de la batería) y posteriormente el resto de instrumentos.

2.4. ¿Qué es la relación señal/ruido?

Los medios de transmisión nunca están totalmente libres de señales indeseables (*ruido*) que pueden interferir en el receptor con la señal que nosotros intentamos transmitir. En adelante, llamemos a esta señal (la que nosotros deseamos transmitir), *señal de datos*.

A la relación entre la energía de señal de datos y la energía del resto de señales que llegan también al receptor (y que perturban la recepción de la señal de datos) es lo que se conoce como *relación señal-ruido* o SNR (Signal-Noise Ratio). Dicho de otra forma:

$$\mathrm{SNR} = \frac{\text{Energía de la señal de datos}}{\text{Energía de la señal de ruido}}$$

2.5. ¿Qué es una señal digital?

Las *señales digitales* son aquellas que transportan información digital [10]. Estas señales se diferencian de las analógicas en que sólo unos determinados valores de la señal tienen un significado en concreto. Por ejemplo, una señal digital puede oscilar entre 0 y 5 voltios. El primer voltaje indicaría que se transmite la información "0" y el segundo que se transmite la información "1". Un voltaje de $2{,}5$ voltios no significa nada.

[1] En los medios lineales (como pueda ser un cable) la pérdida es lineal con la distancia recorrida. En los medios de difusión (como es el aire), la pérdida es exponencial con la distancia.

Cuestión 1: Discuta si el grado de abertura de una puerta es una información digital o analógica.

2.6. La digitalización de señales

Las señales (independientemente de si representan fuentes de información analógicas o digitales) necesitan digitalizarse antes de ser procesadas por un sistema digital (una computadora, por ejemplo). El proceso al que deben ser sometidas se conoce como *digitalización* y consiste básicamente en registrar el valor aproximado de la señal cada cierto intervalo de tiempo.

El número de muestras capturadas durante la digitalización está directamente relacionado con la máxima componente de frecuencia de la señal. Dicha relación es expuesta por el *Teorema del Muestreo Uniforme* de esta forma:

> **Si capturamos $2n$ muestras/segundo de una señal entonces la máxima componente de frecuencia registrada de dicha señal tiene n Hercios.**

La digitalización implica normalmente una pérdida irreversible de parte de la información que transporta la señal, por dos motivos:

1. Si la señal no está limitada en banda por mucho que aumentemos la frecuencia de muestreo no será posible registrar todas sus componentes de frecuencia.
2. Si la señal no es digital, es imposible utilizando un número finito de bits representar cada uno de sus estados.

Evidéntemente, las señales digitales digitalizadas sólo adolecen del primer problema.

2.6.1. Digitalización de señales digitales

Una forma sencilla de crear versiones digitales de señales digitales consiste en replicar tantas veces como sea preciso cada uno de los distintos valores que toma la señal digital. Esto es lo que realiza el programa `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/sampler.c` (véase el Apéndice H.8). Nótese que si la señal no fuera digital este proceso no sería tan simple.

Cuestión 2: Razone por qué no sería tan simple.

En la entrada de `sampler.c`, cada número real (representado físicamente como un número en punto flotante de 32 bits de precisión) indica un estado de la señal digital. La salida es simplemente una replicación de cada uno de estos estados.

Cuestión 3: ¿No cree que sobran bits para representar cada muestra? ¿Cuántos cree que serían realmente necesarios? ¿Por qué cree que se han usado números reales?

Para generar el fichero de entrada para `sampler.c` podemos usar el programa `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/ascii2float.c` (véase el Apéndice H.2).

Finalmente, para averiguar el contenido de un fichero `.float` podemos utilizar el programa `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/float2ascii.c` (véase el Apéndice H.5).

Taller 2.1:

1. Ejecute el PC virtual. El resto de pasos se realizarán en él.
2. Descarge los ficheros fuente necesarios para realizar esta práctica.
3. Compile los fuentes en C. Sitúese en el directorio donde están los fuentes y el fichero `Makefile`. Escriba:

   ```
   make all
   ```

4. Asegúrerse de que los scripts (ficheros con la extensión `.sh`) poseen permiso de ejecución para el dueño del fichero (aparece una "`x`" en la cuarta posición comenzando desde la izquierda cuando escribimos el comando `ls -l` en el directorio en el que hemos descargado el script).

Taller 2.2: Ejecute:

```
ascii2float << EOF | float2ascii
```

y escriba una serie de números reales separados por la tecla <Enter>. Cuando no desee introducir más números escriba EOF y pulse finalmente <Enter>. Nota: con el redirector de flujo de entrada << provocamos que todo lo que tecleemos se convierta en la entrada estándar del programa sobre el que estamos realizando la redirección, hasta que no escribimos el código de fin de entrada (en este caso la cadena EOF). Nota 2: el redirector de flujo | provoca que la salida estándar del programa que queda a la izquierda se la entrada estándar del programa que queda a la derecha.

2.7. Espectro de frecuencias de una señal

Las señales pueden representarse en el dominio del tiempo o en el dominio de la frecuencia [9]. En el primer dominio, la señal queda especificada por el valor de la señal a lo largo del tiempo. En el segundo, la señal queda definida por un conjunto (en principio infinito) de funciones sinusoidales puras (senos y cosenos) que sumadas entre sí generan dicha señal. A la contribución relativa de cada una de estas señales sinusoidales es a lo que se conoce como *espectro de frecuencias de una señal* o *espectro de Fourier*.

El espectro de Fourier $S(f)$ de la señal $s(t)$ es una función compleja y por tanto, tiene una parte real y otra imaginaria. Para tener una idea de la distribución energética del espectro de $S(f)$ suele representarse su módulo o magnitud que se calcula como:

$$|S(f)| = \sqrt{\mathrm{Re}(S(f))^2 + \mathrm{Im}(S(f))^2}$$

donde $\mathrm{Re}(S(f))$ es la parte real de $S(f)$ y $\mathrm{Im}(S(f))$ es la parte imaginaria de $S(f)$. Para calcular $|S(f)|$, siendo $s(t)$ una señal digitalizada, es posible utilizar la DFT (Discrete Fourier Transform) mediante el programa `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/spectrum_analyzer.c` (véase el Apéndice H.9).

Taller 2.3: Ejecute:

```
ascii2float << EOF > signal.float && spectrum_analyzer signal.float
```

e introduzca una señal con un número par de muestras. A la salida vamos a obtener el módulo del espectro de frecuencias de la señal introducida. Nota: el símbolo > redirige la salida estándar del programa que queda a la izquierda sobre el fichero que aparece a la derecha. Nota 2: los símbolos && representan el operador de ejecución condicional. Significa que, sólo si el comando que queda a la izquierda del operador se ejecuta con éxito, entonces el shell ejecutará el comando que queda a la derecha.

2.8. ¿Cómo es el espectro de frecuencias de una señal digital?

El espectro de una señal digital ideal es discreto y aunque acumula la mayor parte de la energía en las bajas frecuencias también ocupa un *ancho de banda* infinito (véase la Figura 2.1 y la teoría de la asignatura de redes). Esto implica que todos los medios van a filtrar de alguna manera las señales digitales transmitidas.

Podemos visualizar el módulo del espectro de una señal digital utilizando el script `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/draw_signal.sh` (véase el Apéndice H.4).

Taller 2.4: Con este programa visualizar el módulo del espectro de la señal "01" es tan simple como:

```
# Visualizando el espectro de la señal 01
rm -f  signal.txt # Borramos el fichero
echo "-1.0" >> signal.txt # El valor -1 representa el bit 0
echo "1.0" >> signal.txt # El valor  1 representa el bit 1
ascii2float < signal.txt | sampler 128 > signal.float
spectrum_analyzer signal.float > spectrum.txt
draw_signal.sh spectrum.txt "Espectro de la señal 01"
```

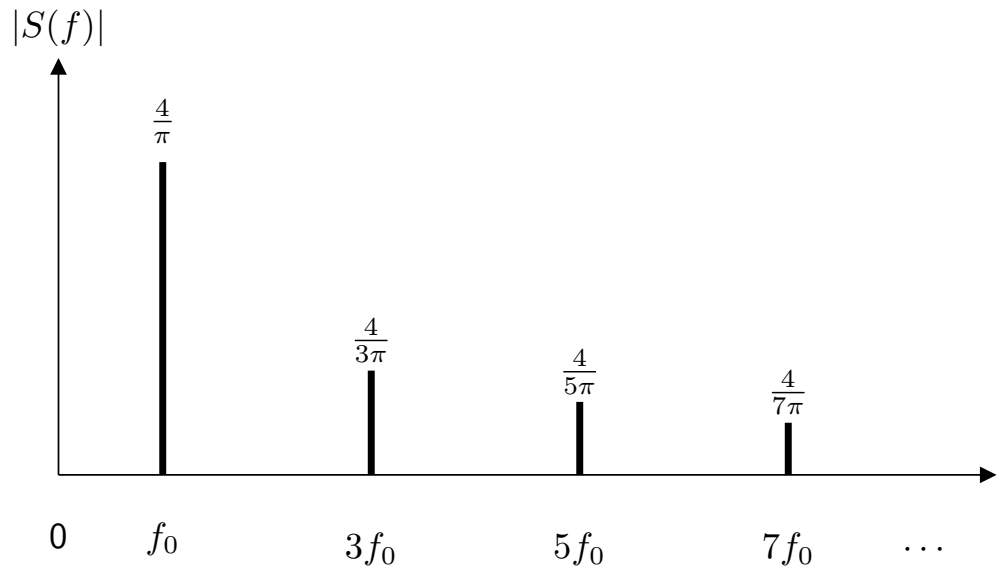


Figura 2.1: Espectro de frecuencias de una señal digital $s(t)$. $1/f_0$ es el tiempo que tarda en transmitirse la secuencia de bits 10 o 10 en la señal digital ...101010....

Como puede apreciarse en la ventana que ha debido aparecer, cada coeficiente de Fourier representa el punto de una polilínea. Modifique el script `draw_signal.sh` para que en lugar de unir los puntos del fichero de entrada usando líneas, se pinten barras verticales. Para saber cómo hacer esto exáctamente, ejecute el programa `gnuplot` y escriba en el intérprete `help plot`. Luego pulse la tecla <q>, y cuando pregunte `Subtopic of plot:` escriba `with`. Aparecerá un mensaje de ayuda con los argumentos que el comando `plot` acepta para pintar las gráficas. Luego use un editor de ficheros ASCII para modificar el script `draw_signal.sh`. Cuando haya conseguido mostrar una gráfica semejante a la que se muestra en la Figura 2.1 modifique de nuevo el script `draw_signal.sh` para dejarlo con su contenido original.

2.9. ¿Cómo afecta la atenuación de las señales a su espectro?

Como ya hemos indicado, los medios de transmisión tienden a atenuar más las altas frecuencias. Esto significa, modelando de una forma bastante pobre el proceso, que las componentes de alta frecuencia de las señales digitales van a ser eliminadas (van a ser cero). El espectro de una señal digital atenuada será semejante al que se presenta en la Figura 2.1, aunque a partir de una determinada banda de frecuencia $i \cdot f_0$, no poseerá energía. En otras palabras, si $s(t)$ es la señal original y $\hat{s}(t)$ es la señal filtrada (atenuada), se cumple que:

$$\hat{S}(f) = \begin{cases} S(f) & \text{si } f < i \cdot f_0 \\ 0 & \text{en caso contrario} \end{cases}$$

2.10. El filtrado de señales

Un filtro de señales es un sistema físico que modifica la potencia relativa de las distintas componentes de frecuencia de dichas señales [10]. En concreto, un filtro paso bajo es aquel que deja pasar las bajas frecuencias y elimina las altas, a partir de una determinada frecuencia f_c. En la Figura 2.2 podemos ver la *función de transferencia de un filtro paso bajo*[2]. A la frecuencia f_c se le conoce como *frecuencia de corte del filtro*.

El programa `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/low_pass_filter.c` (Apéndice H.6) permite filtrar una señal, dejando pasar la banda de frecuencias más bajas y eliminando el resto.

Taller 2.5: Como ejemplo, vamos a eliminar la mitad de las altas frecuencias de la señal "01" cuando la transmitimos a una tasa de bit (bit-rate) de 1 bit/segundo. Utilizaremos una precisión

[2]Que no es más que la respuesta del filtro paso bajo (la señal generada por el filtro paso bajo) representada en el dominio de la frecuencia cuando a la entrada colocamos una función impulso (una función que siempre vale 0 excepto durante un intervalo de tiempo muy pequeño).

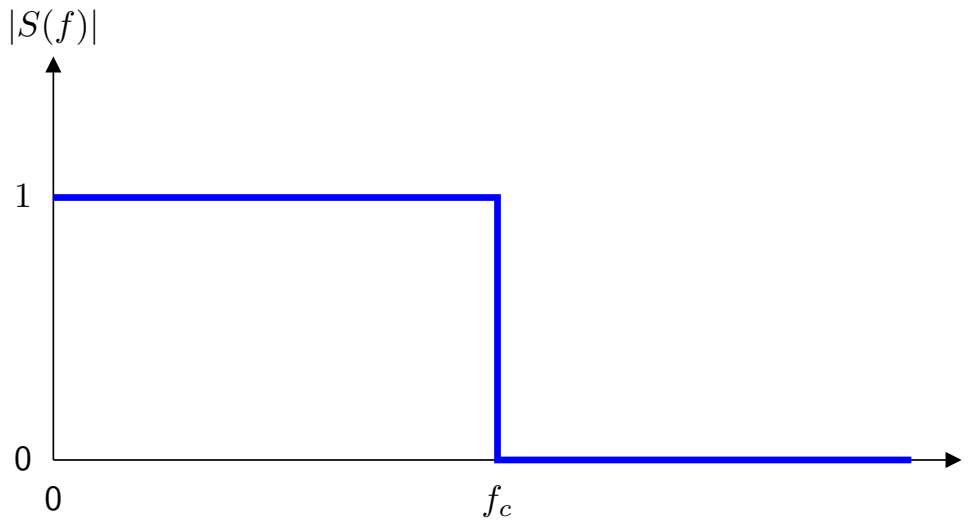


Figura 2.2: Función de transferencia de un filtro paso bajo.

de 128 muestras/bit en nuestro experimento. Para simplificar todo lo posible el proceso ha sido diseñado el script `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/filtrado_01.sh` cuyo contenido se muestra a continuación:

```
#!/bin/bash

#
# Filtrado de la señal 01
#

#
# Creamos la señal
#
rm -f 01_signal.txt # Borramos el fichero
echo "-1.0" >> 01_signal.txt # El valor -1 representa el bit 0
echo "1.0" >> 01_signal.txt # El valor  1 representa el bit 1
ascii2float < 01_signal.txt | sampler 128 > 01_signal.float

#
# Visualizamos su espectro
#
spectrum_analyzer 01_signal.float > 01_spectrum.txt
draw_signal.sh 01_spectrum.txt "Espectro de la señal 01"

#
# filtramos la señal
#
low_pass_filter 0.5 01_signal.float > 01_filtered.float

#
# Visualizamos la señal filtrada
#
float2ascii < 01_filtered.float > 01_filtered.txt
draw_signal.sh 01_filtered.txt "Señal 01 filtrada"

#
# Visualizamos el espectro de la señal filtrada
#
spectrum_analyzer 01_filtered.float > 01_filtered_spectrum.txt
draw_signal.sh 01_filtered_spectrum.txt "Espectro de la señal 01 filtrada"
```

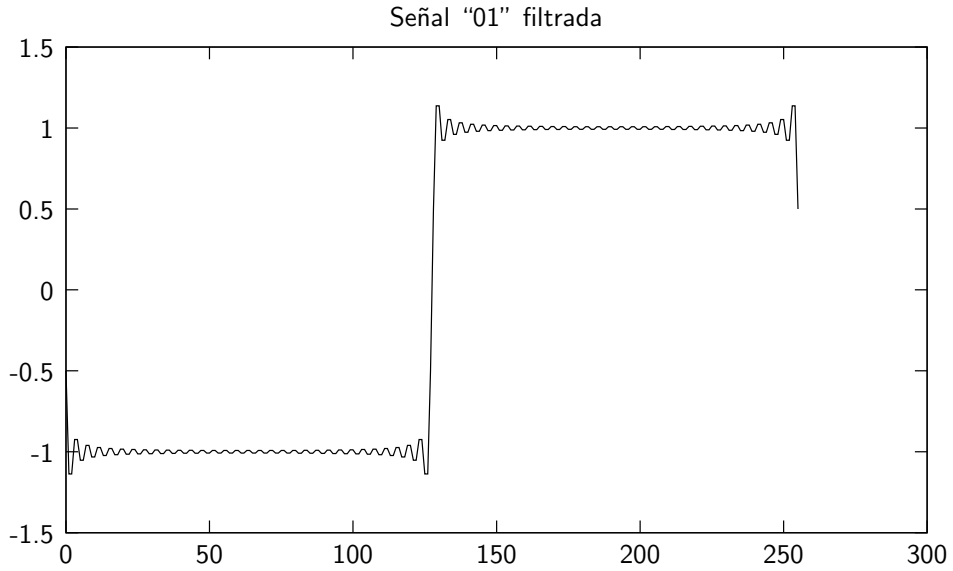


Figura 2.3: La señal "01" transmitida a razón de 1 bit/segundo cuando se han eliminado la mitad de las altas frecuencias.

Ejecute el script `filtrado_01.sh` escribiendo:

```
filtrado_01.sh
```

Si todo ha ido bien debería aparecerle una ventana con una curva semejante a la que se muestra en la Figura 2.3.

Cuestión 4: ¿Por qué el número de coeficientes de Fourier (número de componentes de frecuencia) es la mitad que el número de muestras introducidas?

Cuestión 5: ¿Qué controla el parámetro `128` que toma el programa `sampler` en el script que visualiza el espectro de la señal "01" desde el punto de vista del espectro de la señal? Es decir, ¿en qué varía el espectro de la señal cuando variamos dicho parámetro?

Cuestión 6: Repita el último experimento cuando se transmiten dos bits/segundo y eliminando todas las componentes de frecuencia por encima de 8 Hz (modifique el script `filtrado_01.sh`). ¿La señal recibida tiene ahora una mayor "calidad" (se parece más a la original)? ¿Por qué o por qué no? Explique las modificaciones que ha realizado en el script para llevar a cabo este experimento.

Práctica 3

Multiplexación de señales

3.1. Multiplexar, ¿para qué?

Existen situaciones donde un único medio de transmisión debe ser compartido por varios emisores. En algunos casos, como ocurre en nuestras casas cuando estamos hablamos por teléfono y a la vez descargamos datos de Internet, la multiplexación permite usar un único par trenzado metálico para ambas cosas al mismo tiempo, es decir, la multiplexación abarata costes al no ser necesario tener que instalar dos pares de hilo. En otros casos, como cuando vamos en el coche escuchando una emisora de radio, no hay más remedio de multiplexar las señales ya que el medio de transmisión (el aire) es único.

El dispositivo que permite multiplexar varias señales sobre un mismo medio de transmisión se llama *multiplexor* y el que realiza el proceso inverso, *desmultiplexor* (véase la Figura 3.1).

Cuestión 1: Indique otras situaciones donde se hace uso de la multiplexación de señales.

3.2. Multiplexación de señales en el dominio del tiempo

Dos (o más) señales pueden multiplexarse en el dominio del tiempo entrelazando sus muestras (siguiendo algún patrón predefinido) y transmitiéndolas como si de una única señal de se tratara. En la Figura 3.2 se muestra un ejemplo de *multiplexación en el tiempo* o TDM (Time Division Multiplexing) de dos señales en la que cada slot de tiempo transmite una muestra de cada señal. En este caso el patrón de entrelazamiento es enviar sólo una muestra de cada señal antes de enviar la siguente. Nótese que la tasa de datos de la señal resultante es el doble que la tasa de datos de cada una de las señales multiplexadas y que por lo tanto se necesita el doble de ancho de banda.

Cuestión 2: Imagine otros patrones de multiplexación de dos señales en el tiempo. Expóngalos.

3.3. La modulación de señales

La modulación de una señal $s(t)$ consiste en desplazar el espectro de la misma $S(f)$, conservando su forma, para generar una *señal modulada* [9]. Esto puede realizarse en el dominio del tiempo multiplicando la señal a modular $s(t)$ por una *señal moduladora* (o *portadora*) $w(t)$ para generar $s'(t)$ mediante

$$s'(t) = s(t)w(t),$$

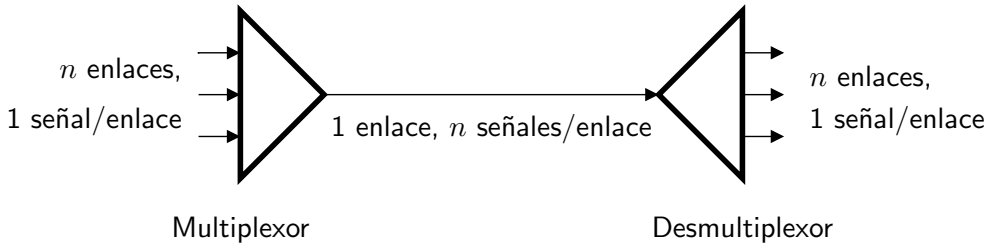


Figura 3.1: Multiplexación de un enlace.

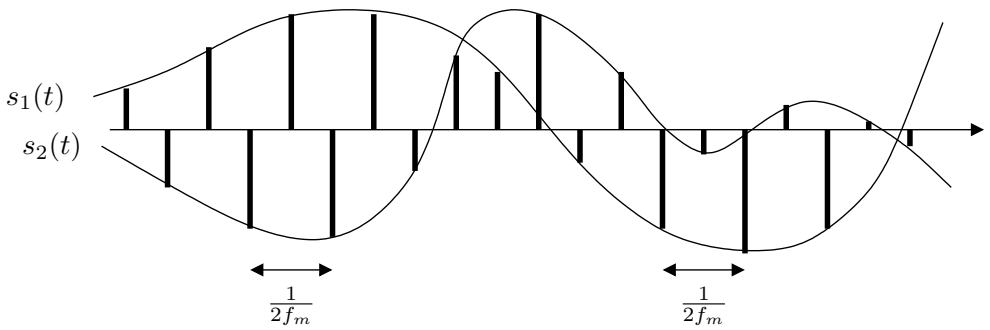


Figura 3.2: Multiplexación en el dominio del tiempo de dos señales. f_m es la máxima componente de frecuencia de cada una de ellas.

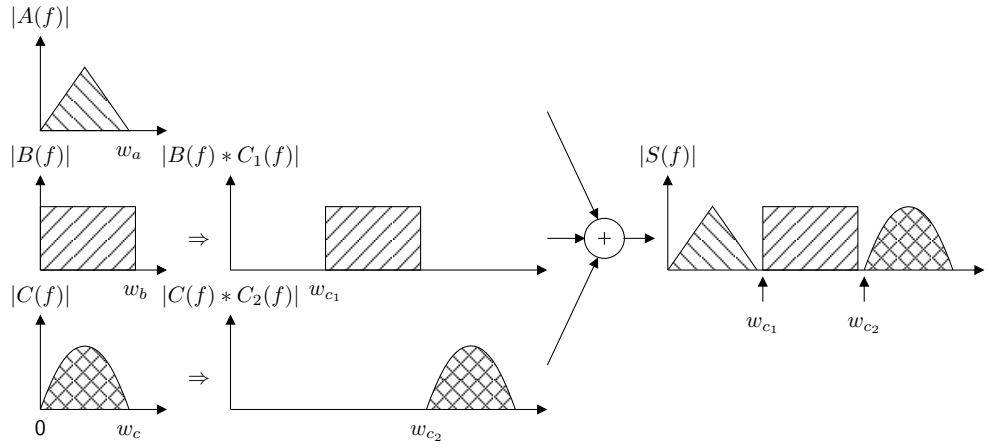


Figura 3.3: Multiplexación en el dominio de la frecuencia usando modulación en amplitud. $*$ denota a la convolución entre señales. $\Rightarrow$ a la modulación de señales.

o en el dominio de la frecuencia, convolucionando ambos espectros con

$$S'(f) = S(f) * W(f).$$

$w(t)$ es una señal sinusoidal pura (compuesta de una única componente de frecuencia w_0) y su espectro $W(f)$ es la *función Delta de Dirac*. Por tanto, la convolución de $S(f)$ con $W(f)$ consiste en desplazar $S(f)$ hasta w_0. El programa `modulator.c` (`http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/modulator.c`) (Apéndice H.7) realiza este proceso y el programa `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/demodulator.c` (Apéndice H.3) el inverso.

Cuestión 3: Revise la bibliografía recomendada (y otra que usted considere conveniente) y exponga una definición formal de la convolución entre funciones. ¿Qué diferencias hay entre la convolución de funciones y de señales?

3.4. Multiplexación de señales en el dominio de la frecuencia

La alternativa a TDM es la *multiplexación en el dominio de la frecuencia* o FDM (Frecuency Division Multiplexing). En este caso (como ocurre con las emisiones de radio) cada emisor transmite de forma continua en un rango de frecuencias diferentes. A cada una de estas bandas de frecuencias se les denomina *canal*. Este proceso se describe para tres señales en la Figura 3.3.

En dicha figura w_a, w_b y w_c son los anchos de banda de tres señales $a(t)$, $b(t)$ y $c(t)$, respectivamente. w_{c_1} y w_{c_2} son la frecuencia de dos señales portadoras $c_1(t)$ y $c_2(t)$ que modulan a las señales $b(t)$ y $c(t)$, respectivamente. Si se cumple que $w_{c_1} > w_a$ y que $w_{c_2} > w_a + w_b$ es posible construir una señal $s(t)$ como

$$s(t) = a(t) + b(t)c_1(t) + c(t)c_2(t),$$

o lo que es lo mismo, construir $S(f)$ como

$$S(f) = A(f) + B(f) * C_1(f) + C(f) * C_2(f),$$

de la que es posible, más tarde, extraer las señales multiplexadas $a(t)$, $b(t)$ y $c(t)$. Para ello hay que seleccionar el canal de frecuencias correspondiente y desmodular si es preciso.

Taller 3.1:

1. Ejecute el PC virual. El resto de pasos se realizarán en este PC.
2. Vamos a multiplexar y desmultiplexar en la frecuencia dos señales digitales. Lo primero que tenemos que tener en cuenta es que el espectro de las señales digitales no está limitado en banda (recuérdese la práctica anterior). Por tanto, tendremos que filtrar ambas señales antes de multiplexarlas. En este ejemplo supondremos que vamos a transmitir dos señales $a(t)$ y $b(t)$ que transportan 16 bits cada una a una tasa de 2 bits/segundo y que el ancho de banda disponible es de 16 Hz. Dedicaremos, por tanto, 8 Hz a cada canal. A continuación modularemos una de las señales ($b(t)$, por ejemplo) utilizando una portadora de 8 Hz y sumaremos las dos señales. Por tanto, la operación que estamos haciendo es

$$s(t) = a(t) + b(t)c(t),$$

donde $c(t)$ es la señal portadora. En el dominio de la frecuencia estamos realizando el cálculo

$$S(f) = A(f) + B(f) * C(f).$$

Para desmultiplexar cada señal hay que seleccionar el canal de frecuencias correspondiente. En el caso de $a(t)$ sólo hay que aplicar un filtro paso bajo que seleccione el canal $[0, 8[$ Hz. Con esto estamos eliminando $B(f) * C(f)$ y no es necesario desmodular. Para la recuperar señal $b(t)$ tendremos que eliminar la banda $[0, 8[$ Hz con lo que eliminamos $A(f)$ y a continuación desmodular para obtenemer $B(f)$.

El script `http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/multiplexacion-desmultiplexacion.sh`, cuyo contenido se muestra a continuación:

```
#!/bin/bash

#
# Multiplexación y desmultiplexación de dos señales digitales
#

#
# Creamos la señal a(t)
#
rm -f a_signal.txt # Borramos el fichero
echo "-1.0" >> a_signal.txt # 1. El valor -1 representa el bit 0
echo "1.0" >> a_signal.txt  # 2. El valor  1 representa el bit 1
echo "1.0" >> a_signal.txt  # 3.
echo "1.0" >> a_signal.txt  # 4.
echo "-1.0" >> a_signal.txt # 5.
echo "-1.0" >> a_signal.txt # 6.
echo "-1.0" >> a_signal.txt # 7.
echo "1.0" >> a_signal.txt  # 8.
echo "1.0" >> a_signal.txt  # 9.
echo "1.0" >> a_signal.txt  # 10.
echo "-1.0" >> a_signal.txt # 11.
echo "-1.0" >> a_signal.txt # 12.
echo "1.0" >> a_signal.txt  # 13.
echo "-1.0" >> a_signal.txt # 14.
echo "-1.0" >> a_signal.txt # 15.
echo "-1.0" >> a_signal.txt # 16
ascii2float < a_signal.txt | sampler 128 > a_signal.float

#
# Creamos la señal b(t)
#
rm -f  b_signal.txt # Borramos el fichero
```

```
echo "1.0" >> b_signal.txt  # 1. El valor -1 representa el bit 0
echo "-1.0" >> b_signal.txt # 2. El valor  1 representa el bit 1
echo "-1.0" >> b_signal.txt # 3.
echo "1.0" >> b_signal.txt  # 4.
echo "-1.0" >> b_signal.txt # 5.
echo "-1.0" >> b_signal.txt # 6.
echo "-1.0" >> b_signal.txt # 7.
echo "1.0" >> b_signal.txt  # 8.
echo "-1.0" >> b_signal.txt # 9.
echo "1.0" >> b_signal.txt  # 10.
echo "-1.0" >> b_signal.txt # 11.
echo "1.0" >> b_signal.txt  # 12.
echo "1.0" >> b_signal.txt  # 13.
echo "1.0" >> b_signal.txt  # 14.
echo "-1.0" >> b_signal.txt # 15.
echo "1.0" >> b_signal.txt  # 16.
ascii2float < b_signal.txt | sampler 128 > b_signal.float

#
# Filtramos a(t). Transmite a una tasa de 2 bps y hay 128 muestras/bit.
# Por tanto, la frecuencia de muestreo es 256 muestras/segundo.
# Esto significa que la máxima componente de frecuencia registrada para a(t)
# es de 128 Hz. Como vamos a filtrar a 8 Hz, cut-off_band = 8/128 = 0.0625.
#
low_pass_filter 0.0625 a_signal.float > a_filtered_signal.float

#
# Lo mismo hacemos para b(t).
#
low_pass_filter 0.0625 b_signal.float > b_filtered_signal.float

#
# Modulamos la señal b(t) usando una portadora c(t).
# b(t) tiene un total de 128*16=2048 muestras
# y 2048/2=1024 bandas de frecuencia. Como el ancho de banda registrado es de
# 128 Hz, cada banda representa un total de 128/1024 = 0.125 Hz. Si queremos
# modular b(t) hasta los 8 Hz, 8/0.125=64 bandas.
#
modulator 64 b_filtered_signal.float > b_modulated_signal.float

#
# Sumamos s(t) = a(t) + c(t)b(t).
#
add a_filtered_signal.float b_modulated_signal.float > s_signal.float

#
# Visualizamos a(t) filtrada, b(t) filtrada y s(t).
#
float2ascii < a_filtered_signal.float > a_filtered_signal.txt
draw_signal.sh a_filtered_signal.txt "a(t) filtrada"
float2ascii < b_filtered_signal.float > b_filtered_signal.txt
draw_signal.sh b_filtered_signal.txt "b(t) filtrada"
float2ascii < s_signal.float > s_signal.txt
draw_signal.sh s_signal.txt "s(t)"

#
# Visualizamos A(f) filtrada, B(f) filtrada y S(f).
#
./spectrum_analyzer a_filtered_signal.float > a_filtered_spectrum.txt
```

```
draw_signal.sh a_filtered_spectrum.txt "A(f) filtrada"
./spectrum_analyzer b_filtered_signal.float > b_filtered_spectrum.txt
draw_signal.sh b_filtered_spectrum.txt "B(f) filtrada"
./spectrum_analyzer s_signal.float > s_spectrum.txt
draw_signal.sh s_spectrum.txt "S(f)"

#
# Desmultiplexamos a(t). Eliminamos la banda de frecuencia
# ]8, 16] Hz.
#
low_pass_filter 0.0625 s_signal.float > a_desmultiplexed_signal.float

#
# Desmultiplexamos b(t). Filtramos y desmodulamos.
high_pass_filter 0.0625 s_signal.float > b_unmodulated_signal.float
demodulator 64 b_unmodulated_signal.float > b_desmultiplexed_signal.float

#
# Visualizamos a(t) desmultiplezada y b(t) desmultiplexada.
#
float2ascii < a_desmultiplexed_signal.float > a_desmultiplexed_signal.txt
draw_signal.sh a_desmultiplexed_signal.txt "a(t) desmultiplexada"
float2ascii < b_desmultiplexed_signal.float > b_desmultiplexed_signal.txt
draw_signal.sh b_desmultiplexed_signal.txt "b(t) desmultiplexada"
```

realiza el proceso que acabamos de describir. Podemos ejecutar dicho script escribiendo:

```
multiplexacion-desmultiplexacion.sh
```

Cuestión 4: Repita el experimento usando 32 Hz de ancho de banda por canal y presente el script diseñado. ¿Ha mejorado la calidad de las señales desmultiplexadas? ¿Por qué?

Cuestión 5: Los espectros de las señales son simétricos respecto de la banda de frecuencia 0. Esto significa que hasta ahora simplemente sólo hemos dibujado la mitad de los espectros (ignorando las frecuencias negativas). Si multiplicamos una señal digitalizada por una portadora cuya frecuencia es igual a la máxima componente de frecuencia de la señal a modular, ¿cómo sería el espectro de la señal modulada en comparación con el espectro de la señal original? Razone su respuesta.

Práctica 4

Configuración del IP

TCP/IP es el nombre que recibe el conjunto de protocolos que permiten conectarnos a Internet [5]. Entre estos protocolos el TCP (Transmission Control Protocol) que se encarga de controlar los errores de transmisión, y el IP (Internet Protocol) que controla el routing de los mismos, son los más utilizados.

Todos los sistemas operativos actuales, y entre ellos por supuesto Windows y Linux implementan el TCP/IP. Para que éste pueda hacer su trabajo, el IP antes debe ser adecuadamente configurado.

4.1. Nociones básicas sobre el TCP/IP

Para comprender el funcionamiento del IP es esencial conocer cómo se referencian las aplicaciones de red. En Internet hay millones de hosts y dentro de cada uno de ellos se ejecutan decenas de aplicaciones de red. Las direcciones IP y los puertos de red se utilizan para diferenciar a una aplicación de red de las demás que se están ejecutando en Internet.

4.1.1. Las direcciones IP

Las direcciones IP (en la versión 4 del IP) son números de 32 bits. Por definición, cada host público en Internet posee una dirección IP diferente.[1] La forma más frecuente de representar las direcciones IP es mediante la notación decimal punteada. Según esta notación los 4 bytes de una dirección IP se representan en decimal separados por puntos. Por ejemplo, si hacemos un ping a `www.google.es` veremos que tiene una dirección IP asignada:

```
# La opción "-c 1" se utiliza para enviar sólo una petición de eco.
usuario$ ping -c 1 www.google.es
PING www.l.google.com (209.85.129.104) 56(84) bytes of data.
64 bytes from fk-in-f104.google.com (209.85.129.104): icmp_seq=1 ttl=128 time=56.7 ms

--- www.l.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 56.774/56.774/56.774/0.000 ms
```

Taller 4.1: Ejecute el PC virtual. Desde él, haga un ping a un host conocido para comprobar que la red funciona correctamente.

Toda dirección IP consta de dos partes: (1) la dirección de la red a la cual pertenece y (2) la dirección del host que posee dicha dirección IP dentro de la red. Por ejemplo, todas las direcciones IP de los hosts de este aula son de la forma:

```
192.168.6.X
```

y por lo tanto comparten el mismo prefijo. El último byte, `X`, indica el host dentro de la red del aula. Nótese que el tamaño de las direcciones IP es fijo (siempre 32 bits). Por tanto, si dedicamos más bits a especificar el host dentro de la red, entonces el número de bits que indican la dirección de la red será menor, y viceversa.

Para conocer cuántos bits de la dirección IP especifican la dirección de la red y cuántos la dirección del host dentro de la red, se utiliza la máscara de red. Una máscara de red (en la versión 4 del IP) es un número

[1] Esto no es cierto para los hosts que se encuentran en redes privadas, es decir, los que están conectados a Internet a través de un NAT.

de 32 bits que comienza siempre por 1 y acaba en 0. Por ejemplo, todos los ordenadores del aula tienen la siguiente máscara de red:

`255.255.255.0`

Conociendo este valor, si la dirección IP de mi host es, por ejemplo:

`192.168.6.100`

entonces, la dirección IP de la red del aula es:[2]

`192.168.6.0`

y la dirección de mi host dentro de la red es:

`100`

El número de bits a 0 que hay en la máscara de red determina el tamaño de la red (el número máximo de adaptadores de red que pueden estar conectados a la red). En nuestro caso hay 8 ceros. Por tanto, el tamaño de la red del laboratorio es $2^8 = 256$ direcciones IP.

Cuestión 1: ¿Cuál sería la máscara de red para una red que está conectada a Internet y en la que sólo hay dos hosts? Razone su respuesta.

4.1.2. La puerta de enlace

Todas las redes conectadas a otras lo hacen a través de, al menos, un dispositivo de conmutación especial llamado *router*. Este nodo recibe comúnmente los nombres de *puerta de enlace* y *gateway*. Generalmente la dirección IP del gateway es la primera del rango útil de direcciones de la red a la que estamos conectados. Por ejemplo, la dirección IP de la red del aula es:

`193.147.118.0`

Esta dirección sólo puede utilizarse para designar a la red, no puede ser utilizada por ningún interface de red. La siguiente dirección IP del rango disponible es:

`193.147.118.1`

Esta debería ser la dirección IP del interface de red del gateway que conecta nuestra red con el resto de Internet. Finalmente, indicar que puesto que la dirección IP del interfaz dentro de la red tiene un tamaño de 8 bits, la última dirección IP del rango de direcciones asociadas a nuestra red es:

`193.147.118.255`

Esta dirección IP tampoco puede ser asignada a ningún interfaz porque es la dirección de broadcast de la red local. Cualquier mensaje enviado a esta dirección llegará a todos los interfaces de la red local.

4.1.3. El servidor de nombres

Las direcciones IP raramente se utilizan cuando estamos navegando por Internet porque recordar direcciones IP es complicado. Es mucho más fácil recordar `www.google.es` que `142.234.211.117`, por ejemplo.

El servicio que permite asignar nombres a los hosts en Internet se conoce como servicio de nombres de dominio o DNS (Domain Name Service). Cuando configuramos el TCP/IP generalmente también se especifica la dirección IP de uno o más servidores DNS. Así, evitaremos tener que introducir las direcciones IP de cada uno de los hosts a los que fuéramos a acceder.

Taller 4.2: Compruebe que el DNS del PC virtual funciona correctamente. Hacer un ping a un host al que no se haya conectado previamente en esta sesión es suficiente.

Cuestión 2: En el caso del PC virtual, ¿pertenece el servidor de nombres de dominio que está utilizando a su red local? Razone su respuesta. Nota, el servidor DNS se define en el fichero `/etc/resolv.conf`.

[2] Recuérdese que todas las direcciones IP en IPv4 son de 32 bits, independientemente de si referencian a un host o a una red de hosts.

4.2. Configuración del TCP/IP en Windows XP

Todos los hosts del laboratorio están ya configurados para conectarnos a Internet. Por tanto, lo que vamos a indicar son los pasos que habría que ir realizando para configurar el TCP/IP, aunque en realidad no sea necesario llevarlos a cabo.

Lo primero que debemos saber es que la modificación de la configuración del TCP/IP podría afectar a otros usarios que están utilizando la computadora. Como Microsoft Windows XP es un sistema operativo multiusuario (varios usuarios pueden estar utilizando la computadora al mismo tiempo), sólo un usuario con los permisos de administrador podría modificar los parámetros de configuración del TCP/IP.

Bien. Bajo la suposición de que hemos accedido al ordenador con permisos de administración, ejecutaremos los siguientes pasos (supondremos instalada una versión en español):

1. Acceder a *Panel de control*.
2. Acceder a *Conexiones de red*.
3. Seleccionar el interface de red titulado *Conexiones de área local*. Si estuviera deshabilitado, pinchar en el icono con el botón derecho del ratón y habilitarlo. Pinchar de nuevo en el icono usando el botón derecho de ratón y elegir la entrada *Propiedades*.
4. Se abrirá una nueva ventana titulada *Propiedades de Conexión de área local*. En la ventana interna titulada *Esta conexión utiliza los siguientes elementos:* buscar la entrada *Protocolo Internet (TCP/IP)*. Esta entrada debe estar activada (en la parte izquierda debe aparecer una especie de `V`) y además seleccionada con el ratón (debe aparecer en vídeo inverso, con el fondo el azul y las letras en blanco). Cuando todas estas premisas se cumplan, pulsar en *Propiedades*.
5. Aparecerá una nueva ventana titulada *Propiedades de Protocolo Internet (TCP/IP)* que dentro tiene una única pestaña titulada *General*. En esta pestaña aparecen los campos que permiten configurar el TCP/IP.
   - Entre *Obtener una dirección IP automáticamente* y *Usar la siguiente dirección IP* eligiremos esta última opción, porque los hosts del laboratorio (actualmente) tienen una IP fija. En ese instante se habilitarán las zonas donde escribiremos la dirección IP la máscara de la red y la puerta de enlace. Toda esta información la deberíamos encontrar en una etiqueta que hay en el frontal de nuestro ordenador.
   - Entre *Obtener la dirección del servidor DNS automáticamente* y *Usar las siguientes direcciones de servidor DNS* seleccionaremos esta última opción, porque el DNS es estático y conocido para los hosts del laboratorio (en concreto, `filabres.ual.es`). Se habilitarán dos zonas donde escribir la dirección IP de dos servidores DNS. Uno debería ser suficiente. El segundo sólo se utiliza cuando el primero no funciona. La dirección IP del servidor DNS principal debería aparecer también en el frontal del ordenador.

Cuestión 3: Averigue las diferencias, a nivel de pasos que realiza el sistema operativo, que existen entre configurar el TCP/IP a mano y hacerlo mediante el DHCP.

4.3. Configuración del TCP/IP en Linux

Todos los hosts deberían tener configurado el TCP/IP. Por tanto, vamos a explicar qué pasos serían los que daríamos en el caso de que esto no fuera así.

En primer lugar, saber que sólo el súper-usuario (`root`) puede modificar los parámetros del TCP/IP. Sin embargo, como en realidad no necesitamos modificar ningún fichero, todo el proceso que a continuación se expone debería poderse realizar como usuario sin privilegios de administración.

4.3.1. ¿Reconoce el kernel el hardware de red?

Para conocer si el núcleo (kernel) del sistema operativo reconoce el adaptador de red que estamos utilizando podemos realizar diferentes pruebas. La primera consiste en ver si el TCP/IP ya está configurado, con lo que sabremos que el kernel necesariamente lo reconoce. Esto se puede hacer escribiendo en un `shell`:

```
root# ifconfig
```

La salida es una lista de interfaces, junto con algunos parámetros de configuración. Lo importante es que veamos que aparece el interface `eth0`, esto es, el primer interface de red Ethernet.

Si esto no fuera así, hay dos opciones:

Cargar el módulo correspondiente: Conociendo el modelo del adaptador de red tenemos que averiguar cómo se llama el módulo que lo manipula.[3] Lo localizamos en el sistema de ficheros y lo cargamos usando `insmod` o `modprobe`. Los detalles de este proceso deberían ser consultados en un kernel-HOWTO o en un manual de referencia de la distribución de Linux que estamos utilizando.

Compilar un nuevo kernel y usarlo: Para hacer esto necesitamos los fuentes de un kernel reciente (preferiblemente la última disponible). Activaremos el módulo correspondiente y compilaremos el kernel. Después lo copiaremos en un directorio accesible por el *boot loader* y reiniciaremos la máquina. Los detalles de este proceso deberían encontrarse también en la documentación anteriormente comentada.

4.3.2. Configuración temporal del TCP/IP

Como se ha indicado en la sección anterior, puede ocurrir que el kernel sí que reconozca el hardware de red, pero éste no ha sido activado. La forma estándar de configurar temporalmente un interfaz de red consiste en usar las utilidades `/sbin/ifconfig` y `/sbin/route` desde la línea de comandos. Ejemplo:

```
# Establecemos la dir IP
root# ifconfig eth0 192.168.213.128

# Establecemos la máscara de red
root# ifconfig eth0 netmask 255.255.255.0

# Levantamos el interfaz. Sustituir "up" por "down" para desactivarlo.
root# ifconfig eth0 up

# Establecemos el gateway (router de nuestra red)
root# route --inet add -net 0.0.0.0/0 gw 192.168.213.2 eth0

# Definimos el servidor de nombres
root# echo "nameserver 150.214.156.2" > /etc/resolv.conf
```

Taller 4.3: Determine el rango de direcciones IP de la subred en la que está el PC virtual (utilice `ifconfig` sin argumentos) y cambie la dirección IP del PC virtual. Compruebe que la conexión con Internet sigue funcionando.

Cuestión 4: Indique la dirección IP que ha utilizado. ¿A qué sub-red pertenece dicha dirección IP? ¿Es una dirección privada? Razone su respuesta.

Taller 4.4: Cambie el servidor de nombres del PC virtual por el de la Universidad de Almería (`150.214.156.2`).

Cuestión 5: ¿Sigue funcionando el DNS? Intente contestar a esta pregunta en dos situaciones distintas: (1) cuando está en la universidad y (2) cuando fuera de la universidad.

4.3.3. Configuración permanente del TCP/IP

La configuración permanente se consigue haciendo que los scripts de arranque de la computadora ejecuten en algún instante los comandos especificados durante la configuración temporal. El problema al que nos enfrentamos ahora es localizar los ficheros de configuración que estos scripts utilizan para realizar dicha tarea. Por desgracia, estos scrips suelen tener diferentes localizaciones y contenidos, dependiendo de la distribución de Linux utilizada. A continuación se explicará cómo localizar dichos ficheros.[4]

Bien. Antes de comenzar, un poco de nomenclatura; en adelante hablaremos de los programas ejecutables como *procesos*. En Unix (y por tanto en Linux) los procesos son tareas que pueden invocar a otros procesos y

[3] Lo más sencillo es buscar en Internet.

[4] Aunque esta sección ha sido particularizada para Debian Linux, el procedimiento de localización de los ficheros de configuración del TCP/IP puede ser fácilmente adaptado a otras distribuciones de Linux.

que el sistema operativo ejecuta durante un cierto intervalo de tiempo. Cuando los procesos resuelven tareas útiles para el sistema operativo de forma indefinida, se habla de *demonios* (*daemons* en inglés).

Para encontrar el o los ficheros que contienen la configuración del TCP/IP en una máquina con Linux los pasos a realizar deberían ser los siguientes:

1. Localizar el fichero `/etc/inittab`. Este fichero es utilizado por el proceso `/sbin/init` que se encarga de configurar la computadora y de cargar el resto de procesos después de que el sistema operativo arranque y monte el sistema de ficheros raíz [3]. De hecho, `/sbin/init` es el proceso padre de todos los demás procesos que corren en su máquina. Esto último puede comprobarse escribiendo:

   ```
   usuario$ ps ax -H
   ```

   Observe el tabulado dentro de la columna "COMMAND".

2. En Debian, el fichero `/etc/inittab` debería tener una entrada semejante a:

   ```
   si::sysinit:/etc/init.d/rcS
   ```

   El primer campo[5] (`si`) es un identificativo del *runlevel* (nivel de ejecución) en el que se lanza cada proceso y significa *system initialization*.[6] El segundo campo es una lista de niveles de ejecución en los que se va a ejecutar el proceso especificado por el cuarto campo (`/etc/init.d/rcS`) y debe estar vacío si el tercer campo, que especifica la acción a llevar a cabo en ese runlevel vale `sysinit`. El script `/etc/init.d/rcS` se encarga de levantar el sistema y sería el equivalente al fichero `AUTOEXEC.BAT` en MS-DOS.

   Tras la inicialización especificada en `/etc/init.d/rcS`, `/sbin/init` lleva la computadora al estado normal de funcionamiento. Para ello debe lanzar un montón de demonios que hacen cosas como leer el teclado y el ratón, mandar los trabajos de impresión a las impresoras, recibir los correos electrónicos, etc. Todos estos procesos son especificados en las entradas:

   ```
   l0:0:wait:/etc/init.d/rc 0
   l1:1:wait:/etc/init.d/rc 1
   l2:2:wait:/etc/init.d/rc 2
   l3:3:wait:/etc/init.d/rc 3
   l4:4:wait:/etc/init.d/rc 4
   l5:5:wait:/etc/init.d/rc 5
   l6:6:wait:/etc/init.d/rc 6
   ```

   Escribir

   ```
   usuario$ man inittab
   ```

   para conocer la sintaxis exacta de `/etc/inittab`.

   Como podemos apreciar, estas entradas tienen un elemento extra que es un parámetro al script `/etc/init.d/rc`. Cuando ejecutamos:

   ```
   /etc/init.d/rc 0
   ```

   Todos los procesos que en el directorio `/etc/rc0.d` comiencen por una `S` van a ser lanzados en el runlevel 0, en orden alfabético. Los procesos que comienzan por `K` son killed (matados) cuando abandonamos ese runlevel. Como podemos ver, todos estos procesos son enlaces simbólicos a otros procesos localizados en el directorio:

   ```
   /etc/init.d/
   ```

3. En `/etc/rc0.d` encontramos que el proceso `S??networking` (donde `??` son dos dígitos decimales) apunta al proceso:

[5] Cada campo está delimitado por el carácter ":".

[6] El runlevel define el estado en el que se encuentra computadora. Por ejemplo, si vale 0 es que la computadora está apagada o se está apagando, si vale 1 es que está en modo single-user. Los runlevels que van desde el 2 hasta el 5 se dedican para ejecutar procesos en modo multi-user y el runlevel 6 para indicar que la computadora se está reiniciando.

```
/etc/init.d/networking
```

que es el encargado de configurar el TCP/IP. Este proceso debe invocarse escribiendo:

```
root# /etc/init.d/networking start
```

para configurar por primera vez la red[7]. Si lo editamos y buscamos en la sección `start` veremos que el proceso que configura el interface de red se llama `/sbin/ifup`[8].

4. `/sbin/ifup` tiene una documentación asociada que puede consultarse escribiendo:

   ```
   usuario$ man ifup
   ```

   Revisando dicha documentación descubriremos que `ifup` y `ifdown` (el programa que se utiliza para deshabilitar los interfaces de red) leen los datos para configurar el TCP/IP del fichero:

   ```
   /etc/network/interfaces
   ```

Bien. Si hemos realizado correctamente el proceso de rastreo anterior llegaremos a que los ficheros de configuración del TCP/IP son, dependiendo de la distribución de Linux:

Debian Linux:

Como acabamos de ver, el fichero que contiene los parámetros son:

```
/etc/network/interfaces
```

Su sintaxis es:

```
root# cat /etc/network/interfaces
### etherconf DEBCONF AREA. DO NOT EDIT THIS AREA OR INSERT TEXT BEFORE IT.
auto lo eth0

iface lo inet loopback

iface eth0 inet static
  address 193.147.118.57
  netmask 255.255.255.0
  gateway 193.147.118.1

### END OF DEBCONF AREA.  PLACE YOUR EDITS BELOW; THEY WILL BE PRESERVED.
```

En él podemos leer que este fichero ha sido creado de forma automática por la utilidad `debconf` cuando instaló el paquete `etherconf`. Por tanto, si quisiéramos modificar este fichero deberíamos escribir:

```
root# dpkg-reconfigure etherconf
```

Para aprender más acerca de la utilidad `dpkg` acceder a la Guía de Referencia Debian (`http://www.debian.org/doc/user-manuals#quick-reference`).

Fedora Core Linux:

El fichero que configura el interfaz `eth0` es `/etc/sysconfig/network-scripts/ifcfg-eth0` cuya sintaxis es:

```
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
NETMASK=255.255.255.0
```

[7] Por primera vez se refiere a que es la primera vez desde que la computadora ha arrancado.
[8] Ejecútese `which ifup`.

```
IPADDR=193.147.118.57
GATEWAY=193.147.118.1
TYPE=Ethernet
USERCTL=no
IPV6INIT=no
PEERDNS=no
```

Este fichero sólo debería modificarse utilizando un programa que lo edite. Por ejemplo, en X-window podemos utilizar `system-config-network`.

Gentoo Linux:

El fichero que configura a todos los interfaces de red es:

```
/etc/conf.d/net
```

Su sintaxis es:

```
root# cat /etc/conf.d/net
# This blank configuration will automatically use DHCP for any net.*
# scripts in /etc/init.d.  To create a more complete configuration,
# please review /etc/conf.d/net.example and save your configuration
# in /etc/conf.d/net (this file :]!).

config_eth0=( "193.147.118.57 netmask 255.255.255.0 brd 193.147.118.255" )
routes_eth0=( "default gw 193.147.118.1" )
```

En Gentoo este fichero se modifica usando un editor de ficheros ASCII.

4.3.4. Usando las modificaciones permanentes

La modificación de los ficheros de configuración del TCP/IP no basta para que probemos los cambios. Hay que inicializar los scripts que utilizan dichos ficheros. Esto se hace así:

Debian Linux:

```
root# /etc/init.d/networking restart
```

Fedora Core Linux:

```
root# /etc/init.d/network restart
```

Gentoo Linux:

```
root# /etc/init.d/net.eth0 restart
```

4.3.5. Configuración del nombre y el dominio del host

Debian Linux:

El nombre y dominio del host se encuentran en el fichero `/etc/hostname`:

```
gogh.ace.ual.es
```

Fedora Core Linux:

El nombre y dominio del host se encuentran en el fichero `/etc/sysconfig/network`:

```
NETWORKING=yes
NETWORKING_IPV6=yes
HOSTNAME=gogh.ace.ual.es
```

Gentoo Linux:

El nombre del host aparece en el fichero `/etc/conf.d/hostname`:

```
root# cat /etc/conf.d/hostname
# /etc/conf.d/hostname

# Set to the hostname of this machine
HOSTNAME="gogh"
```

El dominio del host aparece en el fichero `/etc/conf.d/domainname`:

```
# /etc/conf.d/domainname

# When setting up resolv.conf, what should take precedence?
#  0 = let dhcp/whatever override DNSDOMAIN
#  1 = override dhcp/whatever with DNSDOMAIN

OVERRIDE=1

# To have a proper FQDN, you need to setup /etc/hosts and /etc/resolv.conf
# (domain entry in /etc/resolv.conf and FQDN in /etc/hosts).
#
# DNSDOMAIN merely sets the domain entry in /etc/resolv.conf, see
# the resolv.conf(5) manpage for more info.

DNSDOMAIN="ace.ual.es"

# For information on setting up NIS, please see:
#   http://www.linux-nis.org/nis-howto/HOWTO/

NISDOMAIN=""
```

4.3.6. Configuración del DNS

Aunque parezca mentira, la configuración del servidor de nombres es independiente de la distribución de Linux. El fichero que contiene la dirección IP del servidor DNS es siempre `/etc/resolv.conf` que, para todos los hosts de la Universidad de Almería, debería tener al menos la entrada[9]:

```
nameserver 150.214.156.2
```

Si existen servidores alternativos, aparecen más entradas pero con otra dirección IP claro.

Otro aspecto importante a considerar es el papel del fichero `/etc/hosts`. Este fichero se utiliza siempre (aunque el servidor DNS haya sido especificado,) y contiene una lista de resoluciones estáticas de nombres. En este fichero se incluyen las resoluciones más frecuentes (por ejemplo, para hosts a los que nos conectamos todos los días) y descargamos así al DNS. Por otra parte, si este servicio cae, la resolución para estos hosts sigue funcionando. El contenido típico de este fichero es:

```
root# cat /etc/hosts
# /etc/hosts
#
# This file describes a number of hostname-to-address
# mappings for the TCP/IP subsystem.  It is mostly
# used at boot time, when no name servers are running.
# On small systems, this file can be used instead of a
# "named" name server.  Just add the names, addresses
# and any aliases to this file...
#
```

[9] A no ser que el host esté dentro de una red privada, en cuyo paso es probable que tenga la dirrección IP de otro host de la sub-red que funiona como servidor DNS.

```
127.0.0.1             localhost
193.147.118.57        gogh.ace.ual.es   gogh
150.214.156.2         filabres.ual.es   filabres alias_filabres

# IPV6 versions of localhost and co
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Taller 4.5: Usando el fichero `/etc/hosts`, defina un alias para `filabres.ual.es`. Pruebe a hacer un `ping` a filabres usando su alias.

Cuestión 6: ¿Podría navegar por Internet si no ha especificado correctamente el servidor de nombres? Razone su respuesta.

Cuestión 7: ¿Qué ventajas tiene el usar un servidor de nombres local frente a usar un servidor de nombres lejano?

Cuestión 8: ¿Qué fichero contiene la dirección IP del gateway? ¿Y la máscara de red?

Cuestión 9: Desde el punto de vista de la aplicación cliente, ¿qué diferencia existe entre usar el nombre canónico o usar el alias de un determinado servidor?

Cuestión 10: Cuando realiza un ping a su servidor de nombres usando su nombre canónico, ¿se está usando su servidor DNS para resolver la dirección IP de su servidor DNS?

Práctica 5

Acceso remoto 1: Telnet y Ftp

Las primeras computadoras eran extremadamente caras y debían ser compartidas por muchos usuarios a la vez para justificar su construcción. En esta situación cada usuario usaba un terminal "tonto" (con poca potencia de cálculo) para conectarse al host principal, "el mainframe". Para que los usuarios pudieran usar los recursos del host principal se escribieron una serie de aplicaciones de red que perduran hasta nuestros días. De estos programas, los más interesantes son Telnet y Ftp.

5.1. Telnet

Telnet es una aplicación cliente-servidor que se utiliza para el acceso remoto a un host. El servidor se ejecuta en el host remoto y el cliente en el local. De esta forma podemos acceder a los recursos de la computadora remota a través de un shell, igual que si estuviéramos sentados frente a un teclado directamente conectado al host remoto.

5.1.1. La interacción cliente-servidor

Telnet utiliza el protocolo TCP para transmitir, carácter a carácter, hasta el host remoto cada una de las teclas que pulsamos en el teclado del host cliente. De hecho, Telnet es (aparte de una conexión con un shell en el host remoto) una aplicación básica de *eco* (*echo* en inglés). Cada vez que pulsamos una tecla el cliente la transmite hasta el servidor y éste nos la devuelve. Así el usuario comprueba si ha existido algún error.

Como hemos dicho, para llevar a cabo estas comunicaciones Telnet utiliza el TCP. Esto significa que todas las comunicaciones están virtualmente libres de errores de transmisión gracias al sistema de retransmisión automática de paquetes que el TCP implementa.[1]

5.1.2. Utilizando Telnet

Para usar Telnet simplemente invocamos al cliente desde la línea de comandos y escibimos a continuación el nombre del host al que queremos conectarnos:

```
usuario$ telnet <host_remoto>
# Nota: los <> no pertenecen a la dirección del host
```

Además, Telnet puede ser utilizado también para acceder a un servidor distinto de `telnetd`. Por ejemplo, podemos utilizar un cliente Telnet cuando deseemos comprobar el funcionamiento de un servidor SMTP que escucha en el puerto 25. Para seleccionar dicho servicio invocaremos a (el cliente) `telnet` de la siguiente forma:

```
usuario$ telnet <servidor_SMTP> 25
```

5.1.3. Acerca de la seguridad en las comunicaciones

Un aspecto importante a tener en cuenta a la hora de utilizar Telnet es que todas las transmisiones circulan por la red en texto plano, sin ningún tipo de cifrado. Y ésto es cierto incluso para la transmisión del nombre de usuario y del password cuando accedemos a la computadora remota. Por este motivo, Telnet sólo debería ser utilizado en aquellos contextos donde se está seguro que este hecho no va a ser un problema.

[1] Por tanto, si vemos que existe un error en algún comando que hayamos escrito, muy probablemente se trate de un error de mecanografía.

En la práctica existen pocas situaciones donde el uso de Telnet sea seguro: (1) cuando accedemos localmente a nuestro host y (2) cuando usemos un canal cifrado, usando por ejemplo, SSH.

Taller 5.1: Usando el programa Telnet que viene con el sistema operativo Windows intente hacer Telnet a un servidor que conozca. El programa Telnet puede ser invocado desde el intérprete de comandos del MS-DOS a través del comando `telnet`.

5.1.4. Instalación del cliente

Debian Linux: Instalamos el paquete Linux NetKit (`http://www.hcs.harvard.edu/~{}dholland/computers/netkit.html`).

```
root# apt-get install telnet
```

Fedora Core Linux: Instalamos el paquete Linux NetKit.

```
root# yum install telnet
```

Gentoo Linux: Instalamos el paquete OpenBSD (`ftp://ftp.suse.com/pub/people/kukuk/ipv6/`).

```
root# emerge telnet-bsd
```

Taller 5.2: Instale un cliente Telnet. Intente acceder a un host que conoza usando el servicio Telnet desde el PC virtual.

5.1.5. Instalación del servidor

Debian Linux:

```
root# apt-get install telnetd
```

Fedora Core Linux:

```
root# yum install telnet-server
```

Gentoo Linux:

```
root# emerge telnet-bsd xinetd
```

5.1.6. Configuración del servidor

Debian Linux: No hay que hacer nada.

Fedora Core Linux: Hay que editar el fichero `/etc/xinetd.d/telnetd` y comprobar que el servidor de Telnet no está "disabled". A continuación reiniciar el servicio `xinetd` (véase el Apéndice C).

Gentoo Linux: Hay que editar el fichero `/etc/xinetd.d/telnetd` y comprobar que el servidor de Telnet no está "disabled". A continuación reiniciar el servicio `xinetd` (véase el Apéndice C).

Taller 5.3: Instale un servidor Telnet. Intente acceder al host virtual. No utilice el usuario `root` porque en muchos casos, por motivos de seguridad, el demonio Telnet está configurado para negar el acceso al administrador. Intente el acceso desde:

1. El PC virtual (Linux).
2. El PC huésped (Windows). Aunque el PC virtual está dentro de una red privada (una red 192.168.0.0), el NAT del VMware redirige todas las conexiones entrantes hacia el PC virtual correspondiente, utilizando su dirección IP privada. Ejecute `ifconfig` dentro del PC virtual para averiguar su dirección IP privada y utilícela desde Windows para hacer el Telnet.

5.1.7. Activación y desactivación del servicio

Cuando deseamos que el demonio quede activado o desactivado de forma definitiva (aunque apaguemos y encendamos la máquina) hay que modificar los correspondientes scripts de arranque.

El demonio Telnet es un sub-demonio y por tanto, su activación y desactivación no es trivial. Esto es exáctamente lo que hay que hacer:

Debian Linux:
```
# Activación definitiva
root# update-inetd --enable telnet

# Desactivación definitiva
root# update-inetd --disable telnet
```

Fedora Core Linux: El demonio que controla Telnet se llama `xinetd`. Para activar y desactivar Telnet hay que modificar el fichero de configuración y reiniciar el demonio.

Gentoo Linux: El demonio que controla Telnet se llama `xinetd`. Para activar y desactivar Telnet hay que modificar el fichero de configuración y reiniciar el demonio.

Taller 5.4: Compruebe que activando y desactivando el servidor Telnet, el servicio se reanuda o deja de prestarse.

5.2. Ftp (File Transfer Program)

Ftp es, junto con Telnet, uno de esos programas que aparecieron en las primeras redes de computadoras. Si Telnet permite acceder de forma remota a un host, Ftp se utiliza para transmitir ficheros entre hosts remotos.

Ftp es un sistema que consta de un servidor y de un cliente. El usuario ejecuta el cliente de la forma:

```
usuario$ ftp <host_remoto>
```

El servidor Ftp solicita un login y un password correctos y cuando dicha información ha sido enviada (sin cifrar), el cliente ejecuta un intérprete sencillo de órdenes que permiten mover ficheros entre nuestro host y el remoto.

5.2.1. Instalación del cliente

Existen muchos clientes Ftp. El que nosotros vamos a instalar se llama `ftp` y pertenece al paquete Linux NetKit (`http://www.hcs.harvard.edu/~{}dholland/computers/netkit.html`):

Debian Linux:
```
root# apt-get install ftp
```

Fedora Core Linux:
```
root# yum install ftp
```

Gentoo Linux:
```
root# emerge ftp
```

Taller 5.5: Instale un cliente Ftp.

5.2.2. Comandos Ftp más usuales

A continuación presentamos una lista de los principales comandos que acepta el cliente:

`pwd`**:** Imprimir el directorio actual.

`ls`**:** Mostrar el contenido del directorio remoto.

`cd <directorio>`**:** Nos permite cambiar de directorio.

`get <fichero>`**:** Transfiere un fichero desde el host remoto al host local.

`put <fichero>`**:** Transfiere un fichero desde el host local al host remoto.

`!<comando local>`**:** Ejecuta un comando en un shell del cliente.

`quit`**:** Cierra el intérprete del Ftp.

Si se desea conocer el resto de comandos que el cliente Ftp acepta, escriba `help` en el shell del cliente Ftp.

> **Taller 5.6:** Conéctese a un servidor Ftp que conozca desde el PC virtual y transfiera algún archivo. Haga lo mismo desde el host huésped (Windows).

5.2.3. Instalación del servidor

También existen múltiples servidores Ftp. El que nosotros vamos a instalar se llama vsftpd (`http://vsftpd.beasts.org/`):

Debian Linux:

```
root# apt-get install vsftpd
```

Fedora Core Linux:

```
root# yum install vsftpd
```

Gentoo Linux:

```
root# emerge vsftpd
```

5.2.4. Configuración del servidor

En el `vsftpd`, por defecto, sólo tiene acceso el usuario `anonymous`. Para habilitar el acceso a todos los usuarios del sistema es necesario descomentar la línea `local_enable=YES`. Además, si queremos habilitar la escritura en los home's de los usuarios, hay que descomentar la línea `write_enable=YES`. No olvide reiniciar el servicio si ha modificado algún parámetro en el fichero de configuración (véase el Apéndice C).

Debian Linux: El fichero de configuración es `/etc/vsftpd.conf`. Excepto lo indicado anteriormente no hay que realizar nada extra.

```
# Editar el fichero y salvarlo
root# vi /etc/vsftpd.conf
```

Fedora Core Linux: Si se desea que los usuarios tengan acceso a su home's en la máquina remota hay deshabilitar selinux modificando el fichero `/etc/selinux/config`. Esto puede hacerse con el comando:

```
root# /usr/sbin/setsebool -P ftp_home_dir 1
```

Gentoo Linux: Es necesario crear un fichero de configuración. Por suerte, la que viene en el fichero `/etc/vsftpd/vsftpd.conf.example` es suficiente para la mayoría de los servicios más comunes:

```
root# cp /etc/vsftpd/vsftpd.conf.example /etc/vsftpd/vsftpd.conf
```

5.2.5. Activación y desactivación del servicio

El demonio que hemos instalado se activa y desactiva de forma estándar. Véase el Apéndice C para saber cómo hacer esto.

Taller 5.7:

1. Instale un servidor Ftp en un PC virtual y configurelo para que los usuarios puedan escribir en sus home's.
2. Acceda desde el host huésped (Windows) a su cuenta de usuario en el host virtual y transfiera algún fichero. Esto puede hacerse invocando desde el intérprete de comandos del MS-DOS al programa `ftp`. Pruebe a hacer lo mismo desde el PC virtual.
3. Compruebe que activando y desactivando el servidor, el servicio se reanuda o deja de prestarse.

Cuestión 1: Describa brevemente para qué se utilizan los programas Telnet y Ftp.

Cuestión 2: Usando Wireshark (véase el Apéndice F) diseñe un time-line que muestre una interacción real entre un cliente y un servidor Telnet suponiendo que el usario sólo ejecuta el comando `exit` una vez que ha accedido al servidor remoto. Muestre sólo los paquetes generados por el protocolo Telnet que dependen de la sesión (los generados cuando introducimos nuestro login y nuestro password[2], y los que resultan de ejecutar el comando `exit`). En cada paquete transmitido indique el contenido del *payload* (la parte del paquete que es generada en la parte más alta de la pila de protocolos) y el instante en el que es capturado por el sniffer.

Cuestión 3: En la aplicación Ftp, ¿qué diferencias hay entre transmitir los ficheros en formato binario y en formato ASCII? Investigue en Internet.

Cuestión 4: Actualmente es imposible realizar Telnet o Ftp a un ordenador de la Universidad de Almería desde el resto de Internet, si estos servicios están siendo ofrecidos en los puertos estándar (véase el fichero `/etc/services`).[3] ¿Por qué esto es así? Sabiendo que, por ejemplo, el tráfico Web sí está permitido, ¿podría instalarse un servidor Telnet en el puerto 80 (el puerto usado por los servidores Web) y conectarnos a él desde fuera de la Universidad? Razone su respuesta. Indique en este último caso si se podría ofrecer otro servicio escuchando en el puerto 80.

[2] Altere el login y el password para no revelarlos ...

[3] Sin usar el redireccionamiento de puertos mediante una aplicación como SSH.

Práctica 6

La Web

En este módulo analizamos una de las aplicaciones más importantes de Internet: la Web. Básicamente veremos los siguientes aspectos:

1. Instalar y configurar básicamente un cliente Web.
2. Instalar y configurar básicamente un servidor Web.
3. Instalar y usar un proxy Web.
4. Depurar una interacción Web utilizando un sniffer.

6.1. Más sobre la Web

La World Wide Web (también llamada WWW o simplemente "la Web") es un sistema de información (actualmente enorme, posiblemente el más grande de la historia) basado en millones de páginas Web distribuidas por Internet. Esta información es servida bajo demanda por los servidores Web y consultada por los usuarios a través de los clientes (también llamados navegadores) Web.

6.1.1. Los servidores Web

Un servidor Web es un servicio que proporciona acceso a objetos Web alojados en un host con acceso a Internet. Los servidores Web aceptan peticiones (generalmente a través del puerto 80) de los clientes Web.

Existen muchos servidores Web. Aquí mostramos una lista con algunos de ellos: CERN httpd (`http://www.w3.org/Daemon`), Apache (`http://httpd.apache.org`) y Microsoft Internet Information Server (`http://www.microsoft.com/WindowsServer2003/iis/default.mspx`).

6.1.2. Los navegadores Web

Un navegador es la parte cliente de la Web. Cada vez que utilizamos uno y descargamos una página estamos realizando una petición HTTP a un servidor Web, y éste nos contesta con una respuesta HTTP que contiene el objeto Web solicitado.

Inicialmente los navegadores Web sólo podían mostrar texto. Sin embargo, actualmente existen muchos tipos de objetos Web que los navegadores son capaces de procesar (como imágenes, sonidos y vídeos, programas escritos en Java y Flash, etc.). Esta es una lista de algunos de los navegadores Web más famosos: Amaya (`http://www.w3.org/Amaya`), Epiphany (`http://www.gnome.org/projects/epiphany`), Galeon (`http://galeon.sourceforge.net`), Windows Internet Explorer (`http://www.microsoft.com/spain/windows/ie/default.mspx`), Konqueror (`http://konqueror.org`), Lynx (`http://lynx.browser.org`), Mozilla Firefox (`http://www.mozilla-europe.org/es/products/firefox`), Netscape Navigator (`http://browser.netscape.com`), Opera (`http://www.opera.com`), Safari (`http://www.apple.com/es/macosx/features/safari`) y Shiira (`http://hmdt-web.net/shiira/en`).

6.1.3. El HyperText Transfer Protocol

El HTTP (`http://www.w3.org/Protocols`) es el protocolo de transferencia de información básico[1] en la Web. Fue ideado a principios de los 90 en el CERN (`http://www.cern.ch`) por Tim Berners-Lee

[1] Ahora existen muchos otros que también se utilizan como el File Transfer Protocol (FTP).

(http://www.w3.org/People/Berners-Lee).

6.1.4. Los objetos Web y las URL's

El nombre común para todo lo que es posible transferir a través de la Web se conoce como *objeto Web*. Por definición cada objeto puede ser referenciado a través de una URL (Uniform Resource Locator). En los navegadores Web, las URLs se muestran en la entrada que se sitúa normalmente en la parte superior de la ventana del navegador.

6.1.5. El HyperText Markup Language

El HTML (http://www.w3.org/MarkUp) es el lenguaje en el que están escrito las páginas Web. En su versión más básica, una página Web es un conjunto de instrucciones escritas en HTML que referencian a otros objetos Web.

6.1.6. La W3C

La World Wide Web Consortium (W3C) es el organismo internacional encargado de regular el desarrollo de la Web. La Web es el sistema de información más importante de Internet y muchas compañías y grupos de investigación (entre otros) añaden constantemente nuevas funcionalidades. La W3C se encarga (básicamente) de que cualquier cliente Web sea capaz de comunicarse con cualquier servidor Web.

6.1.7. Los proxys Web

La Web consume un gran porcentaje del ancho de banda de Internet. Para reducir dicho consumo puede instalarse un proxy Web que es un sistema que funciona como una caché para los usuarios que los usan.

El funcionamiento de un proxy Web es bastante simple de entender. Un proxy típicamente es utilizado por los usuarios de una red con suficientemente ancho de banda. Si suponemos que generalmente dichos usuarios van a acceder a un conjunto de objetos Web más de una vez en un intervalo de tiempo dado, es más eficiente colocar dichos objetos en el proxy Web y acceder a él.

Un proxy Web funciona como servidor y como cliente. Cuando un usuario configura su navegador Web para utilizar un proxy Web y accede a un objeto Web a través de su URL original (que referencia el objeto en el servidor Web, no en el proxy Web), en realidad accede (siempre) al proxy Web. El proxy Web entonces busca el objeto en su caché y si lo encuentra "fresco"[2], se lo sirve al navegador Web cliente. Nótese que en este contexto el proxy Web está funcionando como un servidor Web.

Pero, ¿qué ocurre cuando el proxy Web no contiene el objeto Web solicitado? Entonces el proxy Web actúa como un cliente Web e intenta acceder al objeto Web a través de su URL original. Cuando lo consigue, actualiza su caché y lo envía al navegador Web del usuario.

Finalmente indicar que, aunque no configuremos nuestro navegador Web para utilizar un proxy Web, es probable que lo estemos utilizando. Muchos ISP colocan **proxys Web transparentes** para reducir el tráfico Web a través de su red.

Existen decenas de proxys Web. Sin embargo el más utilizado es, con diferencia Squid (http://www.squid-cache.org).

6.1.8. La caché de los navegadores Web

El tema del ahorro de ancho de banda es una cuestión fundamental, y no sólo para los administradores de las redes. Por este motivo, la mayoría de los navegadores Web implementan su propia caché con el objeto de reducir el tráfico. Así, cuando accedemos a un objeto, el navegador primero comprueba si tiene una copia del mismo en su caché y si este está "fresco", entonces el navegador visualiza el objeto sin necesidad de descargarlo de Internet.

6.2. Mozilla Firefox

Mozilla Firefox es un navegador Web que ha sido desarrollado por Mozilla Corporation (http://www.mozilla.org/reorganization) y cientos de voluntarios, siendo, junto con Microsoft Internet Explorer y Apple Safari, uno de los navegadores más utilizados. Además, está disponible prácticamente para todos los

[2]Para ello el proxy le pregunta al servidor Web que contiene el objeto si su copia es tan nueva como la que él tiene.

sistemas operativos con entorno gráfico y gracias a que se distribuye en código fuente es considerado, por muchos (y entre ellos el autor de este texto) como el navegador más seguro que existe.

6.2.1. Instalación

Debian Linux:

```
root# apt-get install firefox
```

Fedora Core Linux:

```
root# yum install firefox
```

Gentoo Linux:

```
root# emerge firefox
```

Taller 6.1: Instale firefox (si hiciera falta).

6.2.2. Ejecución

Muy sencillo:

```
usuario$ firefox &
```

Un detalle, sólo podemos lanzar Firefox en una consola gráfica (`xterm`, por ejemplo).

6.3. Apache

Apache es un servidor Web. De hecho, es el servidor Web más difundido actualmente (`http://news.netcraft.com/archives/web_server_survey.html`). Si a esto sumamos que la Web es la aplicación más difundida en Internet, podemos asegurar que Apache es el servidor más extendido de la historia.

6.3.1. Instalación

Debian Linux:

```
root# apt-get install apache2
```

Fedora Core Linux:

```
root# yum install httpd
```

Gentoo Linux:

```
root# emerge apache
```

Taller 6.2: Instale apache. Una vez instalado este paquete deberíamos poder conectarnos al servidor. Esto puede comprobarse si accedemos con el navegador a la URL:

```
http://localhost
```

y comprobamos que se lee un objeto Web (normalmente el fichero `index.html` que es el objeto por defecto cuando accedemos a un directorio).

También deberíamos tener acceso al servidor desde la línea de comandos:

```
root# apache2 -h
 apache2 -h
Usage: apache2 [-D name] [-d directory] [-f file]
               [-C "directive"] [-c "directive"]
               [-k start|restart|graceful|graceful-stop|stop]
               [-v] [-V] [-h] [-l] [-L] [-t] [-S] [-X]
Options:
  -D name            : define a name for use in <IfDefine name> directives
  -d directory       : specify an alternate initial ServerRoot
  -f file            : specify an alternate ServerConfigFile
  -C "directive"     : process directive before reading config files
  -c "directive"     : process directive after reading config files
  -e level           : show startup errors of level (see LogLevel)
  -E file            : log startup errors to file
  -v                 : show version number
  -V                 : show compile settings
  -h                 : list available command line options (this page)
  -l                 : list compiled in modules
  -L                 : list available configuration directives
  -t -D DUMP_VHOSTS  : show parsed settings (currently only vhost settings)
  -S                 : a synonym for -t -D DUMP_VHOSTS
  -t -D DUMP_MODULES : show all loaded modules
  -M                 : a synonym for -t -D DUMP_MODULES
  -t                 : run syntax check for config files
  -X                 : debug mode (only one worker, do not detach)
```

Pruebe finalmente a usar el servidor Web que ha instalado desde el sistema operativo huésped (Windows). Para ello deberá averiguar la dirección IP que utiliza el host virtual (Debian) y especificar la URL usando dicha dirección.

6.3.2. Configuración

La configuración de Apache depende del contenido del directorio:

Debian Linux:

```
/etc/apache2/
```

Fedora Core Linux:

```
/etc/httpd/
```

Gentoo Linux:

```
/etc/apache2/
```

Dicho directorio contiene una serie de ficheros y directorios. El fichero de configuración principal es `apache2.conf` o `httpd.conf` (dependiendo de la distribución). El resto de ficheros de configuración son cargados desde éste.

Taller 6.3: Localice el directorio con los ficheros de configuración de Apache.

Entre los directorios encontramos los que indican qué módulos hay disponibles y qué módulos hay habilitados (veremos más tarde qué son los módulos), así como qué *sites* hay disponibles y habilitados (esto está relacionado con los *virtual hosts*, más tarde también hablaremos de esto).

Cada vez que Apache es ejecutado (o re-ejecutado) se leen los ficheros de configuración que están bastante bien documentados. Dichos ficheros se organizan en secciones que contienen un conjunto de directivas. Cada directiva debe escribirse en una única línea. Veamos cómo controlar algunas de las posibilidades de Apache modificando dichas directivas y el contenido de los directorios de configuración:

Localización de los objetos Web en el servidor

La directiva `DocumentRoot` define el directorio que contiene los objetos Web del servidor. Dicha directiva cuando no se declara, tiene el siguiente valor por defecto:

Debian Linux:

```
/var/www/apache2-default
```

Fedora Core Linux:

```
/var/www/html
```

Gentoo Linux:

```
/var/www/localhost/htdocs
```

Taller 6.4: Localice el directorio que almacena los objetos Web.

Apache además puede servir los objetos que se colocan en el directorio `public_html` de los usuarios. Por ejemplo, para acceder al directorio `public_html` del usuario "alumno" en la máquina local deberímos utilizar la URL:

```
http://localhost/~alumno
```

Para conseguir esto debemos hacer lo siguiente:

1. Cargar el módulo "userdir". Apache es capaz de realizar las más diversas tareas que se le pueden encargar a un servidor Web. Como hay muchos usuarios que no utilizan a la vez todas las posibilidades, el servidor se ha diseñado de forma modular. Por tanto, dependiendo de la tarea que deseamos realizar habrá que cargar el módulo correspondiente. En este caso necesitamos cargar el módulo "userdir" (si es que no está ya cargado).

   ```
   # Comprobamos si el módulo "userdir" está cargado
   root# apache2 -M
   Loaded Modules:
    core_module (static)
    log_config_module (static)
    :
    status_module (shared)
   Syntax OK

   # Esto también puede hacerse mostrando el contenido del directorio:
   root# ls -l /etc/apache2/mods-enabled/
   total 0
   lrwxrwxrwx 1 root root 40 2007-01-08 18:51 actions.load ->\
    /etc/apache2/mods-available/actions.load
   lrwxrwxrwx 1 root root 28 2007-01-03 16:03 alias.load -> ../mods-available/alias.load
   :
   lrwxrwxrwx 1 root root 29 2007-01-03 16:03 status.load -> ../mods-available/status.load
   # Aunque aquí sólo aparecerían los módulos de tipo "shared", que no se
   # instalan por defecto en el servidor

   # Si no lo está, configuramos Apache para que lo carge
   root# a2enmod userdir

   # Forzamos a Apache a leer todos los módulos
   root# /etc/init.d/apache2 force-reload
   ```

2. Crear el directorio que almacena los objetos Web:

```
# Creamos el directorio
alumno$ mkdir public_html

# Comprobamos que tenga premisos de lectura y de acceso
alumno$ ls -la public_html/
total 8
drwxr-xr-x  2 alumno alumno 4096 2007-01-13 13:27 .
drwxr-xr-x 14 alumno alumno 4096 2007-01-13 13:27 ..

# Copiamos un fichero con un objeto Web
alumno$ cp /var/www/apache2-default/index.html public_html

# Comprobamos que tenemos acceso a dicho fichero
alumno$ firefox http://localhost/~alumno &
```

Taller 6.5: Habilite Apache para que los usuarios puedan publicar objetos Web en sus directorios `public_html`. Compruebe que el servicio funciona correctamente.

Taller 6.6: Desinstale e instale (de nuevo) el módulo `userdir`. Compruebe ambas situaciones comprobando si el módulo está funcionando.

Binding

Apache por defecto escucha en el puerto 80 a través de todos los interfaces de red de los que dispone el host. Este comportamiento puede modificarse para:

Escuchar en un puerto distinto del 80: Esto se hace a través de la directiva `Listen`. Ejemplo:

```
# Usando un editor de ficheros ASCII, buscar en el fichero de
# configuración de Apache la directiva "Listen 80" y cambiarla por
# "Listen 8080". Si dicha directiva aparece en el fichero
# "/etc/apache2/ports.conf", esto puede hacerse también con los
# siguientes comandos:
root# sed "s/80/8080/g;" /etc/apache2/ports.conf > /etc/apache2/ports.conf.new
root# mv /etc/apache2/ports.conf.new /etc/apache2/ports.conf
# Ojo que es posible que además haya que modificar algún fichero extra!
# Léase detenidamente los comentarios en /etc/apache2/ports.conf
# acerca de esta posibilidad.

# Re-iniciamos Apache.
root# /etc/init.d/apache2 restart

# Comprobamos el cambio.
alumno$ firefox http://localhost:8080 &

# Deshacemos el cambio.
root# sed "s/8080/80/g;" /etc/apache2/ports.conf > /etc/apache2/ports.conf.new
root# mv /etc/apache2/ports.conf.new /etc/apache2/ports.conf

# Reiniciamos Apache
root# /etc/init.d/apache2 restart
```

Taller 6.7: Realice las modificaciones oportunas para que Apache escuche en el puerto 8080 y compruebe que realmente es así. Finalmente deshaga dichas modificaciones y compruebe que Apache vuelve e escuchar en el puerto 80.

Escuchar en más de un puerto: Ejemplo:

```
root# cat >> /etc/apache2/ports.conf << EOF
Listen 8080
EOF

# Reiniciamos Apache
root# /etc/init.d/apache2 restart

# Comprobación
alumno$ firefox http://localhost:80 &
alumno$ firefox http://localhost:8080

# Desahacemos el cambio.
root# sed "s/Listen 8080//g;" /etc/apache2/ports.conf > /etc/apache2/ports.conf.new
root# mv /etc/apache2/ports.conf.new /etc/apache2/ports.conf

# Reiniciamos Apache
root# /etc/init.d/apache2 restart
```

Taller 6.8: Realice las modificaciones oportunas para que Apache escuche simultaneamente en los puertos 80 y 8080, y compruebe que realmente es así. Finalmente deshaga dichas modificaciones.

Escuchar por diferentes direcciones IP y mostrando diferentes objetos Web: ... en función de la dirección IP utilizada por el cliente. Aunque no es muy frecuente, sí que es posible instalar en un host más de un adaptador de red, cada uno configurado con una dirección IP distinta. En este contexto se puede configurar Apache para que sirva diferentes objetos Web en función del adaptador por el que recibe las peticiones. A esta cualidad de Apache se le llama *IP-based Virtual Hosting*. Para ver un ejemplo nos vamos a aprovechar de que el host con el que estamos trabajando tiene asignadas dos direcciones IP diferentes, una para `localhost` y otra para el nombre de la máquina `lab-redes`.

```
# Comprobamos la primera dirección IP.
alumno$ ping -c 1 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.739 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.739/0.739/0.739/0.000 ms

# Comprobamos la segunda (nótese que ahora es .1.1)
alumno$ ping -c 1 lab-redes
PING lab-redes.localdomain (127.0.1.1) 56(84) bytes of data.
64 bytes from lab-redes.localdomain (127.0.1.1): icmp_seq=1 ttl=64 time=0.048 ms

--- lab-redes.localdomain ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.048/0.048/0.048/0.000 ms

# Añadimos el host virtual.
root# cat >> /etc/apache2/apache2.conf << EOF

<VirtualHost lab-redes>
  DocumentRoot /var/www/lab-redes/
</VirtualHost>
EOF

# Creamos el nuevo "site".
root# mkdir /var/www/lab-redes

# Metemos dentro un objeto.
```

```
root# cp /var/www/apache2-default/index.html /var/www/lab-redes/

# Lo cambiamos un poco:
root# sed "s/It works/Funciona/g;" /var/www/lab-redes/index.html >\
 /var/www/lab-redes/index.html.new
root# mv /var/www/lab-redes/index.html.new /var/www/lab-redes/index.html

# Reiniciamos Apache
root# /etc/init.d/apache2 restart

# ¿Funciona?
alumno$ firefox localhost &
alumno$ firefox lab-redes
```

Taller 6.9: Realice las anteriores configuraciones y compruebe que funcionan.

Escuchar en una única dirección IP, pero mostrando diferentes objetos: ... en función del alias del servidor que utiliza el cliente. El DNS tiene la posibilidad de asignar más de un nombre a una dirección IP. A uno de estos nombres se le conoce como nombre canónico y al resto como aliases. Aprovechándonos de esta posibilidad, el servidor Apache puede servir un conjunto de objetos Web diferente en función el alias que está usando el cliente para reclamar los objetos al servidor.[3] De esta forma parecerá que estamos accediendo a un servidor ejecutado en otra máquina diferente, cuando en realidad estamos utilizando el mismo. A esta posibilidad se le conoce como *Name-based Virtual Hosting*. Un ejemplo:

```
# Añadimos un alias del host en el DNS.
root# sed \
"s/127.0.1.1\tlab-redes.localdomain\tlab-redes/127.0.1.1\tlab-redes.localdomain\tlab-redes\talia
 /etc/hosts > /etc/hosts.new
root# mv /etc/hosts.new /etc/hosts

alumno$ ping -c 1 alias_lab-redes
PING lab-redes.localdomain (127.0.1.1) 56(84) bytes of data.
64 bytes from lab-redes.localdomain (127.0.1.1): icmp_seq=1 ttl=64 time=0.060 ms

--- lab-redes.localdomain ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.060/0.060/0.060/0.000 ms

# Añadimos el host virtual.
root# cat >> /etc/apache2/apache2.conf << EOF

NameVirtualHost alias_lab-redes
<VirtualHost alias_lab-redes>
  DocumentRoot /var/www/alias_lab-redes/
</VirtualHost>
EOF

# Creamos el nuevo "site".
root# cp -rav /var/www/lab-redes/ /var/www/alias_lab-redes

# Lo cambiamos un poco:
root# sed "s/Funciona/Funciona de nuevo/g" /var/www/alias_lab-redes/index.html\
 > /var/www/alias_lab-redes/index.html.new
root# mv /var/www/alias_lab-redes/index.html.new /var/www/alias_lab-redes/index.html

# Reiniciamos Apache
```

[3] Esto puede ser muy útil para servir las páginas en diferentes idiomas en función del nombre que usamos para referenciar al servidor Web.

```
root# /etc/init.d/apache2 restart

# ¿Funciona?
alumno$ firefox alias_lab-redes &
```

Taller 6.10: Realice las anteriores configuraciones y compruebe que funcionan.

6.4. Squid

Squid (`http://www.squid-cache.org/`) es un proxy Web[4] que se distribuye en código fuente (bajo la licencia GNU).

6.4.1. Instalación

Debian Linux:

```
root# apt-get install squid
```

Fedora Core Linux:

```
root# yum -y install squid httpd
```

Gentoo Linux:

```
root# emerge squid
```

Taller 6.11: Instale Squid.

6.4.2. Configuración

Para configurar Squid necesitamos editar el fichero:

`/etc/squid/squid.conf`

Por defecto, Squid funciona como un proxy Web escuchando en el puerto 3128, pero no atiende a ningún cliente.

Aunque es una configuración muy abierta, configure Squid para que atienda a cualquier cliente. Para ello debe comentar la línea que dice "`http_access deny all`" y escribir otra que diga "`http_access allow all`". Esto es suficiente para nosotros y por tanto, no entraremos en detalle sobre el resto de opciones. Para más información, el fichero de configuración está debidamente autocomentado.

Taller 6.12: Modifique el fichero de configuración de Squid para que escuche a cualquier host.

6.4.3. Utilización

Para usar Squid es necesario modificar la configuración de nuestro navegador:

Mozilla Firefox 1.5 *Preferencias -> General -> Configuración de Conexión -> Configuración manual del proxy*. Donde pone *Proxy HTTP*: poner la dirección IP del host que sirve el proxy y el puerto 3128.

Mozilla Firefox 2.0 *Herramientas -> Opciones -> Avanzado -> Red -> Configuración manual del proxy*. Donde pone *Proxy HTTP*: poner la dirección IP del host que sirve el proxy y el puerto 3128.

Microsoft Explorer *Herramientas -> Opciones de Internet -> Conexiones -> Configuración de LAN -> Usar un servidor proxy para la LAN*: poner la dirección IP del host que sirve el proxy y el puerto 3128.

Iceweasel *Editar -> Preferencias -> avanzado -> Conexión/Configuración*: Poner la dirección IP del host que sirve el proxy y el puerto 3128.

Taller 6.13: Tanto en el host huésped (Windows) como en el host virtual (Debian), configure su navegador para utilizar el proxy Web que acaba de instalar. Compruebe que puede navegar. Finalmente deshaga dichos cambios en el navegador, pero no desinstale el proxy.

[4] Aunque puede funcionar como proxy FTP, Gopher, WAIS, DNS, ...

6.5. Análisis de las interacciones Web

En esta sección vamos a analizar el protocolo HTTP usando la aplicación Wireshark. Para saber cómo utilizar esta utilidad véase el Apéndice F.

6.5.1. Análisis de una interacción Web básica

La interacción Web básica transfiere un objeto Web por primera vez desde un servidor Web hasta un cliente Web.

Taller 6.14:

1. Ejecute Wireshark (recuerde, como `root`) y colóquelo en modo de captura a través del interface `eth0`.
2. Ejecute un navegador Web (firefox, preferiblemente) y acceda al objeto Web `http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file1.html`.
3. Cuando el objeto Web haya sido mostrado por el navegador, detenga la captura de paquetes. Debería ver (al menos) dos paquetes HTTP, un GET y un OK.

Cuestión 1: ¿Qué versión del HTTP está usando su navegador? ¿Y el servidor?

Cuestión 2: ¿Qué idiomas acepta el navegador?

Cuestión 3: ¿Cuál es la dirección IP del host que ejecuta el navegador? ¿Cuál es la dirección IP del host que ejecuta el servidor?

Cuestión 4: ¿Cuál fue el código de estado retornado por el servidor?

Cuestión 5: ¿Cuánto tiempo ha pasado desde que el GET fue emitido hasta que el OK fue recibido?

Cuestión 6: ¿Cuándo fue por última vez modificado el objeto que se está reclamando al servidor? ¿Está este fichero constantemente siendo modificado?

Cuestión 7: ¿Cuántos bytes de datos ocupa el objeto Web?

Cuestión 8: ¿Su navegador ha solicitado, además, un objeto llamado `favicon.ico`? ¿Qué contiene dicho objeto?

6.5.2. Análisis de una interación Web condicional

Los navegadores utilizan su caché para evitar la descarga de objetos Web cacheados y que están frescos (que son iguales a los que tiene el servidor). Para esto se utiliza el GET condicional.

Taller 6.15:

1. Ejecute su navegador Web y asegúrese de que la caché de éste está vacía (En Mozilla Firefox: *Edit* -> *Preferences* -> *Advanced* -> *Network* -> *Cache: Clear Now*).
2. Ejecute el packet sniffer.
3. Acceda al objeto Web (`http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file2.html`). El browser debería mostrar un fichero HTML.
4. Re-acceda al mismo objeto (pulse el botón de "Reload current page").
5. Detenga la captura de paquetes y use el filtro `http`.

Cuestión 9: ¿Qué cabecera HTTP identifica al GET condicional? ¿En qué GET (de los dos que hay en la lista de paquetes capturados) aparece dicha cabecera? ¿Qué diferencia existe entre las dos respuestas que ha enviado el servidor?

6.5.3. Análisis de una interacción Web que transmite un objeto "largo"

El TCP del emisor fragmenta los mensajes largos. El receptor recibe un conjunto de paquetes, cada uno en un segmento diferente.

Taller 6.16:

1. Ejecute su navegador Web y borre su caché.
2. Ejecute el packet sniffer.
3. Reclame el objeto Web `http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file3.html` que contiene la *US Bill of Rights*.
4. Detenga la captura de paquetes y use el filtro `http`.

Cuestión 10: ¿Cuántos mensajes de petición HTTP ha generado el navegador Web? ¿Cuántos objetos Web se han transmitido? ¿Cuántos segmentos TCP se han recibido?

6.5.4. Análisis de una interacción Web con objetos empotrados

Cuando accedemos a un objeto Web que a su vez referencia a otros objetos Web, el cliente se encarga de ir reclamando cada uno de ellos.

Taller 6.17:

1. Ejecute su navegador Web y borre su caché.
2. Ejecute el packet sniffer.
3. Reclame el objeto Web `http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file4.html` que contiene un corto fichero HTML con dos imágenes.
4. Detenga la captura de paquetes y use el filtro `http`.

Cuestión 11: ¿Cuántos mensajes de petición HTTP ha generado el navegador Web?

Cuestión 12: ¿Qué servidor Web ha servido cada imagen?

Cuestión 13: ¿Cuántos objetos Web se han transmitido?

Cuestión 14: Especifique si los distintos objetos Web se han descargado en serie o en parelelo e indique por qué.

6.5.5. Análisis del funcionamiento de un proxy Web

Cuestión 15: En esta práctica ha aprendido a instalar un proxy Web. Realice un experimento que le permita conocer si, al menos el proxy Web que acaba de instalar, consulta siempre al siguiente servidor en la jerarquía de la Web (otro proxy o el servidor Web original), cuando recibe una petición por parte de un cliente Web. Explique en detalle dicho experimento y cómo ha determinado el anterior comportamiento.

Práctica 7

El correo electrónico

En esta práctica aprenderemos a:

1. Instalar un servidor de correo electrónico.
2. Instalar un programa para leer y escribir correos electrónicos.
3. Utilizar el protocolo SMTP tal y como lo usan los servidores de correo.

7.1. The Internet mail infrastructure

7.1.1. Servidores de e-mail, servidores SMTP, mail exchangers, mailers y MTA's ???

Un servidor de e-mail, también llamado servidor SMTP, mail exchanger, mailer y MTA (Mail Transfer Agent) es un proceso servidor que recibe e-mails debidamente formateados utilizando el **Simple Mail Transfer Protocol** (SMTP) y los retransmite (si es preciso) a otros servidores de e-mail. Utilizaremos en adelante sólo el término *mailer* para referirnos a un servidor SMTP.

7.1.2. Lectores de e-mails, escritores de e-mails y MUA's

Aunque es posible utilizar Telnet (hablando en SMTP con el mailer) para leer y escribir correos directamente, es mucho más cómodo utilizar un programa que simplifique dicho proceso. Estos programas generalmente se llaman MUAs (Mail User Agents). Ejemplos más que conocidos son Microsoft Outlook (`http://www.microsoft.com/spain/office/products/outlook/default.mspx`), Mozilla Thunderbird (`http://www.mozilla-europe.org/es/products/thunderbird/`) y Novell Evolution (`http://www.gnome.org/projects/evolution/`).

Un MUA no tiene que ser necesariamente un programa que se instala en nuestra computadora. En los últimos tiempos se han puesto de moda los **MUAs vía Web** (también llamados "Web Mail") (como Hotmail (`http://www.hotmail.com`), Yahoo! Mail (`http://mail.yahoo.com`) y Gmail (`http://gmail.google.com/`), que a través de una páquina Web permiten enviar y recibir e-mails usando simplemente nuestro navegador Web favorito.

7.1.3. Acerca de los clientes y de los servidores

Los MUAs no son los únicos clientes del sistema de transmisión de e-mails propiamente dicho. De hecho, los mailers funcionan como clientes cuando los e-mails que reciben no van dirigidos a una cuenta de usuario del propio host donde se ejecuta el mailer porque se comunican con otro mailer para retransmitir el correo.[1]

7.1.4. Las direcciones de correo

Una dirección de correo es cualquier cadena que contiene, al menos, un símbolo `@`. Ejemplo:

```
vruiz@ual.es
```

[1] En el mundo de los servidores de correo se habla de hacer "relay".

En este ejemplo, la cadena `ual.es` se conoce como *dominio de correo*. A la cadena `vruiz` se le conoce como *box part*.

Otro aspecto importante a tener en cuenta es que las direcciones de correo son interpretadas de la misma forma en mayúsculas y en minúsculas. Por tanto:

```
Vruiz@Ual.Es
vruiz@ual.es
VRUIZ@UAL.ES
```

son la misma dirección.

7.1.5. Los mensajes de correo

Un mensaje es una secuencia de líneas, donde una línea es una cadena de cero o más caracteres imprimibles ASCII de 7 bits (véase la tabla de la Sección I) terminada en `\015\012` (en DOS y Windows) o en `\012` (en Unix). Los caracteres imprimibles ASCII de 7 bits son los que aparecen en la tabla ASCII entre las posiciones 33 y 127, ambas incluídas.

7.1.6. Los sobres (envelopes)

El remitente y el destinatario de un correo electrónico deben estar identificados de forma única mediante direcciones de correo. A un par de direcciones remitente/destinatario es a lo que se conoce como *sobre*. También es posible que un sobre contenga varios destinatarios o ningún remitente.

7.1.7. Bounce messages

Es responsabilidad de los mailers entregar correctamente los correos electrónicos una vez que estos han sido aceptados. Sin embargo, cuando por alguna circunstancia esto no es posible (porque por ejemplo, el destinatario no existe o el mailer destino está apagado) algunos mailers envían correos de aviso al remitente indicando el problema. Estos mensajes se llaman *bounce messages*.

7.1.8. Registros DNS, distancias y responsabilidades de los mailers

Los mailers consultan al DNS cuando tienen que reenviar hacia otro mailer un correo electrónico. Por ejemplo, con:

```
alumno$ /usr/bin/host -t MX ual.es
ual.es mail is handled by 10 mail.rediris.es.
ual.es mail is handled by 2 smtp.ual.es.
```

estamos preguntando al DNS qué mailers son los responsables para el dominio `ual.es`.[2] Como podemos ver, existen dos hosts responsables del intercambio del correo electrónico en la Universidad de Almería; (`mail.rediris.es` y `smpt.ual.es`). El primero (`mail.rediris.es`) está a una distancia (en hops[3]) de 10 de la red que tiene como dominio `ual.es`, mientras que el segundo está a una distancia de 2. Esto significa que cualquier mailer (dentro o fuera de la Universidad) elegirá primero a `smtp.ual.es` para entregar un correo electrónico con este dominio. Cuando esta conexión no sea posible (porque `smpt.ual.es` ha caído, por ejemplo) se utilizará `mail.rediris.es`. Si ninguno de estos hosts está disponible, el mailer generalmente envía un bounce message al remitente e intentará la conexión SMTP durante los próximos días (generalmente 6). Si finalmente el mensaje no es entregado se enviará otro *bounce message* al remitente indicando este evento y el mailer destruye el mensaje.

Taller 7.1: Encuentre los mailers para el dominio `ya.com`, o cualquier otro dominio de correo que conozca. Haga un ping a dichos mailers para comprobar que están "vivos".

7.2. El SMTP (Simple Mail Transfer Protocol)

El sistema de transferencia de correos electrónicos utiliza el modelo cliente/servidor y un protocolo, el SMTP (RFC 821, RFC 1123, RFC 1425, RFC 1651, RFC 1869 y RFC 2821) para comunicarse.

[2] Recuerde que el resultado de esta consulta no depende del host en que se realiza.

[3] Pasos por un router.

7.2.1. Peticiones y respuestas

En una conversación SMTP, un MUA o un mailer (cliente) envía un conjunto de *peticiones* (en inglés, *requests*) a un mailer (servidor) y éste le contesta con *respuestas* (*responses*). A grandes rasgos el proceso petición-respuesta es así:

1. Cuando un cliente se conecta a un servidor, éste envía una respuesta que indica si acepta la conexión o la rechaza (temporal o permanentemente). Esta respuesta inicial se conoce como *saludo* (*greeting*).
2. Si el servidor acepta la conexión, el cliente envía cero o más peticiones al servidor. En general el cliente no debe enviar una nueva petición al servidor si la anterior respuesta no ha sido recibida.

7.2.2. Los verbos

El formato de una petición es siempre una cadena ASCII imprimible:

```
<verb> [<parameter>]\015\012
```

donde `<verb>` puede ser:

`HELO`: Abreviatura de "hello". Va seguido de un parámetro que indica al servidor, el nombre del cliente. Ejemplo:

```
HELO gogh.ace.ual.es
```

`RSET`: Abreviatura de "reset". Se utiliza para borrar el sobre que se acaba de enviar al servidor. Esto generalmente se realiza cuando se van a enviar varios correos electrónicos desde un mismo cliente a un mismo servidor durante una única conexión SMTP (que en este caso se llama conexión SMTP persistente).

`NOOP`: Abreviatura de "no operation". Se utiliza para testear el estado de la conexión sin enviar información alguna.

`MAIL`: Debe ir seguido de la cadena "`FROM:`" y de la dirección de correo del remitente.

`RCPT`: Abreviatura de "recipient". Debe ir seguido de la cadena "`TO:`" y de la dirección de correo del destinatario. Cuando hay varios destinatarios se envían varios "`RCPT TO:`".

`DATA`: No tiene argumentos y se utiliza para indicar al servidor que todas las líneas que van a continuación forman el cuerpo del mensaje del correo electrónico. Este cuerpo acaba en una línea que sólo contiene un punto "`.`".

`QUIT`: Solicita al servidor el cierre de la conexión.

Véase el RFC correspondiente para conocer el resto de verbos.

7.2.3. Códigos

El formato de una respuesta es siempre una secuencia de cadenas ASCII imprimibles:

```
<code> <string>\015\012
```

donde `<code>` es un código de error que indica el carácter de la respuesta. Por ejemplo, el código 250 indica que la petición ha sido aceptada. Véase el RFC correspondiente para conocer el resto de códigos.

7.3. Las cabeceras de los correos electrónicos

En esta sección vamos a analizar el formato de las cabeceras que se transmiten como parte de los correos electrónicos. Dichas cabeceras son utilizadas por los mailers para personalizar el envío de los e-mails. Nótese que en ningún caso esta información debe ser confundida con la que se transfiere en una interacción SMTP. Toda esta información es creada bien por el remitente como parte del cuerpo del mensaje, bien por el mailer durante el procesamiento y la transmisión del e-mail.

7.3.1. Formato

Una cabecera de correo electrónico es una secuencia de líneas no vacías que concatenan una lista de campos. A su vez, un campo es una secuencia de una o más líneas no vacías que contienen un nombre de campo y un valor. Un ejemplo:

```
Received: (qmail-queue invoked by uid 666);
  30 Jul 1996 11:54:54 -0000
From: "D. J. Bernstein" <djb@silverton.berkeley.edu>
To: fred@silverton.berkeley.edu
Date: 30 Jul 1996 11:54:54 -0000
Message-ID: <19951223192543.3034.qmail@silverton.berkeley.edu>
Reply-to: <djbernstein@gmail.com>
Subject: Go, Bears!
```

Como podemos ver el primer campo de nombre `Received:` tiene un valor que ocupa dos líneas. Además, la cabecera finaliza siempre en una línea vacía.

7.3.2. Descripción de los principales campos

La lista de campos que se muestra a continuación no es exhaustiva. Para una lista mucho más completa, véase la documentación asociada al mailer Qmail (`http://cr.yp.to/immhf/index.html`) y/o los RFC correspondientes.

El campo `Received:`

Cada mailer inserta un campo `Received:` cuando procesa un mensaje entrante para indicar quién se lo envía y cuándo. De esta manera podemos averiguar qué saltos ha dado (por qué mailers ha pasado) el e-mail hasta que ha llegado hasta nosotros.

El campo `Date:`

Fecha en la que el primer MTA creó el e-mail. La diferencia de tiempos entre el que especifica `Received:` y `Date:` es el tiempo que ha tardado el e-mail en transferirse desde el mailer origen hasta el destino.

El campo `To:`

Especifica la lista de destinatarios. Cada destinatario es separado del siguiente mediante una "," (una coma). Ejemplo:

```
To: The Boss <God@heaven.af.mil>,
    <angels@heaven.af.mil>,
    Wilt . (the  Stilt) Chamberlain@NBA.US
```

En este ejemplo hay tres destinatarios.

El campo `Cc:`

`Cc` significa *carbon copy* (copia al carbón) y se utiliza para especificar una lista de destinatarios de "segundo orden". De esta manera, los que reciben el e-mail saben si el correo está directamente dirigidos a ellos o están recibiendo simplemente una copia.

El campo `Bcc:`

`Bcc` significa *bind carbon copy* (copia al carbón oculta) y se utiliza para especificar una lista de destinatarios del e-mail que no se van a identificar entre sí. Para hacer esto, el MUA elimina el campo `Bcc:` antes de enviar el mensaje.

El campo `Subject:`

Introduce una cadena explicatoria del contenido del e-mail. Este campo lo rellena el remitente del e-mail y suele ser la línea que sigue al verbo `DATA` en la transacción SMTP.

El campo `Reply-to:`

Se utiliza para especificar una dirección de remitente distinta a la que se especificó con el verbo `MAIL` en la interacción SMTP. Esto es útil cuando el remitente tiene una cuenta en (acceso a) diferentes mailers y quiere recibir las respuestas (*replies*) siempre a la dirección de correo indicada por `Reply-to:`.

El campo `Message-ID:`

Introduce una cadena que identifica al e-mail. Para intentar conseguir una etiqueta única se utiliza generalmente un número de serie seguido del dominio del mailer que está procesando el e-mail.

Los campos `Resent-From:`**,** `Resent-To:` **y** `Resent-Subject:`

Estos campos son creados cuando reenviamos un e-mail. Los campos `From:`, `To:` y `Subject:` son copiados a `Resent-From:`, `Resent-To:` y `Resent-Subject:`, respectivamente. De esta manera quedan libres para obtener otros valores cuando se reenvía el e-mail.

El campo `Return-Path:`

Indica la lista de mailers a los que puede enviarse un bounce message cuando aparece algún problema con la entrega del e-mail.

7.4. Utilidad de un servidor local

En esta práctica vamos a aprender a instalar un servidor de correo electrónico. Algunas de las ventajas de que nuestro *mail exchanger* resida en el host local son:

1. No es necesario estar permanentemente conectado a Internet para enviar e-mails a usuarios remotos. El sistema de correo se encarga de encolar los mensajes salientes y de chequear periódicamente si la conexión se ha restablecido.
2. Aunque no tengamos nunca conexión a Internet, los distintos usuarios de nuestro host (recuérdese que Linux es un sistema operativo multiusuario) podrán escribirse entre sí.

7.5. El correo electrónico en redes privadas

Como cualquier otro servicio que se ofrece dentro de una red privada tenemos que tener en cuenta que, a menos que configuremos nuestro NAT redireccionando los puertos correspondientes, el servicio sólo va a estar disponible para los hosts de la red privada.

Puesto que el servicio de correo electrónico escucha de forma estándar en el puerto 25, simplemente deberíamos redireccionar este puerto desde el NAT hacia el host y el puerto (probablemente, también el 25) donde el mailer está escuchando.

7.6. Exim

Exim (`http://www.exim.org`) es un potente mailer que (a diferencia de otros como Qmail (`http://cr.yp.to/qmail.html`), puede ser distribuido mediante código compilado. Veremos además que su configuración es bastante simple.

7.6.1. Instalación

Debian:

```
root# apt-get install exim4
```

Fedora Core:

```
root# yum install exim
```

Gentoo:

```
root# emerge exim
```

7.6.2. Configuración

Para configurar un servidor de correo debemos tener en cuenta varios aspectos:

1. Si vamos a aceptar correo entrante para usuarios del host que ejecuta el mailer. Normalmente la respuesta será afirmativa.
2. Si estamos en una red privada y si nuestro router-NAT tiene asignada una dirección IP dinámica. Si así es, deberíamos configurar el mailer para que él no entrege directamente el correo saliente porque es probable que nos consideren como un spammer (y no nos hagan ni caso). En este caso no queda más remedio que usar otro mailer (probablemente el del ISP que en el caso de ejecutar la máquina virtual en un PC de la Universidad de Almería debería ser el host `smtp.ual.es`) que haga la entrega por nosotros. A este mailer se le suele llamar *smarthost*. Evidentemente deberemos conocer su dirección IP o su nombre[4].
3. El dominio que vamos a usar para conocer si el correo entrante va dirigido a un usuario local. Este dominio debería coincidir con el que figura en el fichero `/etc/resolv.conf`. Es importante utilizar un dominio real si vamos a mandar correos al exterior porque muchos servidores SMTP comprueban que el dominio del remitente exista.[5]
4. Los mailers que tienen permiso para entregarnos correo entrante. En principio, todos.
5. Otros dominios que los destinatarios quieren usar para recibir el correo. Por supuesto el DNS tiene que estar al tanto de esto.
6. Si vamos a aceptar correo entrante para usuarios que no tienen cuenta en el host que ejecuta el mailer. En otras palabras, si el mailer que estamos instalando va a ser un *smarthost*.
7. El formato de los *Mail Boxes*. Normalmente usaremos el formato "mbox", que es el estándar. En este formato todos los correos aparecen concatenados en un único fichero ASCII.

Debian:

```
root# dpkg-reconfigure exim4-config
# El instalador solicitará toda la información denecesaria con una
# serie de preguntas. Esta información se escribe en el fichero
# "/etc/exim4/update-exim4.conf.conf".
```

Fedora Core: Es necesario configurar el servidor modificando los ficheros correspondientes. Acceder a la Web de Exim (`http://www.exim.org`) para más información.

Gentoo: Como la mayoría de las configuraciones en Gentoo, hay que modificar los ficheros de configuración correspondientes. En `http://gentoo-wiki.com/Mail_Server_based_on_Exim_and_Courier-imap` encontrará una buena documentación.

Taller 7.2: Instale y configure Exim.

7.7. Un MUA: Mutt

Para no tener que ir leyendo "a pelo" los mail-boxes y usando el SMTP, un MUA como Mutt (`http://www.mutt.org`) nos va a venir de perlas.

[4] Usar el comando `host` para determinar la dirección IP del smarthost a partir de su nombre.
[5] En dicho dominio no debería aparecer el nombre del host, sólo el dominio ("`ual.es`", por ejemplo).

7.7.1. Instalación

Debian:

```
root# apt-get install mutt
```

Fedora Core:

```
root# yum install mutt
```

Gentoo:

```
root# emerge mutt
```

Taller 7.3: Instale Mutt.

7.7.2. Utilización

Mutt es un MUA que funciona en una consola de texto, pero no por ello deja de ser muy funcional. Veamos su utilización básica:

Enviar un correo: La forma más simple de enviar un correo es:

```
# En este ejemplo nos enviamos un correo
alumno$ mutt alumno@localhost
```

Mutt preguntará si deseamos modificar el destinatario. Pulsar <Enter> si no es así. Mutt preguntará por un *subject*. Pulsar <Enter> cuando lo hayamos especificado. Mutt usará el editor especificado en la variable de entorno EDITOR para crear el cuerpo del mensaje (o en su ausencia, un editor que corre en la consola que ya esté instalado, tal como joe, vi, nano, vim, etc.). Cambiar este comportamiento con:

```
alumno$ export EDITOR=/path/al/editor
```

Cuando hayamos salvado el cuerpo del mensaje (recuérdese que Mutt utiliza un editor de ficheros ASCII externo) y hayamos cerrado el editor, Mutt mostrará un menu para enviar el mensaje (pulsar <y>), abortar (<q>), cambiar el destinatario (<t>), añadir un CC (<c>), adjuntar un archivo (<a>), cambiar el subject (<d>) o ayuda (<?>). Normalmente pulsaremos <y>.

Una de las ventajas de Mutt es que puede crear el cuerpo del mensaje usando la redirección de la entrada estándar. Así, es muy sencillo enviar a alguien un fichero:

```
alumno$ mutt alumno@host < fichero.txt
```

Leer un correo: Basta con llamar a Mutt:

```
alumno$ mutt
```

Mutt mostrará el Mail Box con los mensajes antiguos que no hayan sido borrados y los nuevos. Nos moveremos con el cursor hasta el correo que queremos leer y pulsamos <Enter>.

Una nota. Exim almacena el correo entrante en la cola correspondiente (/var/spool/mail/usuario) y de ahí lo lee Mutt. Cuando es leído por Mutt (no todavía por nosotros), Mutt tratará de almacenarlo en nuestro home, concretamente en el fichero mbox.[6] Mutt pude procesar cualquier Mail Box a través de la opción -f. Ejemplo:

```
# Releyendo los mensajes de correo almacenados en nuestro Mail Box:
alumno$ mutt -f $HOME/mbox
```

---

[6] Aunque esto depende de si el mailer ya lo ha hecho o no.

Taller 7.4: Envíese un e-mail al host local. Envíese un fichero a través del e-mail al host local. Repita ambos procesos usando una dirección de correo en otro host donde tenga cuenta de correo (Hotmail, Gmail, etc.). Compruebe que las entregas han sido correctas.

Taller 7.5: Para comprender cómo funciona el SMTP vamos a enviarnos un correo anónimo. Este correo se distingue porque cuando es recibido, entre la información que acompaña al cuerpo del mensaje (el correo en sí mismo) no figura la dirección de correo electrónico real de quien lo envía. Esto se puede hacer porque los mailers cuando aceptan enviar un correo no chequean que el remitente del correo existe.

Nos vamos a enviar un correo anónimo a nosotros mismos en dos casos: (1) utilizando el mailer que acabamos de instalar en nuestro host y (2) utilizando el mailer del dominio en el que estamos. Es importante saber que si usamos un mailer que no pertenece a nuestro ISP es muy probable que no deje que lo usemos para enviar correos. Cuando esto sea así, recibiremos un mensaje por parte del servidor de correo indicando este hecho.

Usando el mailer local: Para enviar el correo vamos usar el programa Telnet y nos conectaremos al puerto 25, donde escuchan los servidores de correo:

1. Ejecutar:

   ```
   alumno$ telnet
   ```

   y pulsar <Return>. Aparecerá un mensaje de bienvenida, otro indicando cuál es la secuencia de escape que se utiliza para cerrar de forma inesperada una conexión (como la que vamos a establecer) y finalmente un prompt (una cadena indicando que el interprete de comandos de Telnet está esperando a que le introduzcamos un comando).
2. Escribir:

   ```
   open localhost 25
   ```

   y esperar a recibir el mensaje de bienvenida por parte del servidor de correo. Dicho mensaje comenzará siempre por 220.
3. Una vez que lo veamos, escribiremos textualmente (excepto donde pone `nuestro_login_real` que escribiremos nuestro login real en `localhost`):

   ```
   HELO anonimo
   MAIL FROM: <anonimo@ya.com>
   RCPT TO: <nuestro_login_real@localhost>
   DATA
   SUBJECT: Este es un correo con un remitente falso

   Hola!
   <anonimo@ya.com> no es un usuario real de ya.com.
   .
   QUIT
   ```

   Como puede verse existe una línea compuesta sólo por un punto. El mailer la usa para conocer dónde acaba el cuerpo del mensaje enviado.
4. Compruebe que el correo le ha sido entregado utilizando el MUA que más le guste.

Usando el mailer del ISP:

1. Nuestro primer paso consiste en averiguar el servidor de correo electrónico que es proporcionado por el ISP, es decir, el nombre de aquel host de Internet que se encargaría de enviar nuestros correos electrónicos cuando éstos son escritos en nuestro host. Para encontrar el mailer de la Universidad de Almería escribiremos:

   ```
   alumno$ nslookup -type=MX ual.es
   ```

   Entre otros mensajes que ahora no nos interesan aparecerá uno que dice *mail exchanger* igual a `smtp.ual.es`. Bien, este es el nombre del host que nos dejará conectarnos y enviar nuestro correo sin remitente real. Pudiera ocurrir que existen más de un *mail exchanger*. En este caso podremos utilizar cualquiera de ellos. El número entero que aparece junto a cada nombre indica la distancia en hops (esto puede comprobarse utilizando el programa `traceroute` si es que el host destino contesta). Cuanto menor es el número, mayor es la prioridad.
2. A continuación conectarse a `smtp.ual.es` al puerto 25 usando Telnet:

```
telnet smtp.ual.es 25
```

3. Introducir (ojo, el destinatario cambia respecto del ejemplo anterior):

```
HELO anonimo
MAIL FROM: <anonimo@ya.com>
RCPT TO: <nuestro_login_real@ual.es>
DATA
SUBJECT: Este es un correo con un remitente falso

Hola!
<anonimo@ya.com> no es un usuario real de ya.com.
.
QUIT
```

Cuestión 1: ¿Qué protocolo de transporte (TCP o UDP) utiliza el SMTP? ¿A la luz de los talleres realizados, podría este protocolo de transporte ser distinto al que utiliza el programa Telnet? Explíquese.

Cuestión 2: ¿Cómo podemos conocer qué mailers ha atravesado un correo electrónico? Envíe un correo desde su host virtual usando `mutt` a una dirección suya externa (en Hotmail, por ejemplo). ¿Cuántos mailers están involucrados? ¿Cuánto tiempo ha estado viajando el correo? Si no es posible acceder a la cabecera de los correos electrónicos en su servidor de correo, haga un reenvío (*forward*) del correo desde dicho servidor hasta otro donde sí tenga acceso a dicha cabecera (almanzora.ual.es, por ejemplo, donde puede usar el programa `mailx`).

Cuestión 3: Envíe un reply desde un servidor de correo externo hasta su cuenta en el host virtual. ¿A recibido el correo? Explique por qué ocurre esto.

Práctica 8

DNS (Domain Name Service)

El DNS es una de las aplicaciones fundamentales de Internet, con una función primordial: la de traducir los nombres de los hosts en direcciones IP. El servicio de nombres de dominio es una gigantesca base de datos distribuida a nivel mundial que funciona sin pausa, está constantemente actualizándose y resuelve las consultas en tiempo real (¿qué más se puede pedir?).

El DNS se basa en la arquitectura cliente-servidor. En su consulta más frecuente, los clientes piden resoluciones de nombres de dominio y los servidores las contestan indicando sus direcciones IP. La comunicación entre ambos se realiza a través del UDP.

En esta sesión práctica vamos a aprender a instalar un servidor DNS y a hacer uso del mismo. Esto genera varias ventajas. La primera y la más interesante es que las consultas son resueltas utilizando un nodo local (en nuestra sub-red), con lo que el tiempo de acceso se minimiza cuando este nodo almacena en su caché una resolución previa. La segunda es que descargamos de trabajo al resto de servidores de la jerarquía, cosa que algunos administradores nos agradecerán. También aprenderemos a realizar consultas de diversa índole con el objetivo de aprender qué posibilidades ofrece el sistema DNS.

8.1. Los nombres de dominio

Un dominio es, básicamente hablando, una organización cualquiera dentro de Internet que desea ser referenciada mediante un *nombre de dominio*. Por ejemplo, la Universidad de Almería es una organización que posee una serie de computadoras llamadas de la forma "`*.ual.es`". "`ual.es`" es el nombre del dominio de la Universidad de Almería. En la práctica para mucha gente es demasiado largo decir "nombre de dominio" y generalmente se dice simplemente "dominio".

8.2. Dominios y subdominios

Es frecuente que un nombre de dominio conste de varios subdominios, normalmente separados por un punto. Por ejemplo, el dominio "`ual.es`" es en realidad un subdominio del dominio "`es`".

Este hecho genera una **jerarquía de dominios** en la que cualquier nodo público de Internet puede ser localizado. Por ejemplo, el nodo `filabres.ual.es` es un host de la Universidad de Almería, que se trata de una universidad española.

8.3. La jerarquía de dominios

En cada dominio normalmente se instala un **servidor de nombres de dominio**, también llamado servidor DNS. Dicho servidor se encarga de mantener la base de datos de registros DNS para ese dominio. Por ejemplo, en la Universidad de Almería hay un servidor DNS que conoce todas las resoluciones de las direcciones IP para todos los hosts con nombres de la Universidad. En concreto, `filabres.ual.es` (`150.214.156.2`) es un servidor DNS del dominio `ual.es`. Otra forma de decir esto consiste en llamar a `filabres.ual.es` el servidor de nombres autorizado para el dominio `ual.es`.

8.4. El proceso de resolución

Cuando un servidor DNS no conoce una resolución, normalmente utiliza otro servidor DNS situado inmediatamente superior en la jerarquía de dominios para intentar resolverla. Esta política de **consultas recursivas** finalmente encuentra un servidor DNS que sí conoce la resolución. Entonces el registro DNS solicitado viaja desde dicho servidor DNS deshaciendo el camino hasta el servidor DNS al que inicialmente realizamos la consulta. En este proceso, cada servidor DNS actualiza su caché con esta información por un determinado periodo de tiempo.

Por ejemplo, cuando queremos acceder a `www.nasa.gov` desde un host de la Universidad de Almería, dicho host (generalmente[1]) debería preguntar a `filabres.ual.es`. Si `filabres` no almacena en su caché la resolución, pregunta al siguiente servidor DNS en la jerarquía que podría ser un nodo del CICA (Centro Informático Científico de Andalucía). Si éste fallara, preguntaría a un servidor de RedIRIS (la red académica y de investigación nacional), y así sucesivamente hasta llegar a un servidor DNS raíz. Si éste falla, entonces la consulta comienza a descender por la jerarquía de servidores DNS hasta llegar al responsable del dominio `nasa.gov` que necesariamente debe conocer la resolución.

Existe una versión diferente de este algoritmo recursivo que se realiza una **consulta iterativa**. La diferencia radica en que cuando un servidor DNS falla en lugar de hacer él la consulta le indica al cliente a qué otro servidor DNS puede preguntar. Este proceso, sin embargo, no actualiza las cachés de los servidores.

8.5. Instalación de un servidor DNS

Para convertir un host con Linux en un servidor DNS basta con instalar el programa BIND (Berkeley Internet Name Domain) (`http://www.isc.org/products/BIND`):

Debian Linux:

```
root# apt-get install bind9 bind9-doc dnsutils
```

Fedora Core Linux:

```
root# yum install bind bind-utils caching-nameserver system-config-bind
```

Gentoo Linux:

```
root# emerge bind bind-utils
```

8.6. Configuración del servidor DNS

El servidor puede configurarse para ofrecer servicio de tres formas diferentes:

Servidor de nombres autorizado del dominio X: Si almacena los registros de resolución para los hosts del dominio X.

Servidor de nombres réplica: Si mantiene una copia de los registros de otro servidor DNS.

Servidor sólo caché: Cuando no almacena ningún registro, excepto los adquiridos en las consultas.

La primera configuración es la que utilizaría el administrador de una nueva red que desea que sus hosts tengan nombres de dominio asignados. En dicha configuración se generarían los registros de resolución para cada uno de estos hosts y se almacenarían en el servidor de nombres autorizado. Además, el servidor DNS del ISP debería tener en cuenta el servidor DNS instalado. Por desgracia, esto último es imposible llevarlo a la práctica si no se adquiere legalmente un dominio, previo pago al correspondiente ISP.

Auque parezca que se esto se hace así por motivos económicos, en realidad no es así. Si este tema no estuviera controlado, un usuario malicioso puede inventarse un dominio con el objetivo de introducir "ruido" en el DNS, y hacer otras cosas peores como veremos más tarde. Por estos motivos, en esta ocasión no vamos a instalar un servidor DNS autorizado. Sin embargo, bastaría con modificar el fichero de configuración correspondiente y construir los registros DNS utilizando alguna herramienta como `system-config-bind` de Red Hat Linux.

[1] En realidad puede consultar a qualquier otro que se deje consultar, pero lo más lógico es usar filabres.

La segunda configuración sirve para aumentar la fiabilidad del DNS ya que permite replicar los registros de resolución. Así, si el servidor de nombres autorizado fallase, el otro pasaría a resolver las consultas. Por ejemplo, en la Universidad de Almería hay un servidor DNS autorizado para el dominio `ual.es` (`filabres.ual.es`, `150.214.156.2`) y otro réplica (`150.214.35.10`). Instalar un servidor DNS réplica plantea los mismos problemas de seguridad que instalar uno autorizado y además, una replicación sin control constituye un buen ataque por denegación de servicio. Por estos motivos, desistiremos de realizar dicha opción.

Finalmente, la tercera y última configuración, en la que instalamos un servidor DNS que funciona en realidad como un proxy DNS, sí que es posible en cualquier situación y tiene sentido si queremos reducir el tráfico a través del gateway de la red y el tiempo de resolución para las máquinas a las que servimos. Por suerte, esta configuración es la que por defecto se instala con BIND.

8.6.1. Configuración como servidor caché

Como hemos indicado anteriormente, en principio no es necesario hacer ninguna modificación en el fichero `named.conf` para conseguir que el servidor DNS funcione como un proxy DNS. Sin embargo, sí que vamos a hacer un pequeño cambio para que BIND utilice los servidores DNS más próximos cuando la caché falle.[2] Para evitar esto hay que modificar el fichero `named.conf` insertando el código:

```
forward first;
forwarders {
  150.214.156.2;
  150.214.35.10;
};
```

en su sección `options`. Veamos exactamente qué hacer en cada distribución:

Debian Linux: La instalación de bind crea el fichero `/etc/bind/named.conf` y otros dos ficheros asociados `/etc/bind/named.conf.local` y `/etc/bind/named.conf.options`. El código anterior debe insertarse en este último fichero.

Fedora Core Linux: El fichero que configura el servidor es `/etc/named.conf`. Para configurarlo como un servidor DNS cache ejecutar:

```
root# system-config-bind # Salir sin hacer ninguna modificación
```

A continuación insertar el código anterior en el fichero.

Gentoo Linux: La instalación de BIND crea el fichero `/etc/bind/named.conf` para que funcione como un DNS caché. Simplemente insertaremos el código anterior en este fichero.

Finalmente hay que modificar el fichero `/etc/resolv.conf` indicando que ahora es `localhost` el servidor DNS. Si tuviéramos más hosts en nuestra red local, dicha modificación debería realizarse en todos ellos.

Taller 8.1: Instale BIND y configurelo como servidor DNS caché.

Cuestión 1: En este punto, ¿están utilizando ya las aplicaciones el nuevo servidor DNS? Razone su respuesta.

8.7. Configuración del cliente

La única opción de configuración que permiten los clientes es la especificación del (o los) servidor(es) DNS. En Linux esta información está declarada en el fichero:

`/etc/resolv.conf`

Este es un fichero ASCII y puede ser editado como administrador.

Taller 8.2: Configure el DNS para que las aplicaciones de red hagan uso del servidor DNS que acaba de instalar.

Cuestión 2: Plantee un experimento que compruebe que las aplicaciones hacen uso correcto del servidor DNS que acaba de instalar.

[2] De lo contrario las resoluciones funcionan porque se consultan directamente a los servidores de nombres raíz con el consiguiente aumento de la latencia.

8.8. Ejemplos de consultas

8.8.1. ¿Cuáles son mis servidores DNS?

A la hora de determinar el servidor DNS que estamos utilizando existen dos alternativas:

1. Acceder a esta información en los ficheros de configuración correspondientes. Por ejemplo, en Linux hay que leer el fichero

   ```
   /etc/resolv.conf
   ```

   Evidentemente, esta opción sólo funciona bajo Linux.

   En este fichero aparecen al menos la IP de un servidor DNS. Al primero de ellos se le llama servidor DNS primario y debería ser un servidor DNS autorizado para el dominio en el que nos hayamos o un servidor DNS réplica de este, por motivos de eficiencia. El resto de direcciones IP son los llamados servidores secundarios porque, en el caso de fallar el primerio, se haría uso de estos.

2. Utilizar programas para chequear el funcionamiento del DNS como pueden ser `nslookup` y `dig`. `nslookup` funciona tanto desde la línea de comandos como de forma interactiva. `dig` sólo lo hace desde la línea de comandos. Además, téngase en cuenta que normalmente no se pregunta por el servidor de nombres autorizado de un host, sino por el servidor de nombres autorizado de un dominio (al que pertenece el host). Finalmente indicar que el resultado de la consulta no depende del host, ni de que éste pertenezca al dominio por el que estamos preguntando. El resultado sólo depende del dominio por el que preguntamos. Veamos algunos ejemplos:

   ```
   # Preguntamos a nuestro servidor DNS primario (o en su defecto, a un secundario) por el dominio
   alumno$ dig ual.es

   ; <<>> DiG 9.2.4 <<>> ual.es
   ;; global options:  printcmd
   ;; Got answer:
   ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65305
   ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

   ;; QUESTION SECTION:
   ;ual.es.                                IN      A

   ;; AUTHORITY SECTION:
   ual.es.                 172800  IN      SOA     filabres.ual.es.\
   postmaster.filabres.ual.es. 2006111502 16400 7200 2592000 172800

   ;; Query time: 0 msec
   ;; SERVER: 150.214.156.2#53(150.214.156.2)
   ;; WHEN: Thu Nov 30 10:26:33 2006
   ;; MSG SIZE  rcvd: 80
   ```

   En esta consulta, entre mucha otra información `dig` indica que el servior DNS autorizado para el dominio `ual.es` es `filabres.ual.es`.

Taller 8.3: Recree la(s) consulta(s) anterior(es).

Taller 8.4: Averigue el (o los) servidor(es) de nombres autorizado(s) para el dominio `google.es`.

Cuestión 3: ¿Qué comando ha ejecutado para determinar el (o los) servidor(es) de nombres autorizado(s) para el dominio `google.es`?

8.8.2. ¿Cuál es la dirección IP del host ...?

En la siguiente consulta preguntamos a nuestro servidores DNS primario (o en su defecto, secundario) por la dirección IP del host `www.google.es`:

```
# Preguntamos a nuestro servidor DNS por la IP de "www.google.es"
alumno$ dig www.google.es

; <<>> DiG 9.2.4 <<>> www.google.es
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46300
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 6, ADDITIONAL: 6

;; QUESTION SECTION:
;www.google.es.                 IN      A

;; ANSWER SECTION:
www.google.es.          167616  IN      CNAME   www.google.com.
www.google.com.         425476  IN      CNAME   www.l.google.com.
www.l.google.com.       201     IN      A       209.85.129.104
www.l.google.com.       201     IN      A       209.85.129.99
www.l.google.com.       201     IN      A       209.85.129.147

;; AUTHORITY SECTION:
l.google.com.           81028   IN      NS      a.l.google.com.
l.google.com.           81028   IN      NS      b.l.google.com.
l.google.com.           81028   IN      NS      c.l.google.com.
l.google.com.           81028   IN      NS      d.l.google.com.
l.google.com.           81028   IN      NS      e.l.google.com.
l.google.com.           81028   IN      NS      g.l.google.com.

;; ADDITIONAL SECTION:
a.l.google.com.         23928   IN      A       216.239.53.9
b.l.google.com.         23928   IN      A       64.233.179.9
c.l.google.com.         14956   IN      A       64.233.161.9
d.l.google.com.         81028   IN      A       64.233.183.9
e.l.google.com.         14956   IN      A       66.102.11.9
g.l.google.com.         21711   IN      A       64.233.167.9

;; Query time: 12 msec
;; SERVER: 150.214.156.2#53(150.214.156.2)
;; WHEN: Thu Dec 21 10:39:27 2006
;; MSG SIZE  rcvd: 319
```

Esta respuesta es un poco complicada por dos motivos: (1) `www.google.es` y `www.google.com` son la misma máquina cuando hacemos la consulta a nuestro servidor de nombres y (2), el servidor Web está replicado tres veces en las direcciones IP `209.85.129.104`, `209.85.129.147` y `209.85.129.99`. Esto puede comprobarse si preguntamos por `www.google.com`:

```
# Preguntamos a nuestro servidor DNS por la IP de "www.google.com"
alumno$ dig www.google.com

; <<>> DiG 9.2.4 <<>> www.google.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20533
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 6, ADDITIONAL: 6

;; QUESTION SECTION:
;www.google.com.                        IN      A
```

```
;; ANSWER SECTION:
www.google.com.         420014  IN      CNAME   www.l.google.com.
www.l.google.com.       207     IN      A       209.85.129.104
www.l.google.com.       207     IN      A       209.85.129.99
www.l.google.com.       207     IN      A       209.85.129.147

;; AUTHORITY SECTION:
l.google.com.           75566   IN      NS      a.l.google.com.
l.google.com.           75566   IN      NS      b.l.google.com.
l.google.com.           75566   IN      NS      c.l.google.com.
l.google.com.           75566   IN      NS      d.l.google.com.
l.google.com.           75566   IN      NS      e.l.google.com.
l.google.com.           75566   IN      NS      g.l.google.com.

;; ADDITIONAL SECTION:
a.l.google.com.         18466   IN      A       216.239.53.9
b.l.google.com.         18466   IN      A       64.233.179.9
c.l.google.com.         9494    IN      A       64.233.161.9
d.l.google.com.         75566   IN      A       64.233.183.9
e.l.google.com.         9494    IN      A       66.102.11.9
g.l.google.com.         16249   IN      A       64.233.167.9

;; Query time: 1 msec
;; SERVER: 150.214.156.2#53(150.214.156.2)
;; WHEN: Thu Dec 21 12:10:30 2006
;; MSG SIZE  rcvd: 292
```

Además, como puede verse `www.google.com` es un alias de `www.l.google.com`.

Taller 8.5: Recree la(s) consulta(s) anterior(es).

8.8.3. ¿Cuáles son los servidores de nombres del dominio ...?

```
# Preguntamos por el dominio "mit.edu"
alumno$ dig mit.edu
; <<>> DiG 9.2.4 <<>> mit.edu
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10644
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;mit.edu.                       IN      A

;; ANSWER SECTION:
mit.edu.                60      IN      A       18.7.22.69

;; AUTHORITY SECTION:
mit.edu.                14655   IN      NS      BITSY.mit.edu.
mit.edu.                14655   IN      NS      STRAWB.mit.edu.
mit.edu.                14655   IN      NS      W20NS.mit.edu.

;; ADDITIONAL SECTION:
BITSY.mit.edu.          14655   IN      A       18.72.0.3
STRAWB.mit.edu.         8735    IN      A       18.71.0.151
W20NS.mit.edu.          8735    IN      A       18.70.0.160

;; Query time: 158 msec
;; SERVER: 150.214.156.2#53(150.214.156.2)
```

```
;; WHEN: Thu Dec 21 12:20:46 2006
;; MSG SIZE  rcvd: 157
```

Podemos ver que existen tres servidores de nombres autorizados para el dominio "bit.edu".

Taller 8.6: Recree la(s) consulta(s) anterior(es).

Taller 8.7: Encuentre los servidores nombres autorizados de un dominio que conozca.

Cuestión 4: ¿A qué servidor DNS ha consultado?

8.8.4. ¿Cómo interrogamos a otro servidor DNS?

```
# Preguntamos al servidor DNS "bitsy.mit.edu" por el host "www.google.es"
alumno$ dig @bitsy.mit.edu www.google.es

; <<>> DiG 9.2.4 <<>> @bitsy.mit.edu www.google.es
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7735
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 7, ADDITIONAL: 7

;; QUESTION SECTION:
;www.google.es.                 IN      A

;; ANSWER SECTION:
www.google.es.          304614  IN      CNAME   www.google.com.
www.google.com.         557306  IN      CNAME   www.l.google.com.
www.l.google.com.       174     IN      A       64.233.161.104
www.l.google.com.       174     IN      A       64.233.161.99
www.l.google.com.       174     IN      A       64.233.161.147

;; AUTHORITY SECTION:
l.google.com.           72376   IN      NS      c.l.google.com.
l.google.com.           72376   IN      NS      f.l.google.com.
l.google.com.           72376   IN      NS      d.l.google.com.
l.google.com.           72376   IN      NS      b.l.google.com.
l.google.com.           72376   IN      NS      e.l.google.com.
l.google.com.           72376   IN      NS      g.l.google.com.
l.google.com.           72376   IN      NS      a.l.google.com.

;; ADDITIONAL SECTION:
c.l.google.com.         40415   IN      A       64.233.161.9
f.l.google.com.         72376   IN      A       72.14.235.9
d.l.google.com.         38903   IN      A       64.233.183.9
b.l.google.com.         38903   IN      A       64.233.179.9
e.l.google.com.         38903   IN      A       66.102.11.9
g.l.google.com.         44316   IN      A       64.233.167.9
a.l.google.com.         39459   IN      A       216.239.53.9

;; Query time: 231 msec
;; SERVER: 18.72.0.3#53(bitsy.mit.edu)
;; WHEN: Thu Dec 21 12:15:01 2006
;; MSG SIZE  rcvd: 351
```

Ahora preguntamos al servidor DNS "bitsy.mit.edu" por el host "www.google.com":

```
# Preguntamos al servidor DNS "bitsy.mit.edu" por el host "www.google.com"
alumno$ dig @bitsy.mit.edu www.google.com
```

```
; <<>> DiG 9.2.4 <<>> @bitsy.mit.edu www.google.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29537
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 7, ADDITIONAL: 7

;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         557281  IN      CNAME   www.l.google.com.
www.l.google.com.       149     IN      A       64.233.161.99
www.l.google.com.       149     IN      A       64.233.161.147
www.l.google.com.       149     IN      A       64.233.161.104

;; AUTHORITY SECTION:
l.google.com.           72351   IN      NS      c.l.google.com.
l.google.com.           72351   IN      NS      f.l.google.com.
l.google.com.           72351   IN      NS      d.l.google.com.
l.google.com.           72351   IN      NS      b.l.google.com.
l.google.com.           72351   IN      NS      e.l.google.com.
l.google.com.           72351   IN      NS      g.l.google.com.
l.google.com.           72351   IN      NS      a.l.google.com.

;; ADDITIONAL SECTION:
c.l.google.com.         40390   IN      A       64.233.161.9
f.l.google.com.         72351   IN      A       72.14.235.9
d.l.google.com.         38878   IN      A       64.233.183.9
b.l.google.com.         38878   IN      A       64.233.179.9
e.l.google.com.         38878   IN      A       66.102.11.9
g.l.google.com.         44291   IN      A       64.233.167.9
a.l.google.com.         39434   IN      A       216.239.53.9

;; Query time: 156 msec
;; SERVER: 18.72.0.3#53(bitsy.mit.edu)
;; WHEN: Thu Dec 21 12:15:25 2006
;; MSG SIZE  rcvd: 324
```

Taller 8.8: Recree la(s) consulta(s) anterior(es).

Como podemos ver, ambas respuestas son idénticas (como en el caso de preguntar a `filabres`), pero diferentes comparadas con las que devuelve nuestro servidor de nombres. Esto significa que, si nos conectamos a `www.google.es` desde el MIT, veremos la versión americana de google.

Finalmente, nótese que el servidor DNS no devuelve las direcciones IP de las réplicas del servidor Web siempre en el mismo orden. Esto se utiliza para distribuir la carga.

Taller 8.9: Encuentre los servidores nombres autorizados para el dominio "`ual.es`" interrogando al servidor DNS "bitsy.mit.edu" (o a otro que conozca).

Cuestión 5: ¿Qué comando y con qué argumento ha usado?

Cuestión 6: Si ejecuta el comando `dig @filabres.ual.es ual.es`, ¿está usando el servidor DNS especificado en "/etc/resolv.conf"?

8.8.5. Averiguando la jerarquía de servidores DNS

Como hemos comentado anteriormente, cuando consultamos a un servidor DNS sobre una dirección IP para la cual él no es el servidor DNS autorizado lo más frecuente es que éste tenga que consultar a otro servidor DNS (consulta recursiva) o nos indique a qué otro servidor DNS podemos preguntar nosotros (consulta iterativa).

Activando el flag +trace de dig podemos conocer qué servidores DNS se han consultado. En el siguiente ejemplo preguntamos a bitsy.mit.edu por la dirección IP del host gogh.ace.ual.es:

```
alumno$ dig +trace @bitsy.mit.edu gogh.ace.ual.es

; <<>> DiG 9.2.4 <<>> +trace @bitsy.mit.edu gogh.ace.ual.es
;; global options:  printcmd
.                       488373  IN      NS      a.root-servers.net.
.                       488373  IN      NS      h.root-servers.net.
.                       488373  IN      NS      c.root-servers.net.
.                       488373  IN      NS      g.root-servers.net.
.                       488373  IN      NS      f.root-servers.net.
.                       488373  IN      NS      b.root-servers.net.
.                       488373  IN      NS      j.root-servers.net.
.                       488373  IN      NS      k.root-servers.net.
.                       488373  IN      NS      l.root-servers.net.
.                       488373  IN      NS      m.root-servers.net.
.                       488373  IN      NS      i.root-servers.net.
.                       488373  IN      NS      e.root-servers.net.
.                       488373  IN      NS      d.root-servers.net.
;; Received 436 bytes from 18.72.0.3#53(bitsy.mit.edu) in 158 ms

es.                     172800  IN      NS      NS3.NIC.FR.
es.                     172800  IN      NS      SUN.REDIRIS.es.
es.                     172800  IN      NS      SUNIC.SUNET.SE.
es.                     172800  IN      NS      NS.UU.NET.
es.                     172800  IN      NS      NS1.NIC.es.
es.                     172800  IN      NS      AUNIC.AUNIC.NET.
es.                     172800  IN      NS      NS1.CESCA.es.
es.                     172800  IN      NS      NS2.NIC.es.
;; Received 352 bytes from 198.41.0.4#53(a.root-servers.net) in 134 ms

ual.es.                 7200    IN      NS      chico.rediris.es.
ual.es.                 7200    IN      NS      alboran.ual.es.
ual.es.                 7200    IN      NS      filabres.ual.es.
ual.es.                 7200    IN      NS      sun.rediris.es.
ual.es.                 7200    IN      NS      dns1.cica.es.
ual.es.                 7200    IN      NS      dns2.cica.es.
;; Received 263 bytes from 192.134.0.49#53(NS3.NIC.FR) in 49 ms

ace.ual.es.             172800  IN      NS      filabres.ual.es.
ace.ual.es.             172800  IN      NS      alboran.ual.es.
;; Received 110 bytes from 130.206.1.3#53(chico.rediris.es) in 37 ms

gogh.ace.ual.es.        172800  IN      A       193.147.118.57
ace.ual.es.             172800  IN      NS      filabres.ual.es.
ace.ual.es.             172800  IN      NS      alboran.ual.es.
;; Received 126 bytes from 150.214.156.2#53(filabres.ual.es) in 0 ms
```

Taller 8.10: Recree la(s) consulta(s) anterior(es).

Como podemos ver, bitsy.mit.edu consulta (en la versión recursiva) al servidor de nombres raíz a.root-servers.net, que consulta a NS3.NIC.FR, que consulta a chico.rediris.es, que consulta a filabres.ual.es.

Taller 8.11: Determine qué servidor(es) DNS de nivel más bajo tienen en común los servidores DNS de la Universidad de Almería y los de la Universidad de Córdoba.

Cuestión 7: ¿Qué comando(s) permite(n) encontrar qué servidor(es) DNS de nivel más bajo tienen en común los servidores DNS de la Universidad de Almería y los de la Universidad de Córdoba? ¿Cuáles son dichos servidores?

8.8.6. Resolución inversa

Finalmente, también podemos interrogar al servidor DNS por el nombre de un host a partir de su dirección IP:

```
# Preguntamos al servidor DNS por el nombre del host que tiene la IP
# 193.147.118.57
alumno$ dig -x 193.147.118.57

; <<>> DiG 9.2.4 <<>> -x 193.147.118.57
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44233
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 6, ADDITIONAL: 7

;; QUESTION SECTION:
;57.118.147.193.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
57.118.147.193.in-addr.arpa. 172800 IN  PTR     gogh.ace.ual.es.

;; AUTHORITY SECTION:
118.147.193.in-addr.arpa. 172800 IN     NS      filabres.ual.es.
118.147.193.in-addr.arpa. 172800 IN     NS      alboran.ual.es.
118.147.193.in-addr.arpa. 172800 IN     NS      dns1.cica.es.
118.147.193.in-addr.arpa. 172800 IN     NS      dns2.cica.es.
118.147.193.in-addr.arpa. 172800 IN     NS      sun.rediris.es.
118.147.193.in-addr.arpa. 172800 IN     NS      chico.rediris.es.

;; ADDITIONAL SECTION:
filabres.ual.es.        172800  IN      A       150.214.156.2
alboran.ual.es.         172800  IN      A       150.214.156.32
dns1.cica.es.           163357  IN      A       150.214.5.83
dns2.cica.es.           3265    IN      A       150.214.4.35
sun.rediris.es.         15046   IN      A       130.206.1.2
chico.rediris.es.       15295   IN      A       130.206.1.3
dns2.cica.es.           127925  IN      AAAA    2001:720:c10:9::4

;; Query time: 1 msec
;; SERVER: 150.214.156.2#53(150.214.156.2)
;; WHEN: Thu Dec 21 13:19:57 2006
;; MSG SIZE  rcvd: 332
```

Como podemos ver, el host es `gogh.ace.ual.es`.

Taller 8.12: Recree la(s) consulta(s) anterior(es).

Taller 8.13: Existen servidores Web que prestan este mismo servicio. Ejemplos:

`http://remote.12dt.com/`
`http://www.zoneedit.com/lookup.html`

Utilice algunas de estas páginas Web para comprobar que el resultado coincide con el que devuelve `dig` para una determinada consulta.

8.9. Cuidado con el DNS

El DNS es uno de los servicios más críticos que existen en Internet. A continuación mostramos algunos de los riesgos más importantes a los que nos exponemos cuando utilizamos un servidor DNS inseguro (más información en BULMA (`http://bulma.net/body.phtml?nIdNoticia=1334`).

1. Si definimos un dominio e instalamos un servidor DNS autorizado para el mismo, estamos obligados a ofrecer información sobre el dominio definido. Esto significa que cualquier usuario de Internet puede conocer qué máquinas y con qué direcciones IP existen en nuestro dominio.

2. Los servidores DNS que delegan dominios a otros servidores (es decir, que ya no son servidores autorizados para ese sub-dominio) están obligados a escuchar las modificaciones que en dichos dominios se producen (en caso contrario, el DNS no escalaría). Si no tenemos cuidado cuando configuramos nuestro servidor y controlamos adecuadamente "la transferencia de dominio", un hacker puede instalar un servidor DNS y hacerlo pasar por el servidor DNS autorizado de ese dominio. Si esto ocurre, puede inyectar información falsa en el sistema DNS y hacer que cuando accedamos a un host en realidad entremos en otro. Imagine lo que sería acceder a nuestra cuenta bancaria a través del host del hacker, creyendo que estamos enviando los datos al servidor de nuestro banco cuando en realidad lo estamos haciendo a un host que controla el hacker.

8.10. DNS + DHCP

Cuando utilizamos el DHCP para gestionar redes públicas en las que las direcciones IP asignadas son dinámicas, podemos configurar el servidor DHCP para que avise al servidor DNS autorizado de ese dominio de los cambios que se vayan produciendo en las asignaciones de las direcciones.

Este sistema tiene una gran ventaja: se gestiona sólo. Sin embargo, no es muy recomendable su uso cuando vamos a instalar servicios importantes en los hosts del dominio. Supongamos que en uno de estos host tenemos un servidor Web que tiene miles de accesos diarios. En esta situación la dirección IP de este host está diseminada por las cachés de miles de servidores DNS de toda la Internet. Si dicho host recibe una nueva IP, muchas resoluciones serían incorrectas hasta que las cachés son actualizadas.

Cuestión 8: ¿Cómo cree que está configurado el servidor DNS que especifica el DHCP para su host virtual, como autorizado, réplica, primario, secundario o caché? Exponga sus razones.

Taller 8.14: Configure el sistema operativo huésped (Windows, por ejemplo) para que utilice el servidor DNS que ha instalado en el host virtual (Debian). Construya un experimento que permita capturar la interacción DNS en la una aplicación ejecutada en el sistema operativo huesped consulta al DNS en el host virtual. Responda a las siguientes cuestiones (muestre los resultados que le han permitido encontrar las soluciones a las cuestiones):

Cuestión 9: ¿Qué protocolo de transporte se está usando cuando utiliza el DNS?

Cuestión 10: ¿A qué dirección IP y puerto se ha enviado la consulta?

Cuestión 11: ¿Desde qué dirección IP y puerto se ha efectuado la consulta?

Cuestión 12: ¿Qué tipo de petición DNS se ha generado?

Práctica 9

Acceso remoto 2: SSH

SSH (Secure SHell) [14] es un programa semejante al programa Telnet, pero que a diferencia de éste, SSH cifra toda la comunicación entre el cliente y el servidor. Para ello se vale de un algoritmo de cifrado de clave pública. SSH es muy utilizado para el acceso remoto a hosts, permitiendo a los usuarios trabajar como si estuvieran físicamente sentados frente el teclado del host remoto.

En esta práctica aprenderemos a usar el conjunto de utilidades que vienen con el paquete SSH.

La implementación del SSH que vamos a utilizar en este módulo se llama OpenSSH (`http://www.openssh.com`) e incorpora, entre otras, las siguientes utilidades:

`ssh`**:** Un cliente. El sustituto de Telnet.

`sshd`**:** Un servidor.

`scp`**:** (Secure CoPy). Una utilidad semejante a la utilidad `rcp` que permite copiar ficheros entre hosts remotos de forma segura.

`sftp`**:** (Secure FTP). Una versión segura del programa Ftp.

9.1. Algoritmos de cifrado

Los algoritmos de cifrado (o criptográficos) se utilizan para esconder la información a aquel conjunto de personas que no deberían tener acceso a ella. La idea es muy simple y consiste en alterar la representación de la información usando una *clave* (o *key*). Así, sólo quien conozca la clave es capáz de restaurar la representación original (descrifrada) de la información.

Existen dos tipos distintos de algoritmo de cifrado, dependiendo del número de claves que se utilizan:

Algoritmos de clave simétrica: También se llaman algoritmos de clave secreta y se caracterizan por usar la misma clave para cifrar y descifrar. Así, si enviamos un e-mail cifrado a alguien también tenemos que indicarle la clave que usamos para cifrarlo.

Algoritmos de clave asimétrica: Los sistemas de clave simétrica presuponen que el receptor conoce la clave secreta, pero, ¿cómo le hacemos llegar dicha clave sólo a él a través de un canal inseguro como puede ser Internet? Usando un algoritmo de clave simétrica es imposible.

Para evitar este problema se idearon los algoritmos de clave asimétrica (o algoritmos de clave pública). En estos algoritmos cada extremo de la comunicación maneja dos claves, una pública y otra privada (en total, 4 claves).[1] Se cumple que: (1) lo que se cifra utilizando la clave pública puede descifrarse, sólo, utilizando la clave privada y (2) es imposible derivar la clave privada a partir de la pública (sin información adicional).

Así, cuando dos personas desean intercambiarse información primero deben intercambiar sus claves públicas. Habiendo hecho esto, cuando yo (por ejemplo) deseo enviar un mensaje a alguien lo cifro usando la clave pública de mi interlocutor. Dicho mensaje puede ser interceptado por una tercera persona, pero sólo mi interlocutor va a ser capaz de descifrarlo porque es la única persona que posee la clave privada que puede descifrarlo. De la misma manera, sólo yo podré descifrar lo que mi interlocutor me envía porque sólo yo poseo la clave privada que permite descifrar el mensaje.

[1] El emisor tiene una clave pública y otra privada, y el receptor una clave pública y otra privada.

Cuestión 1: Imagine que está utilizando un sistema de cifrado de clave asimétrica para comunicarse con otra persona. ¿Con qué clave descifrará un mensaje que ella le ha enviado? ¿Con qué clave cifrará le mensaje que usted quiere transmitirle?

9.2. Características del SSH

El SSH utiliza tres mecanismos (tres protocolos) [14]:

Transport Layer Protocol (TLP) [16]: Autentifica el servidor, cifra la comunicación y conserva la integridad de la información transmitida. Suele usar el TCP y adicionalmente puede comprimir los datos.

User Authentication Protocol (UAP) [15]: Utiliza el TLP y autentifica el cliente al servidor.

Connection Protocol (CP) [17]: Se ejecuta sobre el UAP y permite multiplexar muchos canales lógicos sobre un único tunel cifrado.

Existen dos versiones de SSH: la versión 1 y la versión 2. La segunda es la más evolucionada y utilizada. Ambas versiones están basadas en un algoritmo de cifrado de clave pública. SSH permite además comprimir el stream de datos al vuelo (on-the-fly). Con esto ganaremos en velocidad efectiva de tranmisión y aumentaremos la seguridad.

SSH utiliza DES (Data Encryption Standard) en su versión 1 y además AES (Advanced Encryption Scheme) y Blowfish en su versión 2. DES es considerado el menos seguro y AES el más seguro. Debe tenerse en cuenta que el consumo de CPU será proporcional al nivel de seguridad.

Quizás la parte más importante de SSH radica en que posee un sistema para autentificar a los extremos de la comunicación (evita el *spoofing*). De esta manera, se asegura que tanto el emisor como el receptor son realmente quien dicen ser. [2]

A continuación se presenta todo el proceso de comunicación (`http://es.tldp.org/Tutoriales/doc-ssh-intro/intro-ssh/node9.html`) entre un cliente y un servidor (versión 2 del protocolo):

1. Fase de conexión TCP:
   a) El cliente establece una conexión TCP con el servidor.
2. Fase de identificación del protocolo:
   a) Si el servidor acepta la conexión TCP envía al cliente un mensaje (no cifrado) que indica los parámetros fundamentales de la conexión (como por ejemplo, el puerto que utilizará para el resto de la sesión y la versión del protocolo SSH).
   b) El cliente envía al servidor otro mensaje con su propia información acerca de los parámetros que desea utilizar en la comunicación.
3. Fase de autentificación del servidor:
   a) Si el servidor acepta las condiciones del cliente, el servidor envía su *host key* pública, sin cifrar, y un número aleatorio, sin cifrar.
   b) El cliente comprobará que la *host key* recibida es igual a la que recibió en la última conexión con dicho servidor. Si es la primera vez que se conecta, mostrará la *host key* al usuario para que pueda, por ejemplo, telefonear al administrador del host remoto para preguntarle la *host key*. Si no es la primera vez y es diferente (el cliente mira en `/.ssh/known_hosts`), también mostrará la *host key* para evitar el posible spoofing.
   c) Si el cliente continua con la conexión, cifrará y almacenará la *host key* en `/.ssh/known_hosts` para futuras comprobaciones.
4. Generación de la clave de sesión:
   a) El cliente genera una clave de cifrado simétrica llamada *session key* y la cifra utilizando la *host key* pública del servidor.
   b) El cliente envía la *session key* cifrada al servidor.
   c) El servidor descifra la *session key* utilizando su *host key* privada. En este punto ambos extremos conocen la clave de sesión que se utiliza para cifrar y descifrar el resto de la comunicación.

[2] Nótese que esto no se puede verificar simplemente con el algoritmo de cifrado.

5. Fase de identificación y autentificación del usuario (`http://www.ssh.fi/support/documentation/online/ssh/adminguide/32/User_Authentication.html`). Se intentarán secuencialmente cada uno de los siguientes métodos[3]:

   Autentificación basada en el host del usuario: El usuario queda autentificado en estos casos:

   a) Si el host de usuario está en `/etc/hosts.equiv` o en `/etc/ssh/shosts.equiv`, y si el nombre del usuario es el mismo en la máquina local y en la máquina remota.

   b) Si los ficheros `/.rhosts` o `/.shosts` en el directorio *home* del usuario en la máquina remota contiene una entrada de la forma `ususario_maquina_cliente@maquina_cliente`.

   En cualquiera de estos casos es necesario que el fichero `/etc/ssh/ssh_known_hosts` o el fichero `/.ssh/known_hosts` contenga el *host key* pública del cliente.

   Autentificación de clave pública: Para poder usar esta forma de autentificación, el usuario en alguna sesión previa ha debido generar una clave pública y otra privada (generalmente usando RSA). En dicha sesión previa, el usuario ha informado al servidor de su clave de usuario pública. Durante esta fase de autentificación el servidor genera un número aleatorio (llamado *challenge*) que es cifrado usando la clave pública que le envió el usuario. El cliente recibe el desafío, lo descifra usando su clave privada, y lo vuelve a cifrar usando su clave privada. Finalmente se lo envía al servidor. El servidor lo descifra usando la clave pública del usuario y si el número coindice, entonces el usuario queda autentificado.

   Autentificación basada en password: Consiste en enviar un password que sólo el usuario conoce (generalmente el password usado en la cuenta de la máquina remota). Dicho password viaja cifrado con la clave de sesión.

6. Fase de acceso al sistema remoto:

   a) El host remoto ejecuta un shell cuya entrada y salida es proporcionada por el servidor SSH.

7. Fase de desconexión:

   a) Cuando el usuario realiza un `logout`, el shell en el host remoto muere y la conexión TCP se cierra.

9.3. Instalación de SSH (cliente y servidor)

En Linux tenemos disponibles un servidor y un cliente SSH a través del paquete `OpenSSH` (`http://www.openssh.com`). Su instalación es muy sencilla:

Debian Linux:

```
root# apt-get install ssh
```

El demonio se llama `ssh`.

Fedora Core Linux:

```
root# yum install openssh
```

El demonio se llama `sshd`.

Gentoo Linux:

```
root# emerge openssh
```

El demonio se llama `sshd`.

[3] Esta lista no es exhaustiva, pueden existir otras formas de autentificación.

9.4. Configuración del servidor

El servidor SSH se configura modificando el fichero de configuración (ojo con la `d`):

`/etc/ssh/sshd_config`

Dicho fichero está codificado en ASCII y suficientemente autocomentado. A continuación comentaremos las diferentes opciones (más información en `man sshd_config`):

`Port`: Puerto de escucha del servicio. 22 por defecto.

`ListenAddress`: Direcciones que serán atendidas. Todas por defecto.

`Protocol`: Versión del protocolo. 2 por defecto.

`HostKey`: Especifica los ficheros con las host keys. `/etc/ssh/ssh_host_rsa_key` y `/etc/ssh/ssh_host_dsa_key`, por defecto.

`UsePrivilegeSeparation`: Especifica si el fichero `/.ssh/environment` y las opciones `environment=` en el fichero `/.ssh/authorized_keys` son procesadas por `sshd`. Por defecto, no.

`KeyRegenerationInterval`: Periodo de regeneración del server key para la versión 1 del protocolo. Por defecto, 1 hora.

`ServerKeyBits`: Número de bits de la server key para la versión 1 del protocolo. Por defecto 768 bits.

`SyslogFacility`: Tipo de registro de actividades. Por defecto `AUTH`.

`LogLevel`: Nivel de registro de actividades. Por defecto, `INFO`.

`LoginGraceTime`: Tiempo de espera para la autentificación. 120 segundos, por defecto.

`PermitRootLogin`: ¿Se permite acceder al root? Por defecto, sí.

`StrictModes`: Enviar ficheros con todos los permisos a todos los usuarios. Sí, por defecto.

`RSAAuthentication`: ¿Queremos usar RSA (versión 1)? Sí, por defecto.

`PubkeyAuthentication`: ¿Queremos usar clave pública (versión 2)?. Sí, por defecto.

`AuthorizedKeysFile`: Fichero con las claves para la autentificación. `/.ssh/authorized_keys`, por defecto.

`IgnoreRhosts`: ¿Se ignoran los ficheros `/.rhostsyt` `/.shosts`? Sí, por defecto.

`RhostsRSAAuthentication`: Permitir la autentificación por `/etc/rhosts` (versión 1). Por defecto, no.

`HostbasedAuthentication`: Permitir la autentificación por `/etc/rhosts` (versión 2). Por defecto, no.

`IgnoreUserKnownHosts`: Poner a `yes` si no confiamos en el fichero `/.ssh/known_hosts` para la `RhostsRSAAuthentication`.

`PermitEmptyPasswords`: No, por defecto (y no se recomienda poner `yes`".

`ChallengeResponseAuthentication`: Habilita los passwords "challenge-response". Por defecto, no.

`PasswordAuthentication`: Permite deshabilitar los passwords en texto plano cuando se está utilizando tunneling. No, por defecto (los passwords están cifrados, por defecto).

`KerberosAuthentication`: Usar Kerberos para autentificar los usuarios. Por defecto, no.

`KerberosOrLocalPasswd`: Opción "OrLocalPasswd" cuando utilizamos Kerberos. Por defecto, sí.

`KerberosTicketCleanup`: Opción "TicketCleanup" cuando utilizamos Kerberos. Por defecto, sí.

`KerberosGetAFSToken`: Opción "GetAFSToken" cuando utilizamos Kerberos. Por defecto, no.

`GSSAPIAuthentication`: Usar GSSAPI para autentificar los usuarios. Por defecto, no.

`GSSAPICleanupCredentials`: Opción "CleanupCredentials" cuando utilizamos Kerberos. Por defecto, sí.

`X11Forwarding`: Redirigir la conexión TCP usada por el servidor X-Window. Sí, por defecto. Así, cuando accedamos al host remoto desde una consola gráfica (un xterm, por ejemplo) y ejecutemos una aplicación gráfica, ésta aparecerá en nuestro sistema de ventanas.

`X11DisplayOffset`: Offset por defecto para la variable de entorno `DISPLAY` que controla el display gráfico utilizado por las aplicaciones gráficas.[4] Por defecto, 10.

`PrintMotd`: El fichero `/etc/motd` es un mensaje de bienvenida codificado en ASCII y que es mostrado cuando accedemos a través del terminal remoto. Por defecto, no usar.

`PrintLastLog`: Imprimir la salida más reciente del comando de `lastlog` que se utiliza para saber desde dónde nos conectamos al servidor la última vez que utilizamos SSH. Por defecto está a `yes`.

`TCPKeepAlive`: Enviar periódicamente mensajes "TCP keepalive". De esta manera si la conexión falla los extremos son notificados y el terminal no se "cuelga". Por defecto, sí.

`UseLogin`: ¿Usar la utilidad `login` para la autentificación?. Por defecto, no.

`MaxStartups`: Espefifica el número máximo de conexxiones concurrentes sin autentificar que serán permitidas. Por defecto, 10.

`Banner`: El banner es un mensaje que puede enviarse antes de iniciarse la conexión con el servidor (para avisar, por ejemplo, que se está accediendo a un sitio muy chungo :-). Esta opción indica la localización del fichero ASCII que contiene dicho mensaje.

`AcceptEnv`: Espeficica qué variables para la configuración local del lenguaje van a ser aceptadas. Por defecto, todas.

`Subsystem`: Permite especificar la localización en el sistema de ficheros de una determinada utilidad a usar con SSH. Normalmente se utiliza para indicar el fichero que contiene ejecuta la utilidad `sftp`.

`UsePAM`: Autentificar al usuario utilizando el Pluggable Authentication Module. Por defecto, sí.

Tras modificar el fichero de configuración del servidor SSH es necesario relanzar el servicio (véase el Apéndice C).

9.5. Configuración del cliente

El cliente SSH (`ssh`) utiliza el fichero de configuración:

`/etc/ssh/ssh_config`

y demás, el fichero:

`~/.ssh/config`

si es que existe. Los parámetros especificados en él prevalecen sobre el fichero `ssh_config`. La información completa sobre este fichero puede obtenerse mediante `man ssh_config`.

Veámos las principales opciones que podemos configurar:

`Host`: Especifica el host al que se aplican los parámetros que aparecen a continuación. Un "*" (que es el valor por defecto) referencia a todos los hosts. Nótese que el los sistemas en los que comparte el directorio "`/etc`" esto puede ser muy útil.

`ForwardAgent`: Indica si la conexión con el agente de autentificación (si es que existe) será redirigido a la máquina remota. Por defecto, no.

`ForwardX11`: Se utiliza para hacer que las conexiones X11 sean automáticamente redirigidas sobre el canal seguro (utilizando la variable de entorno `DISPLAY`). Por defecto es que no.

`ForwardX11Trusted`: Especifica si el cliente X11 tendrá acceso sin restricciones al servidor. Por defecto es que sí tendrá acceso total.

[4] En el sistema X11 un servidor X11 puede tener varios clientes X11 (varios terminales gráficos). La variable `DISPLAY` permite seleccionar el cliente.

`RhostsRSAAuthentication`: Especifica si usar el fichero `rhosts` con RSA (ver configuración del servidor). Por defecto, no.

`RSAAuthentication`: Especifica si usar RSA authentication (ver configuración del servidor). Por defecto, sí.

`PasswordAuthentication`: Indica si usar autentificación basada en password. Por defecto, sí.

`HostbasedAuthentication`: Especifica si usar la autentificación basada en `rhosts` con la autentificación de clave pública. Por defecto, no.

`BatchMode`: Permite deshabilitar la pregunta sobre el password. Por defecto, no.

`CheckHostIP`: Por defecto está habilitada. Si se deshabilita el cliente SSH no comprobará si el servidor figura en el fichero `/etc/ssh/ssh_known_hosts`. Esto permite detectar el DNS spoofing (el que alguien inserte en el DNS una IP falsa para un host).

`AddressFamily`: Tipos de direcciones IP aceptadas. Por defecto `any`.

`ConnectTimeout`: El TCP utiliza un temporizador para mantener activa la conexión enviando un paquete de datos vacío, aunque no exista actividad. Con esta opción hacemos que sea el SSH el que se encarge de este proceso, en lugar del TCP. Un valor a 0 indica que esta opción seguirá siendo controlada por el TCP.

`StrictHostKeyChecking`: Si este flag se activa a `yes`, el cliente SSH no añadirá el key host al fichero `/.ssh/known_hosts`, y por tanto, recharará la conexión con aquellos hosts que hallan cambiado su host key (por ejemplo, porque han reinstalado el sistema operativo).

`IdentityFile`: Especifica el fichero con el identificador DSA. En la versión 1 del protocolo apunta al fichero `/.ssh/identity`. En la versión 2 puede apuntar al fichero `/.ssh/id_rsa` and `/.ssh/id_dsa`, dependiendo del algoritmo de cifrado usado.

`Port`: Puerto donde espera encontrar al servidor. 22, por defecto.

`Protocol`: Versión del protocolo. 2, por defecto.

`Cipher`: Especifica el cifrador en la versión 1 del protocolo. Por defecto, `3des`.

`Ciphers`: Cifradores para la versión 2, por orden de preferencia. Por defecto, `aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes2 56-cbc`.

`EscapeChar`: Por defecto vale el carácter `~` (`<Alt Gr> + <4>`) y se utiliza para realizar acciones como terminar la conexión (`~.`), enviar un BREAK al sistema remoto (`~B`), acceder al interprete de comandos del cliente (`~C`), mostrar las conexiones redirigidas (`~#`), etc.

`Tunnel`: Esta opción por defecto vale `no` y se utiliza para evitar tener que utilizar la capa de red del TCP/IP. Los otros posibles valores son: `yes`, `point-to-point` y `ethernet`.

`TunnelDevice`: Selecciona el interface de red sobre el que realizar el tunneling. Por defecto, `any:any`.

`PermitLocalCommand`: Controla la posibilidad de ejecutar comandos en la máquina local (no en la remota) usando la secuencia de escape `C`. Por defecto, no.

`SendEnv`: Sirve para especificar el contenido de una variable de entorno (generalmente `LANG` que controla el lenguaje utilizado en el shell).

`HashKnownHosts`: Se usa para aplicar un hashing sobre los nombres de host y direcciones que se especifican en `/.ssh/known_hosts`. No, por defecto.

`GSSAPIAuthentication`: Permitir autentificación GSSAPI. Por defecto, no.

9.6. Uso de SSH

SSH es uno de los programas con los que tendremos que lidiar todos los días que estamos trabajando en una máquina con Unix o Linux. Veámos algunos ejemplos de uso típicos.

9.6.1. Accediendo a un host remoto

Para acceder al host remoto utilizamos el cliente `ssh`. En ese momento se producirá el proceso de autentificación del servidor y el proceso de identificación y autentificación del usuario. Dependiendo del esquema de autentificación de usuario existen distintas alternativas:

Autentificación basada en password: Suele ser la forma más corriente de acceder al host remoto:

```
alumno$ ssh host_remoto
```

Con este comando estamos accediendo al host utilizando el usuario `alumno`. Cuando queramos acceder con otro nombre de usuario utilizaremos:

```
alumno$ ssh otro_nombre_de_usuario@host_remoto
```

Taller 9.1: Instale un servidor SSH. Acceda a su cuenta en su host usando SSH. Pruebe también a hacerlo como administrador.

Autentificación basada en clave pública: Como indicamos en la Sección 9.2, el usuario se autentifica probando que es capaz de firmar (descifrar y cifrar) el desafío que le plantea el servidor. Para poder hacer esto son necesarios los siguientes pasos:

1. Generar las claves de autentificación:

```
# Por curiosidad mostramos el contenido del directorio "~/.ssh".
alumno$ ls -l $HOME/.ssh/
total 4
-rw-r--r-- 1 alumno alumno 1866 2007-02-08 15:12 known_hosts
v
# Generación de la clave usando RSA.
alumno$ ssh-keygen -t rsa
# Selecionar el fichero de salida que se especifica por defecto.
# La clave debería tener al menos 20 caracteres.

# Por curiosidad mostramos el contenido del directorio "~/.ssh".
alumno$ ls -l $HOME/.ssh/
total 12
-rw------- 1 alumno alumno 1743 2007-02-08 22:19 id_rsa
-rw-r--r-- 1 alumno alumno  405 2007-02-08 22:19 id_rsa.pub
-rw-r--r-- 1 alumno alumno 1866 2007-02-08 15:12 known_hosts

# Comprobamos que la clave pública se ha generado.
alumno$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA5HgtUR6OfvEIUv6m2xXM6QK3MH0Z6\
74btlJ20XDTAfh0llRgiDxnqrH492ZwH0r6EjVfkH6KhB9R2Duxvpzp4LeBWZ0ouL\
E5MiJBe0aR7CBUcjkq8FzLL3Caf0c7w3Dbtldn4zQdIoHswsD8sFyk2vT4a3QXBV/\
U3Q0DvCXyb5kZp7JOfzNAMJbCD6IVEbDrwqfu4JfTOXAWY8ckofq9zFK+JtkB9XJy\
a7pVJVtI0dKCnVTw3oU4eufj4OxwSGnAyVrf9rEv6jB+4R7tCQZdtuJO7SPAUbJsc\
rD4qjnOz/BDvx2LUQuQXMrbr7GQ+lBNC6gZWcUF33RGc+wIAjvR8Q==\
 alumno@debian
```

2. Enviar la clave pública al servidor:

```
# Añadimos la clave pública a nuestro fichero de claves públicas en el
# host remoto.
alumno$ ssh <host_remoto> "mkdir .ssh/; umask 077; cat >> .ssh/authorized_keys" < ~/.ssh/id_
```

En este punto deberíamos tener ya acceso al host remoto. Para ello se nos preguntará por la frase que usamos cuando generamos las claves de autentificación. Esta forma de autentificación sólo se establecerá cuando accedemos desde el host local usando el usuario `alumno`. Si accedemos desde otro host o con otro nombre de usuario, se utiliza la autentificación mediante password.

Taller 9.2: Pruebe la autentificación basada en clave pública accediendo a un host remoto. Si no dispone de cuenta en un host remoto, utilice su propio host. Emplee en este caso una cuenta de usuario distinta cuyo login sea "usuario_ssh".

Cuestión 2: Especifique cada uno de los pasos que ha realizado para probar la autentificación basada en clave pública cuando accede a un host "remoto". Indique el objetivo de cada uno de estos pasos.

Autentificación basada en el host del usuario: Para habilitar esta forma de autentificación debemos tener acceso al host remoto como administrador. Estos son los pasos:

1. Acceder al host remoto y comprobar que `HostbasedAuthentication = yes` y que `IgnoreRhosts = no`.
2. Reiniciar el servidor SSH en el host remoto si se ha tenido que modificar su configuración.
3. Copiar la *host key* pública del cliente en el servidor. La clave pública en el cliente se almacena en `/etc/ssh/ssh_host_rsa_key.pub`. Dicha clave debe ser añadida al fichero `/etc/ssh/ssh_known_hosts` o al fichero `/.ssh/known_hosts`.
4. En el servidor, añadir el nombre del host cliente al fichero `/.shosts`.
5. En el cliente, asegurarse de que `HostbasedAuthentication = yes`.
6. En el cliente, asegurarse de que la *host key* privada y pública existen. Esta información debería estar en los ficheros `/etc/ssh/ssh_host_rsa_key` y `/etc/ssh/ssh_host_rsa_key.pub`, respectivamente.

Debido a que necesitamos disponer de la clave de root en el host remoto, esta forma de autentificación es poco corriente. Además, como veremos a continuación, la autentificación basada en clave pública es muy cómoda y segura si utilizamos un agente SSH.

9.6.2. Usando `ssh-agent` y `ssh-add`

Para que la identificación basada en clave pública sea realmente segura la frase que usemos para generar la clave de autentificación debe ser suficientemente larga (20 caracteres o más). Esto puede suponer un embrollo cuando accedemos muchas veces al host remoto.

Para solucionar este problema, el SSH incorpora un demonio llamado `ssh-agent` que se encarga de escribir por nosotros las frases. Este demonio puede ser invocado por el usuario de la siguiente manera:

```
# ¿Qué procesos se están ejecutando?
alumno$ ps x
 PID TTY      STAT   TIME COMMAND
   : :        :      :    :
3096 pts/1    R+     0:00 ps x

# Comprobamos que entre las variables de entorno declaradas
# no aparecen SSH_AGENT_PID y SSH_AUTH_SOCK.
alumno$ export
declare -x HISTCONTROL="ignoredups"
declare -x HOME="/home/alumno"
declare -x LANG="es_ES.UTF-8"
:
declare -x TERM="xterm"
declare -x USER="alumno"

# Lanzamos el demonio. Este comando comprueba primero que no existe
# ya un agente ejecutándose. La directiva del shell "eval" permite
# ejecutar ssh-agent con su parámetro.
alumno$ test -z "$SSH_AUTH_SOCK" && eval "ssh-agent -s"
Agent pid 3256
alumno@debian:~$ export
declare -x HISTCONTROL="ignoredups"
declare -x HOME="/home/alumno"
:
```

```
declare -x SSH_AGENT_PID="3256"
declare -x SSH_AUTH_SOCK="/tmp/ssh-kVGvUg3255/agent.3256"
:
declare -x TERM="xterm"
declare -x USER="alumno"

alumno$ ps x
  PID TTY      STAT   TIME COMMAND
    : :        :      :    :
 3056 ?        Ss     0:00 ssh-agent -s
 3099 pts/1    R+     0:00 ps x
```

Una vez que el demonio está ejecutándose, informarle de nuestra clave de autentificación pública es sencillo:

```
# Deberemos introducir la clave pública introducida en
# la fase de autentificación basada en clave pública...
alumno$ ssh-add
```

En este punto, accederemos al host remoto sin necesidad de escribir la frase de acceso porque ssh-agent lo ha hecho por nosotros.

Evidéntemente, como podemos tener acceso a distintas máquinas con distintas frases de acceso, podemos usar tantas veces `ssh-add` como deseemos. Para saber qué frases son admistradas en un momento dado por `ssh-agent`:

```
alumno$ ssh-add -l
```

Las frases puede ser eliminadas del demonio usando:

```
alumno$ ssh-add -d [fichero_con_la_clave]
```

Usar `man ssh-add` para ver el resto de opciones.

Taller 9.3: Haga uso `ssh-agent` para acceder a un host remoto sin necesidad de introducir la clave pública. Si no dispone de cuenta en un host remoto, utilice su propio host y la cuenta de usuario "usuario_ssh".

Cuestión 3: Presente un volcado de pantalla mostrando la ejecución del anterior taller. Explique brevemente lo que aparece en dicho volcado.

9.6.3. Ejecución de comandos no interactivos en el host remoto

Con SSH es posible ejecutar comandos no interactivos en el host remoto. Un ejemplo:

```
# Averiguando los usuarios actuales en el host remoto
alumno$ ssh <host_remoto> who

# Viendo el contenido actual del directorio /tmp en el host remoto
alumno$ ssh <host_remoto> ls /tmp

# Copiando un fichero al host remoto
alumno$ cat fichero | ssh <host_remoto> "cat > fichero"

# Copiando un directorio completo al host remoto
alumno$ tar cvf - directorio/ | gzip -9c | ssh <host_remoto> "tar xz"
```

Taller 9.4: Pruebe a ajecutar un comando en un host remoto. Si no dispone de cuenta en un host remoto, hágalo en su propio host usando la cuenta de usuario "usuario_ssh".

Cuestión 4: Presente un volcado de pantalla mostrando la ejecución del anterior taller. Explique brevemente su contenido.

9.6.4. Verificación de la host key

Como vimos al comienzo de esta práctica, SSH incorpora un mecanismo para autentificar al servidor basado en la comparación de la *host key* que envía el servidor con la que hay almacenada en el fichero `/.ssh/known_hosts`. Si es la primera vez que nos conectamos o por algún motivo, la *host key* ha cambiado, es una buena costumbre que comprobemos que dicha clave coindice con la que el servidor generaría utilizando las herramientas del SSH. Para mostrar la *host key* en el servidor podemos utilizar el siguiente comando:

```
# Mostrando la host key del servidor.
alumno$ ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
```

En el fichero `/etc/ssh/ssh_host_rsa_key.pub` se almacena la *host key* pública de forma cifrada usando el algoritmo RSA. En el fichero `/etc/ssh/ssh_host_rsa_key` se almacenan la *host key* privada de forma cifrada usando el algoritmo RSA.

Taller 9.5: Fuerce a que SSH le muestre la *host key* de un host remoto. Para ello debe borrar la entrada correspondiente a dicho host en el fichero `~/.ssh/known_hosts` (puede borrar directamente el fichero). Acceda al host remoto y anote la *host key* que le muestra el cliente. En el host remoto ejecute el comando que muestra la *host key* y compruebe que ambas claves coinciden. Si no dispone de cuenta en un host remoto, utilice su propio host y la cuenta de usuario "usuario_ssh".

Cuestión 5: Presente un volcado de pantalla mostrando la ejecución del anterior taller. Comente brevemente su contenido.

9.6.5. Copiando ficheros

SSH incorpora una utilidad semejante a cp para copiar ficheros entre hosts de una forma muy sencilla. Algunos ejemplos:

```
# Copiando "fichero" al home del host remoto:
alumno$ scp fichero <host_remoto>:~

# Copiando "fichero" al /tmp del host remoto:
alumno$ scp fichero <host_remoto>:/tmp

# Copiando "fichero" al /tmp del host remoto y cambiando el nombre:
alumno$ scp fichero <host_remoto>:/tmp/file

# Copiando un "directorio" con todos sus contenidos al home del host remoto:
alumno$ scp -r directorio <host_remoto>:~

# También podemos utilizar sftp (un intérprete semejante a ftp) para
# mover ficheros entre hosts:
alumno$ sftp <host_remoto>
# Los comandos de sftp son similares a los del programa ftp.

# Por último, podemos hacer copias incrementales con la utilidad rsync:
alumno$ rsync * <host_remoto>
```

Taller 9.6: Efectúe una copia remota entre hosts. Si no dispone de cuenta en un host remoto, utilice su propio host y la cuenta de usuario "usuario_ssh".

Cuestión 6: Presente un volcado de pantalla mostrando la ejecución del anterior taller. Explique brevemente su contenido.

9.6.6. SSH forwarding

El SSH tiene la habilidad de multiplexar varias conexiones sobre un mismo tunnel cifrado. A esta propiedad se le conoce como *forwarding* (encaminamiento). Veámos algunos ejemplos interesantes:

Forwarding del agente de autentificación

Como vimos anteriormente, el agente de autentificación es muy útil para lidiar con las frases de clave. El problema es que si hemos accedido a un host remoto A y desde él queremos acceder a otro B usando de nuevo el agente, deberíamos ejecutarlo en A y añadirle las frases. Para conseguir esto podemos hacer dos cosas:

1. Configurar nuestro cliente para que redirija el agente:

```
# También vale el fichero /etc/ssh/ssh_config
alumno$ cat ~/.ssh/config
:
Host nombre_host_remoto
ForwardAgent yes
:
```

2. Invocar al cliente mediante:

```
# Este comando lanzará ssh-agent en el host remoto y copiará las claves.
alumno@host_local$ ssh -A <host_remoto>
```

Para verificar que el redireccionamiento ha funcionado, comprobaremos que el agente en el host remoto contempla nuestras claves:

```
alumno@host_remoto$ ssh-add -l
```

Taller 9.7: Redirija ssh-agent hacia un host remoto y compruebe que las claves se han copiado. Si no dispone de cuenta en un host remoto, hágalo contra su propio host usando la cuenta de usuario "usuario_ssh".

Cuestión 7: Presente un volcado de pantalla mostrando la ejecución del anterior taller. Explique el funcionamiento de ssh-agent en dicho ejemplo.

Redireccionamiento del X11

Cuando usamos SSH desde una consola gráfica, podemos ejecutar aplicaciones gráficas en el host remoto y presentarlas en el host local. Para conseguir esto debemos:

1. Configurar nuestro cliente para que rediriga el X11:

```
# También vale el fichero /etc/ssh/ssh_config
alumno$ cat ~/.ssh/config
:
Host nombre_host_remoto
ForwardX11 yes
:
```

2. Invocar al cliente mediante:

```
alumno@host_local$ ssh -X nombre_host_remoto
```

En este punto podríamos ejecutar una aplicación gráfica, un `xterm`, por ejemplo:

```
alumno@host_remoto$ xterm &
```

Taller 9.8: Para probar la redirección del X11 lo ideal es tener cuenta en un host remoto con el servidor X11 y el servidor SSH activados. Si este no es el caso, podemos instalar en el host huésped (Windows) el software Cygwin (`http://www.cygwin.com/`) y acceder desde una consola gráfica al host virtual usando SSH. Una vez que hayamos accedido al host virtual lanzaremos alguna aplicación gráfica y comprobaremos que la redirección ha funcionado. Más concretamente:

1. Instale Cygwin en el host huésped. Los paquetes que se instalan por defecto instalan un cliente SSH y el sistema gráfico X11. Si esto no ocurriera así, instale la aplicación `xterm`.
2. Ejecute una consola de texto. Tras la instalación anterior, en su escritorio debería haber aparecido un icono que ejecuta dicha consola.
3. Escriba en ella:

   ```
   startx &
   ```

   Transcurridos unos segundos debería aparecer una ventana con un terminal gráfico (este terminal se invoca con la aplicación `xterm`).
4. En el terminal gráfico realice las siguientes acciones:

   ```
   # Habilitamos el sistema gráfico para que cualquier host pueda
   # enviar su salida gráfica X11 a nuestro sistema.
   alumno$ xhost +

   # Accedemos al host virtual
   alumno$ ssh -X <dir_IP_del_host_virtual>
   ```
5. Cuando esté dentro del host virtual, ejecute una aplicación gráfica:

   ```
   alumno$ xeyes &
   ```

Cuestión 8: Presente un volcado de pantalla mostrando la ejecución del anterior taller. Comente brevemente qué es lo que ocurre en dicho volcado de pantalla.

Redireccionamiento de puertos

El redireccionamiento de puertos es una característica utilísima de SSH. Con esto podemos acceder a servicios que están filtrados por un cortafuegos y cifrar transmisiones sobre redes no seguras.

Antes de mostrar cómo se realiza el redireccionamiento, una aclaración importante. El objetivo del redireccionamiento es codificar la comunicación desde nuestro host a otro distinto, situado fuera de nuestra red "segura". En este sentido podemos encontrarnos los siguientes contextos:

- Que el host que ejecuta el servicio (llamémoslo host X) no tiene un servidor SSH instalado o no tenemos cuenta de usuario en él. En este caso deberíamos tener una cuenta en otro host Y situado en la misma red segura que el host X que ejecuta el servicio, y el redireccionamiento se realizaría desde nuestro host al host Y usando un canal seguro, y desde el host Y al host X usando un canal inseguro, pero a través de la red que se considera segura. Usaremos el término *redireccionamiento local indirecto* para referirmos a esta posibilidad.
- Que tenemos cuenta de usuario en el host que ejecuta el servicio. En este caso sólo tenemos que redireccionar el puerto desde nuestro host al host remoto. LLamaremos a esta forma de redireccionamiento, *redireccionamiento local directo*.
- Con SSH es posible utilizar servicios que están tras un cortafuegos que no permite las conexiones SSH entrantes (desde Internet hacia la red protegida). Esto puede hacerse con el *redireccionamiento remoto*. Para ello debemos tener acceso físico al host que ejecuta el servicio y redirigir el puerto del servicio a un puerto en el host que está fuera de la red protegida.

Bien, una vez hechas estas aclaraciones explicaremos cómo redireccionar puertos en cada caso. En los ejemplos que se muestran a continuación supondremos que hemos configurado nuestro navegador Web para que utilize un proxy Web situado en nuestro host (`home_host`) escuchando en el puerto 1234, aunque dicho servidor esté realmente ejecutándose en un host remoto que llamaremos `squid_host` y al que tenemos acceso mediante SSH. Supondremos además que en la red local (y por tanto, segura) de `squid_host` existe otro host remoto `remote_host` en el que también tenemos cuenta.

Redireccionamiento local directo:

```
# La forma estándar es:
alumno@home_host$ ssh -L 1234:squid_host:3128 squid_host
# Nota importante: si "alumno" no tiene cuenta como "alumno" en
# "squid_host" y por lo tanto tiene que especificarlo, este
```

```
# procedimiento puede no funcionar :-(

# Aunque también funciona:
alumno@home_host$ ssh -L 1234:localhost:3128 squid_host
```

Con este ejemplo, hemos redirigido el puerto 1234 del `home_host` al puerto 3128 de host `squid_host`.

Redireccionamiento local indirecto:

```
alumno@home_host$ ssh -L 1234:squid_host:3128 remote_host
# Nota importante: si "alumno" no tiene cuenta como "alumno" en
# "squid_host" y por lo tanto tiene que especificarlo, este
# procedimiento puede no funcionar :-(
```

En este ejemplo el puerto 1234 del host local se ha redirigido al puerto 3128 del host `squid_host` a través del host `remote_host`.

Redireccionamiento remoto:

```
alumno@squid_host$ ssh -R 1234:localhost:3128 home_host
```

Redireccionamos el puerto 3128 del host `squid_host` al puerto 1234 del host `home_host`.

Taller 9.9: Redirija el puerto 1234 hacia un host remoto que ejecuta Squid en el puerto 3128. Configure su navegador Web para que utilice un proxy escuchando en el host local en el puerto 1234. Compruebe que puede navegar. Utilice como host local el PC huésped (Windows/Cygwin o Linux) y como host remoto el PC virtual (Debian). Si no desea o no puede ejecutar Cygwin, también puede hacerlo desde el propio PC virtual. Finalmente, existe la posibilidad de duplicar el PC virtual y acceder desde uno hasta otro.

Cuestión 9: Presente un volcado de pantalla mostrando la ejecución del anterior taller. Explique su funcionamiento.

Práctica 10

Un Pinger basado en el UDP

El objetivo de esta práctica es implementar un "Pinger" utilizando el UDP (User Datagram Protocol) y así aprender a usar su API.

Un Pinger es una aplicación de red que envía uno o varios paquetes de datos desde el host que ejecuta la parte cliente hasta el host que ejecuta la parte servidora. Esta los devuelve y permite calcular el RTT (Round-Trip Time) entre ambos hosts.

10.1. El comando `ping`

Un Pinger es una aplicación básica de chequeo del estado de la red. Por este motivo, en la mayoría de los sistemas operativos encontramos una utilidad llamada `ping`. Veámos un ejemplo:

```
ping www.google.es
PING www.l.google.com (72.14.221.147): 56 data bytes
64 bytes from 72.14.221.147: icmp_seq=0 ttl=238 time=64.6 ms
64 bytes from 72.14.221.147: icmp_seq=1 ttl=238 time=66.7 ms
64 bytes from 72.14.221.147: icmp_seq=2 ttl=238 time=65.1 ms
64 bytes from 72.14.221.147: icmp_seq=3 ttl=238 time=65.5 ms
 (pulsamos <CTRL> + <c>)
--- www.l.google.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 64.6/65.4/66.7 ms
```

Como podemos ver, por defecto se envían paquetes cuyo payload contiene 56 bytes, que sumados a los 8 bytes de cabecera del IGMP dan 64 bytes, el tamaño de los paquetes transmitidos (sin contar la cabecera del IP). Nótese que no se utiliza ningún protocolo de la capa de transporte.

El cliente calcula el tiempo que transcurre desde que cada paquete es enviado hasta que la correspondiente contestación es recibida. Para finalizar, calcula el tiempo mínimo, medio y el máximo.

10.2. El ICMP (Internet Control Message Protocol)

La utilidad `ping` utiliza el protocolo de control de mensajes de Internet o ICMP para funcionar. Dicho protocolo pertenece a la pila de protocolos TCP/IP y contempla un mensaje llamado `Echo Request` (Petición de Eco). Cuando un host funciona adecuadamente, está conectado a Internet y no ha sido expresamente configurado para denegar las contestaciones `Echo Request` envía un mensaje ICMP del tipo `Echo Response` al host que le ha hecho la petición.

Los mensajes de petición y de respuesta son generalmente muy cortos. Esto significa que los tiempos de transmisión son prácticamente nulos y por tanto, `ping` se utiliza normalmente para medir el RTT entre nuestro host y el host remoto.

10.3. El Pinger

El Pinger presentado es una aplicación que utiliza el UDP. El cliente envía paquetes UDP y el servidor los recibe y los retorna. Para describir ambas partes se ha utilizado el lenguaje de programación Java.

10.3.1. El servidor

El servidor es básicamente un bucle infinito que espera a recibir paquetes UDP. Para ello escucha en el puerto que se le indica por la línea de comandos cuando es ejecutado.

Como la mayoría de las ocasiones tendremos que ejecutar el cliente y el servidor en el mismo host, el servidor puede configurarse para:

1. **Simular la pérdida de paquetes:** Cuando el servidor y el cliente están suficientemente alejados (en hops) los paquetes deberían perderse con una cierta probabilidad (puesto que el UDP no es un protocolo fiable (como el TCP), no todos los paquetes que el cliente envía van a llegar al servidor y no todos los paquetes que el servidor devuelve al cliente van a llegar a éste).

   Sin embargo, dentro del mismo host o incluso en la red de laboratorio, es prácticamente imposible que se pierdan paquetes. Para simular dicho comportamiento el servidor deja de enviar un eco al cliente con una cierta probabilidad controlada por el parámetro que llamado `LOSS_RATE`.

2. **Simular el retraso promedio de los paquetes:** El parámetro `AVERAGE_DELAY` se encarga de retrasar las contestaciones del servidor una determinada cantidad de tiempo, en promedio. De esta forma, se simula además una cierta distancia entre el servidor y el cliente, aunque estos estén ejecutándose en el mismo host.

A continuación se presenta el código que implementa el servidor (`http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/PingServer.java`):

```
import java.io.*;
import java.net.*;
import java.util.*;

/*
 * Un servidor que procesa peticiones ping sobre UDP.
 */
public class PingServer {
    private static final double LOSS_RATE = 0.3;
    private static final int AVERAGE_DELAY = 100; // milisegundos

   public static void main(String[] args) throws Exception {
       /* Comprobamos que se han introducido los parámetros de entrada
  desde la línea de comandos. */
      if (args.length != 1) {
  System.out.println("You must specify a port");
  return;
      }

      /* Puerto de escucha del servidor. */
      int port = Integer.parseInt(args[0]);

      /* Usaremos un generador de números aleatorios para simular la
       * pérdida de paquetes y el retraso de la red. */
      Random random = new Random();

      /* Creamos un socket de tipo DatragramSocket para recibir y
       * enviar paquetes UDP. */
      DatagramSocket socket = new DatagramSocket(port);

      /* Processing loop. */
      while (true) {

  /* Creamos la estructura de datos que nos servirá para
     almacenar un paquete UDP. */
  DatagramPacket request = new DatagramPacket(new byte[1024], 1024);

  /* Nos bloqueamos hasta que se recibe el siguiente paquete
```

```
     UDP. */
  socket.receive(request);

  /* Imprimimos el payload del paquete. */
  printData(request);

  /* Decidimos si responder o simular la pérdida del paquete. */
  if (random.nextDouble() < LOSS_RATE) {
      System.out.println("   Reply not sent.");
      continue;
  }

  /* Simulamos el retraso de la red. */
  Thread.sleep((int) (random.nextDouble() * 2 * AVERAGE_DELAY));

  /* Enviamos la respuesta. */
  InetAddress clientHost = request.getAddress();
  int clientPort = request.getPort();
  byte[] buf = request.getData();
  DatagramPacket reply = new DatagramPacket(buf, buf.length, clientHost, clientPort);
  socket.send(reply);
  System.out.println("   Reply sent.");
      }
   }

    /*
     * Imprime los datos del datagrama (payload) sobre la salida estándar.
     */
    private static void printData(DatagramPacket request) throws Exception {
/* "buf" apunta al comienzo de los datos en el paquete. */
byte[] buf = request.getData();

/* Convertimos "buf" en un stream de entrada de bytes. */
ByteArrayInputStream bais = new ByteArrayInputStream(buf);

/* Convertimos "bais" en un stream de entrada de caracteres. */
InputStreamReader isr = new InputStreamReader(bais);

/* Convertimos "isr" en un stream de entrada de caracteres
   buffereado. Así podremos leer líneas completas. Una línea
   es cualquier combinación de caracteres que acaba en
   cualquier combinación de \r y \n. */
BufferedReader br = new BufferedReader(isr);

/* Los datos del mensaje están contenidos en una única
// línea. Así la leemos. */
String line = br.readLine();

/* Imprimimos la dirección del host que envía el mensaje y los
// datos del mismo. */
System.out.println("Received from " +
   request.getAddress().getHostAddress() +
   ": " +
   new String(line) );
    }
}
```

Taller 10.1: Compile el servidor usando el comando:

```
# Este comando genera el fichero "PingServer.class".
```

```
javac PingServer.java
```

Evidéntemente, deberá tener instalado un compilador de Java para realizar la anterior acción. Existen versiones tanto para Linux como para Windows. Se recomienda el compilador y la máquina virtual de Sun para realizar esta parte de la práctica (`http://java.sun.com`).

Taller 10.2: Ejecute el servidor con el comando:

```
# Recuerde que hay que utilizar un puerto mayor que 1024
#   si no somos el usuario root. Además, el puerto no debería estar
#   ocupado por ningún otro proceso.
# El fichero "PingServer.class", generado tras la compilación,
#  debería existir en el directorio actual.
java PingServer 6789
```

10.3.2. El cliente

El cliente envía 10 peticiones de ping al servidor especificado. Cada mensaje contiene un payload de datos donde figura la cadena PING, un número de secuencia y una estámpa de tiempo.

Tras enviar un paquete, el cliente espera hasta un segundo para recibir una respuesta. Si transcurrido este tiempo ésta no llega, el cliente supone que su paquete de petición o el paquete de respuesta (o ambos) se ha(n) perdido.

A continuación se muestra el código del cliente (`http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/PingClient.java`):

```
import java.io.*;
import java.net.*;
import java.util.*;

/*
 * Un cliente que genera peticiones ping sobre UDP.
 */

public class PingClient {
    public static final String CRLF="\r\n";

    public static void main(String[] args) throws Exception {

/* Comprobamos que se han introducido los parámetros de entrada
   desde la línea de comandos. */
if (args.length != 2) {
    System.out.println("You must specify a server and a port");
    return;
}

/* Obtenemos la dir IP del servidor a partir de su nombre. */
InetAddress serverAddress = InetAddress.getByName(args[0]);

/* Obtenemos el puerto en el que escucha el servidor. */
int serverPort = Integer.parseInt(args[1]);

/* Creamos un socket de tipo DatragramSocket para recibir y
 * enviar paquetes UDP. */
DatagramSocket socket = new DatagramSocket();

/* Asignamos un tiempo máximo de espera de recepción a través
 * de este socket (time-out) de un segundo. */
socket.setSoTimeout(1000);
```

```
/* Creamos la estructura de datos que almacenará un paquete UDP. */
DatagramPacket reply = new DatagramPacket(new byte[1024], 1024);

/* Enviamos los 10 paquetes. */
for (int i=0; i<10; ++i) {

    /* Generamos el payload del paquete con el instante en que
     * el mismo es generado. */
    long sendingDate = (new Date()).getTime();
    String payload = "Ping " + (i+1) + " " + sendingDate + CRLF;
    DatagramPacket request =
new DatagramPacket(payload.getBytes(),
   payload.getBytes().length,
   serverAddress,
   serverPort);

    System.out.print("Packet sent with payload: " + payload);

    /* Enviámos el paquete. */
    socket.send(request);

    /* Esperamos la respuesta del servidor. */
    try {
socket.receive(reply);
    }
    catch(SocketTimeoutException e) {
/* Se ha producido un time-out. */
System.out.println("Not response from server");
continue;
    }

    /* Calculamos el tiempo que el paquete ha tardado en regresar. */
    long receivingDate = (new Date()).getTime();
    System.out.println("Packet received after " +
       (receivingDate-sendingDate) + " miliseconds");
}
    }
}
```

Taller 10.3: Compile y ejecute el cliente:

```
javac PingClient.java
java PingClient localhost 6789
```

Cuestión 1: Presente un ejemplo de interacción entre el cliente y el servidor.

Cuestión 2: Escriba una versión del cliente y del servidor Pinger usando el lenguaje de programación C. Compruebe que sus versiones funcionan y presente un ejemplo de interacción entre el cliente y el servidor, escritos ambos en C. Presente también el código desarrollado debidamente comentado.

Cuestión 3: Compruebe que puede intercambiar cada una de sus aplicaciones por la respectiva aplicación escrita en Java. Muéstrese un ejemplo de interacción.

Cuestión 4: Realice un experimento que le permita comprobar si el servidor puede atender a más de un cliente a la vez, es decir, si podemos ejecutar más de un cliente simultáneos que usen el mismo servidor. Muestre dicho experimento.

Cuestión 5: Instale la máquina virtual de Java en el sistema operativo anfitrión y compruebe si el servidor puede servir a un cliente ejecutado en el PC virtual. Realice lo mismo, pero ahora ejecute el cliente en el host anfitrión y el servidor en el PC virtual. Muestre dichos experimentos.

Cuestión 6: Ejecute un servidor en un PC virtual dentro de un host X y un cliente en otro PC virtual dentro de un host Y. ¿Son capaces de comunicarse? Explique su respuesta.

Cuestión 7: Ejecute un servidor en un host X (no virtual) y un cliente en host Y (no virtual). ¿Son capaces de comunicarse? Explique su respuesta.

Cuestión 8: Cree una copia del PC virtual (el PC "origen" debería estar apagado) y ejecute ambas instancias en un mismo host. Ejecute un servidor en uno y un cliente en otro. ¿Son capaces de comunicarse? Explique su respuesta.

Práctica 11

Un servidor simple basado en el TCP

En esta práctica vamos a diseñar un servidor simple basado en el TCP. El objetivo es entender cómo funciona la API del TCP tanto en C como en Java.

11.1. El servidor

Nuestro servidor es una aplicación escrita en C que mantiene una variable que puede ser modificada por los clientes. Cada cliente es servido en un hilo diferente que perdura tanto tiempo como la conexión con el cliente. Como puede haber más de un cliente conectados de forma simultanea, se trata, por tanto, de un servidor concurrente.

El servidor imprimirá por la consola el valor actual de una variable compartida. Este proceso se ejecutará hasta que pulsemos <CTRL> + <C> en la consola donde hemos ejecutado el servidor.

A continuación se muestra el código fuente del servidor (http://www.ace.ual.es/~{}vruiz/docencia/redes/practicas/progs/servidor.c):

```
/*
 * servidor.c
 *
 * Acepta conexiones TCP a través del puerto PORT para que distintos
 * clientes puedan modificar una variable compartida
 * (shared_value). Las acciones que pueden realizar los clientes
 * depende del valor especificado en cada interacción. Si el valor
 * introducido es <= -2, el cliente es desconectado. Si el valor
 * introducido es -1, el cliente es informado del valor actual de la
 * variable compartida. En cualquier otro caso, el valor introducido
 * es tomado por la variable compartida.
 *
 * Para detener el servidor, pulsar <CTRL>+<C> en la consola donde se
 * lanzó el servidor.
 *
 * Compilar escribiendo:
 *
 * gcc servidor.c -o servidor -lpthread
 *
 * gse. 2007.
 *
 */

/*****************************************************************************
 *
 * Ficheros cabecera.
 *
 *****************************************************************************/

/* Entrada y salida de streams buffereados. */
#include <stdio.h>
```

```
/* Biblioteca de funciones estándar (exit(), EXIT_SUCCESS,
   EXIT_FAILURE, ...) */
#include <stdlib.h>

/* Manipulación de cadenas y movimiento de memoria (memset(), ...). */
#include <string.h>

/* Biblioteca estándar de funciones relacionadas con el sistema
   operativo Unix (read(), write(), ...). */
#include <unistd.h>

/* POSIX Threads (pthread_t, pthread_create(), ...). */
#include <pthread.h>

/* Sockets. */
/* Más info en:
 *
 * http://www.csce.uark.edu/~aapon/courses/os/examples/concurrentserver.c
 * http://beej.us/guide/bgnet/
 */

/* Tipos de datos primitivos del sistema. */
#include <sys/types.h>

/* Sockets (socket(), listen(), ...). */
#include <sys/socket.h>

/* Direcciones IP, opciones y definiciones. */
#include <netinet/in.h>

/* Servicio de resolución de nombres. */
#include <netdb.h>

/* Signals (interrupciones) (signal(), SIGINT). */
#include <signal.h>  /* signal(), SIGINT */

/*****************************************************************************
 *
 * Definiciones.
 *
 *****************************************************************************/

/* Puerto de escucha del servidor. */
#define PORT 6789

/* Número máximo de clientes que esperan a que la conexión sea
   establecida. */
#define QLEN 1

/*****************************************************************************
 *
 * Variable globales.
 *
 *****************************************************************************/

/*****************************************************************************
 *
 * Funciones.
```

```
 *
 ******************************************************************************/

/* Cuerpo principal del programa. */
int main(int argc, char *argv[]) {
  work(argc, argv);
  return EXIT_SUCCESS;
}

/* Función que realiza todo el trabajo. */
work(int argc, char *argv[]) {

  /* Un semáforo para acceder "en exclusión mutua" a una zona de
     código. */
  pthread_mutex_t mut;

  /* Creamos el semáforo. */
  pthread_mutex_init(&mut, NULL);

  /* La variable compartida que puede ser controlada por los clientes. */
  int shared_value = 0;

  /* Creamos el socket TCP de escucha. */
  int listen_sd; {

    /* Obtenemos el número del protocolo. */
    struct protoent *ptrp; /* Pointer to a protocol table entry. */
    ptrp = getprotobyname("tcp");
    if((int)(ptrp) == 0) {
      perror("getprotobyname");
      exit(EXIT_FAILURE);
    }

    /* Creamos el socket. */
    listen_sd = socket (PF_INET, SOCK_STREAM, ptrp->p_proto);
    if(listen_sd < 0) {
      perror("socket");
      exit(EXIT_FAILURE);
    }

    /* Usaremos el puerto de servicio sin esperar a que éste esté
       libre. */
    int yes = 1;
    if(setsockopt(listen_sd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) < 0) {
      perror("setsockopt");
      exit(EXIT_FAILURE);
    }
  }

  /* Asignamos una dirección (dir IP, puerto) al socket de escucha. */ {
    struct sockaddr_in sad;              /* Dirección del servidor. */
    memset((char  *)&sad,0,sizeof(sad)); /* Borramos la estructura. */
    sad.sin_family = AF_INET;            /* Usaremos Internet. */
    sad.sin_addr.s_addr = INADDR_ANY;    /* Cualquiera de las IP
    asignadas al host vale. */
    sad.sin_port = htons((u_short)PORT); /* Asignamos el puerto de
    escucha. */
    if (bind(listen_sd, (struct sockaddr *)&sad, sizeof (sad)) < 0) {
      perror("bind");
```

```
      exit(EXIT_FAILURE);
    }
  }

  /* Comenzamos a escuchar. */
  if (listen(listen_sd, QLEN) < 0) {
    perror("listen");
    exit(EXIT_FAILURE);
  }

  fprintf(stderr,"%s: esperando conexiones ...\n", argv[0]);

  /* Si pulsamos CRTL+C, el programa acaba ejecutando la función
     end(). */
  void end() {
    fprintf(stderr,"%s: <CTRL>+<C> detectado. Saliendo ...\n",argv[0]);

    /* Cerramos el socket de escucha. */
    close(listen_sd);

    /* Salimos. */
    exit(EXIT_SUCCESS);
  }
  signal(SIGINT, end);

  /* Lazo del servidor. */
  while(1) {

    /* Socket para "servir" a los clientes. */
    int serve_sd;

    /* Esperamos a que un cliente se conecte. */ {
      struct sockaddr_in cad;        /* Client's address. */
      socklen_t alen = sizeof(cad);  /* Tamaño de la dirección. */
      serve_sd = accept(listen_sd, (struct sockaddr *)&cad, &alen);
      if(serve_sd<0) {
perror("accept");
exit (EXIT_FAILURE);
      }
    }

    fprintf(stderr,"%s: conexión aceptada!\n",argv[0]);

    /* Hilo que controla a un cliente. */
    void *service(void *arg) {

      /* El socket de conmunicación con el cliente. Nótese que cada
 cliente posee una variable "sd" diferente. */
      int sd = (int)arg;

      for(;;) {
int new_shared_value;

inform_to_client(sd, shared_value);
receive_from_client(sd, &new_shared_value);

/* Looping mientras new_shared_value > -2. */
if(new_shared_value < -1) break;
```

```
/* Modificamos shared_value mientras new_shared_value > -1. */
if(new_shared_value > -1) {
  /* Sección crítica. */
  pthread_mutex_lock(&mut);
  shared_value = new_shared_value;
  fprintf(stderr,"%s: shared_value = %d\n", argv[0], shared_value);
  pthread_mutex_unlock(&mut);
  /* Fin de la sección crítica. */
}

      }

      /* El cliente ha introducido -2. */

      /* Cerramos la conexión con el cliente. */
      close(sd);

      /* Finalizamos el hilo. */
      pthread_exit(0);
    }

    /* Lanzamos el hilo. */
    pthread_t service_tid; /* Thread identificator. */
    if(pthread_create(&service_tid, NULL, service, (void *)serve_sd) == -1) {
      perror("pthread_create");
      exit(EXIT_FAILURE);
    }
  }

}

/* Informa a un cliente sobre el valor de shared_value. */
inform_to_client(int sd, int shared_value) {
  char message[80];
  memset(message, 80, 0);
  sprintf(message,"shared_value = %d. Introduzca nuevo valor ... ", shared_value);
  send(sd, message, strlen(message), 0);
}

/* Recibe del cliente un nuevo valor. */
receive_from_client(int sd, int *shared_value) {
  char message[80];
  read_text_line(sd, message);
  sscanf(message,"%d",shared_value);
}

/* Lee una línea de texto ASCII. */
read_text_line(int sd, char *str) {
  int n;
  do {
    n = recv(sd, str, 1, 0);
  } while (n>0 && *str++ != '\n');
}
```

Taller 11.1: Compile el servidor escribiendo:

```
gcc servidor.c -o servidor -lpthread
```

Taller 11.2: Ejecute el servidor escribiendo:

```
./servidor
```

11.2. El cliente

El usar un socket TCP y transmitir los datos en ASCII nos permite utilizar el programa Telnet como cliente. He aquí una interacción ejemplo:

```
usuario$ telnet localhost 6789
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
shared_value = 0. Introduzca nuevo valor ... 3
shared_value = 3. Introduzca nuevo valor ... -1
shared_value = 3. Introduzca nuevo valor ... -2
Connection closed by foreign host.
```

Como puede verse, en cada interacción el cliente obtiene el valor actual de la variable compartida `shared_value` y puede especificar un valor distinto. En función del valor especificado podemos:

- Si el valor introducido es menor o igual que -2, el cliente desea cerrar la conexión con el servidor.
- Si el valor introducido es -1, el cliente desea conocer el valor actual de la variable compartida (sin alterar su valor). Téngase en cuenta que desde que se recibe el último valor de la variable compartida hasta que otra es introducida, otro cliente ha podido modificar dicho valor. Por tanto, especificar el mismo valor que el actual no valdría como consulta.
- Si el valor introducido es mayor o igual que 0, el cliente modifica el valor de la variable compartida.

Taller 11.3: Escriba una versión del servidor usando el lenguaje de programación Java. Compruebe que su versión funciona correctamente.

Cuestión 1: Presente el código desarrollado y un ejemplo de interacción.

Cuestión 2: Usando Wireshark, capture una interacción entre dos clientes y un servidor cuando todos se ejecutan dentro del mismo host. ¿Cómo distingue el servidor a los dos clientes?

Práctica 12

DHCP (Dynamic Host Configuration Protocol)

El DHCP [4], como su nombre indica, es un protocolo para la configuración dinámica del IP. Este protocolo sirve fundamentalmente para dos cosas: (1) que dicha configuración no se tenga que realizar a mano (muy útil cuando hay muchos nodos que configurar o hay que configurar muchas veces con distintos parámetros un nodo) y (2) acomodar a más nodos que direcciones IP hay disponibles (sólo en el caso de que nunca todos los nodos estén encendidos a la vez, claro).

Por su comodidad, el DHCP es muy usado en entornos inalámbricos donde los hosts constantemente entran y salen de las BSS's (Basic Services Sets).

12.1. Clientes, servidores y agentes de retransmisión

El DHCP es un protocolo cliente-servidor. Un cliente es una computadora que desea configurar su TCP/IP y un servidor es otra que sabe cómo hacerlo.

Generalmente existe un servidor por subred. Sin embargo, puede ocurrir también que existan muchos o ninguno. El primer caso se suele dar cuando el conjunto de direcciones IP disponibles es tan grande que el trabajo debe ser dividido. El segundo ocurre cuando este conjunto es tan pequeño que no se justifica un servidor en la subred. En este caso un router, que hace de *DHCP relay agent*, conoce la dirección IP de un servidor externo y permite que los clientes accedan al servidor DHCP a través de él.

12.2. Sobre las configuraciones asignadas

En el sistema DHCP hay uno o varios hosts servidores DHCP que mantienen la base de datos con las configuraciones del resto de hosts que son los clientes DHCP. Las direcciones IP disponibles pueden ser asignadas mediante dos políticas diferentes: (1) estática, cuando los mismos hosts siempre reciben las mismas direcciones IP y (2) dinámica, cuando las direcciones IP recibidas pueden cambiar, dependiendo, por ejemplo, del orden en que se encienden las máquinas. Al rango estático se le suele llamar *intervalo de exclusión* mientras que al dinámico, *ámbito*. A la suma de ambos conjuntos de direcciones se le llama *conjunto de direcciones disponibles*.

12.3. El proceso de concesión

En la Figura 12.1 se presenta un time-line del proceso de *concesión DHCP*. Dicho proceso se realiza siempre bajo demanda del cliente, cuando es encendido o cuando expira la anterior concesión. Toda la comunicación la soporta el UDP.

A continuación explicamos con mayor detalle cada uno de los mensajes.

12.3.1. La solicitud de concesión

Cuando un host cliente es encendido o desea renovar su dirección IP emite a la dirección de broadcast de la subred un paquete UDP dirigido al puerto 67 (servicio `bootps`[1]) y que contiene un mensaje del tipo *DHCP*

[1] Véase el fichero `/etc/services`.

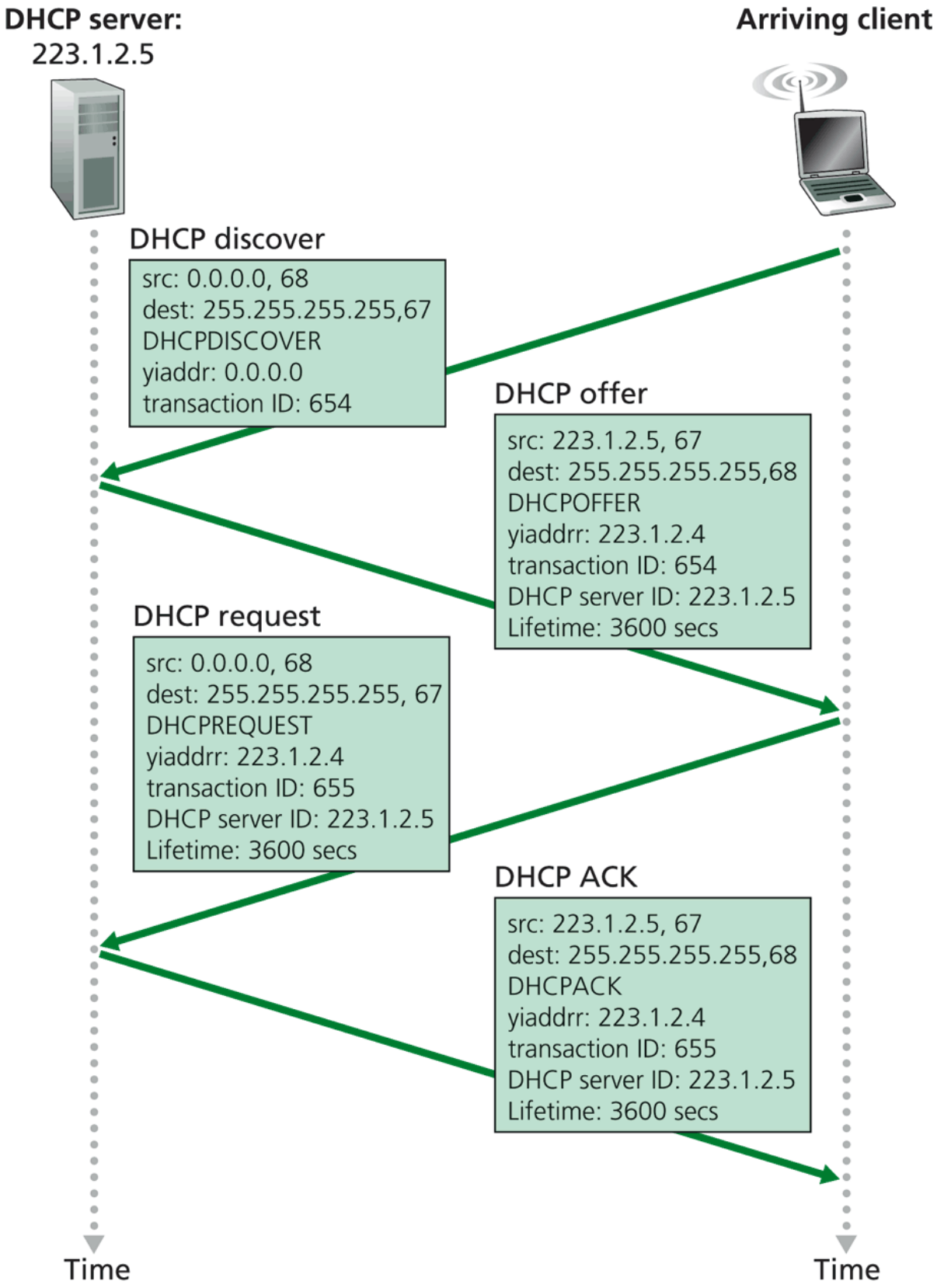


Figure 5.21 ♦ DHCP client-server interaction

Figura 12.1: Proceso de concesión utilizando el DHCP [8]. La notación es la siguiente: "src": dirección IP y puerto origen del paquete, "dest": dirección IP y puerto destino el paquete, "yiaddr": Your Internet ADDRess y "ID": IDentification.

Discover. Nótese que el servidor DHCP debe estar, por tanto, en la misma subred que los clientes (excepto cuando existe un relay agent que se encarga de solventar este problema). Este mensaje va a ser recibido por todos los adaptadores conectados a la red local. Tras esta transmisión, el cliente espera un tiempo a la contestación del servidor.

12.3.2. La oferta de concesión

Los servidores DHCP (puede haber muchos) contestan al cliente con un mensaje del tipo *DHCP Offer*. Este contiene una dirección IP, la máscara de la red, la dirección IP del gateway, la(s) dirección(es) IP con el/los servidores DNS y el *tiempo de préstamo* (*lease time*) de la dirección IP.[2] Como el paquete emitido utiliza el UDP (y por tanto el IP) y el cliente no tiene todavía una dirección IP válida, no queda más remedio que enviarlo, de nuevo, a la dirección de broadcast de la red.

12.3.3. La selección de concesión

De entre todas las direcciones IP ofertadas, el cliente selecciona una, aunque todavía no la usa. Entonces envía un paquete UDP a la dirección de broadcast de la red con el mensaje *DHCP request*. En éste replica los datos de la oferta que acaba de recibir.

12.3.4. La confirmación de selección

Los servidores DHCP van a recibir este mensaje y el que realizó la oferta va a emitir, de nuevo a la dirección de broadcast de la red, un paquete UDP con el mensaje *DHCP Ack* (Acknowledgment). Cuando el cliente lo recibe sabe que ya está en condiciones de utilizar la nueva configuración.

Si el servidor finalmente negara al cliente el uso de la configuración ofertada[3], en lugar de transmitirse un DHCP Ack se transmitiría un *DHCP NAck* (Negative Ack) y todo el proceso comienza de nuevo.

12.4. Instalación del servidor DHCP

Vamos a instalar el paquete dhcp3-server (`http://www.isc.org/products/DHCP`) del Internet Systems Consortium (ISC). En concreto, el comando para instalar el servidor `dhcp3` sería:

Debian Linux:

```
# Instala el servidor y el relay agent.
root# apt-get install dhcp3-server
```

Fedora Core Linux:

```
# Instala el servidor, el relay agent y el cliente.
root# yum install dhcp3
```

Gentoo Linux:

```
# Instala el servidor, el relay agent y el cliente.
root# emerge dhcp
```

12.5. Configuración del servidor

El servidor DHCP se configura modificando el fichero de configuración correspondiente.

Debian: El fichero de configuración del servidor es `/etc/dhcp3/dhcpd.conf`. No hay ningún fichero ejemplo, así que es una buena costumbre hacer una copia de este fichero antes de realizar ninguna modificación sobre el mismo.

Fedora core: El fichero de configuración del servidor es `/etc/dhcpd.conf`. Hay un ejemplo en `/usr/share/doc/dhcp-<version>/dhcpd.conf.sample`.

Gentoo Linux: El fichero de configuración del servidor es `/etc/dhcp/dhcpd.conf`. Hay un ejemplo en `/etc/dhcp/dhcpd.conf.sample`.

[2] Ocurre que cuando los hosts son apagados (o se bloquean, por ejemplo) no existe una obligación de comunicar este hecho a los servidores DHCP. El tiempo de préstamo sirve para que las direcciones IP puedan ser reutilizadas en estos casos.

[3] Nótese que esto puede ocurrir si otro cliente más "adelantadillo" ha reclamado antes para sí la concesión.

12.6. Configuración del cliente

La configuración del cliente DHCP `dhclient` recae en el fichero `dhclient.conf`. La localización de este fichero depende de la distribución de Linux:

Debian Linux:

`/etc/dhcp3/dhclient.conf`

Fedora Core Linux:

`/etc/dhclient.conf`

El fichero `dhclient.conf` debe ser creado desde cero utilizando un editor de ficheros ASCII.

Gentoo Linux:

`/etc/dhcp/dhcpd.conf`

Hay una copia en `/etc/dhcp/dhcpd.conf.sample`.

Los parámetros que podemos controlar en este fichero van desde seleccionar el timeout de contacto con un servidor DHCP hasta el tiempo máximo de arrendamiento de la dirección IP asignada (véase `dhclient.conf(5)` para más información). Sin embargo, como la mayoría de estos parámetros los controla el servidor, rara vez es necesario editarlo. Por este motivo, incluso si dicho fichero no existe, el sistema DHCP funcionará correctamente (con los parámetros por defecto).

12.7. Configuración del TCP/IP en caliente

Cuando un interfaz de red no esté configurado o deseemos re-configurarlo (usando el DHCP), podemos hacerlo escribiendo:

```
# Configurando mediante DHCP el interfaz de red eth0
root# dhclient eth0

# O simplemente
root# dhclient
# si sólo existe un único interface de red (eth0)
```

Por supuesto, es necesario tener acceso a un servidor DHCP.

Taller 12.1: Vamos a instalar un servidor DHCP y a usarlo. Para ello ejecute los siguientes pasos:

1. Ejecute el PC virtual.
2. Instale el servidor DHCP.
3. Configurare a mano el IP. Siga los pasos que vió en la Sección 4.3.2. Determine la dirección de la red y la dirección del gateway que le permita seguir navegando por Internet. Anote estos parámetros (en adelante supondremos que la red es `192.168.213.0/24`, que la dirección IP de nuestro host es la `192.168.213.128` y que nuestro gateway escucha en la `192.168.213.2`).
4. Configure el servidor DHCP usando los anteriores parámetros:
   - El servidor DHCP dará servicio a una red privada `192.168.213.0/24`.
   - La dirección IP `192.168.2.4` se asigna siempre de forma estática (por ejemplo, para un posible servidor). El resto de direcciones IP se asignan de forma dinámica.
   - El tiempo de préstamo por defecto (si el cliente no solicita lo contrario) es de 600 segundos y como máximo de 7200 segundos.
   - El DNS primario es `150.214.156.2` y el secundario `150.214.35.10`.
   - El dominio de los clientes es establecido a `lab_redes.ace.ual.es`.[4]

[4] Consulte `resolv.conf(5)` si necesita más información sobre qué significa el dominio de búsqueda.

- El DNS no es avisado de las asignaciones lo que implica que los hosts de la red local no van a estar dados de alta en el DNS. Esto es lo más frecuente en redes privadas ya que los hosts internos no son visibles desde el exterior y por lo tanto, no tiene sentido darlos de alta en el DNS.
- La asignación dinámica de direcciones IP se producirá (1) cuando los clientes arrancan y (2) cuando expira el uso de la dirección IP anteriormente asignada.
- El servidor DHCP que configuramos va a ser el servidor DHCP oficial de la sub-red.

Con toda esta información, el contenido del fichero `dhcpd.conf` sería:

```
# No vamos a actualizar el DNS
ddns-update-style none;

# Definimos el dominio de los clientes
option domain-name "lab_redes.ace.ual.es";

# Definimos los servidores de nombres que utilizarán los clientes
option domain-name-servers 150.214.156.2, 150.214.35.10;

# Tiempo de las concesiones, por defecto
default-lease-time 600;

# Tiempo máximo de las conexiones
max-lease-time 7200;

# Este es el servidor DHCP que manda
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# Definimos el rango de dirs IP disponibles
subnet 192.168.213.0 netmask 255.255.255.0 {
  range 192.168.213.4 192.168.213.254;
  option broadcast-address 192.168.213.255;
  option routers 192.168.213.2;
}

# Definimos las concesiones estáticas
host laptop {
  # La dir física del NIC para el cliente "192.168.213.3".
  hardware ethernet 00:10:5a:2e:56:a7;
  # Dirección IP del cliente. También se puede usar un nombre. En
  # este último caso el servidor DHCP utilizará el DNS para
  # resolverla.
  fixed-address 192.168.213.3;
  option broadcast-address 192.168.213.255;
  option routers 192.168.213.2;
}
```

5. Reinicie el servicio DHCP que acaba de configurar y compruebe que arranca sin problemas (véase el Apéndice C).
6. El VMware ha instalado un servidor DHCP en la computadora huésped que va a competir con el servidor que a continuación va a instalar. El primer paso consiste en deshabilitar este servidor. Es imprescindible que el PC virtual esté apagado (ni tan siguiera pausado). Apague el PC virtual.
7. Desactive el servidor DHCP del VMware:

   En Microsoft Windows XP: Acceder a Inicio -> Panel de Control -> Herramientas Administrativas -> Servicios. Localizar el servicio VMware DHCP Service, pinchar sobre

la entrada pulsando el botón derecho del ratón y detener (desactivar) o pausar dicho servicio. Es importante que no exista ninguna máquina virtual corriendo.

8. Ejecute el PC virtual. El dispositivo de red Ethernet debe estar conectado y configurado como un NAT.
9. Compruebe que el servidor DHCP está funcionando y si no es así, láncelo. No hace falta que lo dé de alta en los scripts de arranque porque en el resto del curso volveremos a utilizar el servidor DHCP de la computadora huésped.
10. Reconfigure el IP del PC virtual (véase la Sección 12.7) y compruebe que el mensaje `DHCPACK` proviene de su propio host (`192.168.213.128`). Compruebe que el IP ha sido correctamente configurado.
11. Ejecute Wireshark.
12. Conteste ahora a las cuestiones de evaluación.
13. Apague el PC virtual, deshabilite el servidor DHCP instalado en el PC virtual y habilite el servicio VMware DHCP en el host huésped.

Cuestión 1: Renueve la configuración del IP para `eth0` mientras captura paquetes con el sniffer. Escriba en el campo de filtrado "bootp" (el DHCP es una evolución del protocolo bootp que se utiliza para arrancar las máquinas sin disco). Deberíamos ver los paquetes DHCP Discover, DHCP Offer, DHCP Request y DHCP Ack ¿Qué protocolo de la capa de transporte es utilizado por el DHCP?

Cuestión 2: ¿Cuál es la dirección física del interface de red de nuestro host?

Cuestión 3: ¿Coincide ésta con la que devuelve `/sbin/ifconfig`?

Cuestión 4: Renueve la configuración capturando de nuevo los paquetes generados el sistema DHCP. ¿Ha cambiado el campo Transaction-ID del mensaje DISCOVER respecto de la primera concesión?

Cuestión 5: ¿Cuál es la dirección IP del servidor DHCP?

Cuestión 6: ¿A qué puerto son enviadas las peticiones? ¿A través de qué puerto son retornadas las contestaciones?

Cuestión 7: ¿Cuál es la dirección IP concedida?

Cuestión 8: ¿Cuánto dura la concesión?

Cuestión 9: Vacíe el campo de filtrado. ¿Se generan paquetes ARP durante la transacción DHCP? ¿Para qué sirven estos paquetes? (conteste a esta pregunta incluso cuando estos paquetes no se generan).

Práctica 13

Rastreo del TCP

En esta práctica vamos a estudiar el TCP en detalle. Para ello vamos a analizar la traza de los segmentos TCP que se envían y reciben cuando se transfiere un fichero de 150 KB (que contiene el texto de Lewis Carroll: *Alicia en el País de las Maravillas*. Estudiaremos cómo el TCP utiliza los números de secuencia y de reconocimiento para conseguir una transferencia sin errores, y cómo funciona el mecanismo de control del flujo y de la congestión. También calcularemos tiempos de transmisión y tasas de transferencia para conocer el rendimiento de la conexión.

13.1. Capturando ...

Para realizar nuestro estudio necesitamos capturar los paquetes generados en la transmisión utilizando un *packet sniffer* (véase el Apéndice F), cuando transmitimos el fichero anteriormente mencionado desde nuestro host hasta un servidor Web.

Taller 13.1:

1. Ejecute un navegador Web.
2. Acceda a la URL `http://gaia.cs.umass.edu/ethereal-labs/alice.txt` y almacene en disco el fichero `alice.txt`.
3. Acceda a la URL `http://gaia.cs.umass.edu/ethereal-labs/TCP-ethereal-file1.html` donde debería ver una página Web como la que aparece en la Figura 13.2.
4. Utilice el botón *Browse* de esta página para indicar el camino al fichero `alice.txt` en su host. No pulse todavía el botón *Upload alice.txt file*.
5. Ejecute el *packet sniffer* y colóquelo en modo de captura de paquetes.
6. Pulse ahora el botón *Upload alice.txt file*. Cuando el fichero haya sido completamente subido a `gaia.cs.umass.edu`, aparecerá un mensaje que se lo indica (véase la Figura 13.3).
7. Detenga el *packet sniffer*. Debería haber obtenido una captura semejante a la que se muestra en la Figura 13.4).
8. Filtre los paquetes "tcp" (véase la Figura 13.5). Deberían aparecer el three-way handshake que contiene un mensaje SYN, un mensaje HTTP POST y una serie de mensajes "HTTP Continuation". Estos mensajes son la forma en que el *packet sniffer* indica que existen muchos segmentos TCP para transmitir un único mensaje HTTP. También debería ver los paquetes TCP ACK que `gaia.cs.umass.edu` envía hasta su computadora.

Cuestión 1: ¿Cuál es la dirección IP y el número de puerto usado por su host?

Cuestión 2: ¿Cuál es la dirección IP de `gaia.cs.umass.edu`?

Cuestión 3: ¿De qué puerto parten los segmentos en `gaia.cs.umass.edu`?

Cuestión 4: ¿Hacia qué puerto en `gaia.cs.umass.edu` van dirigidos los suyos?

Taller 13.2:

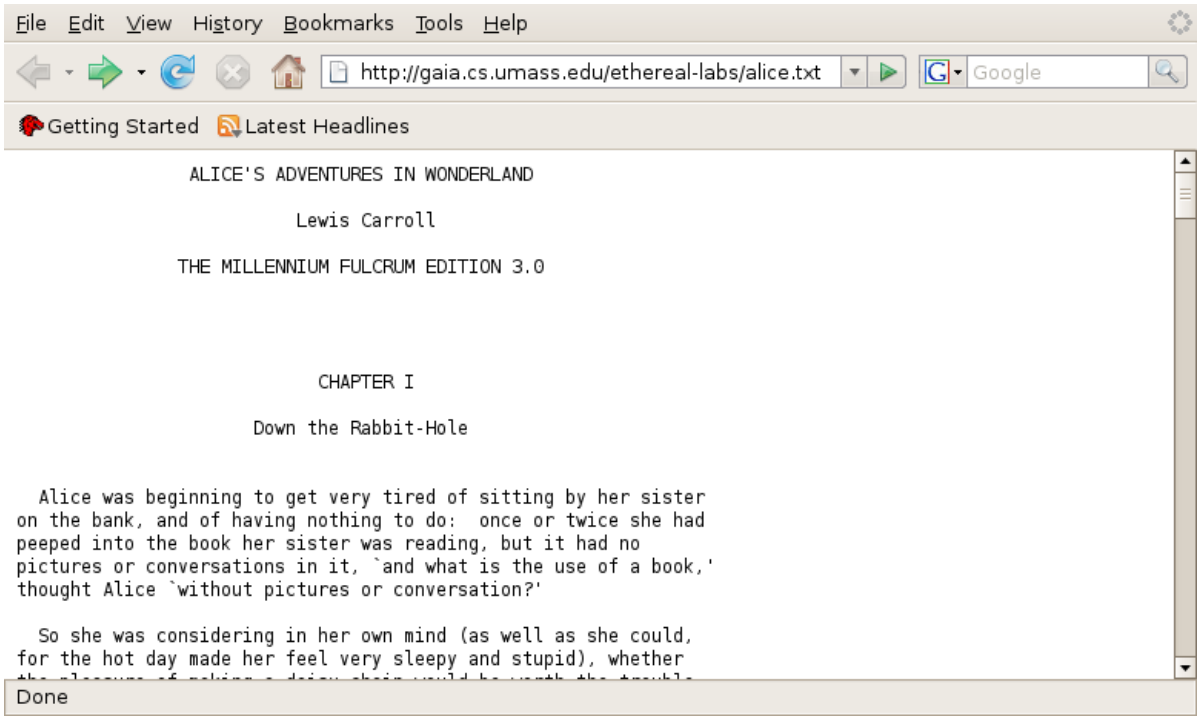


Figura 13.1: Página Web con el fichero `alice.txt`.

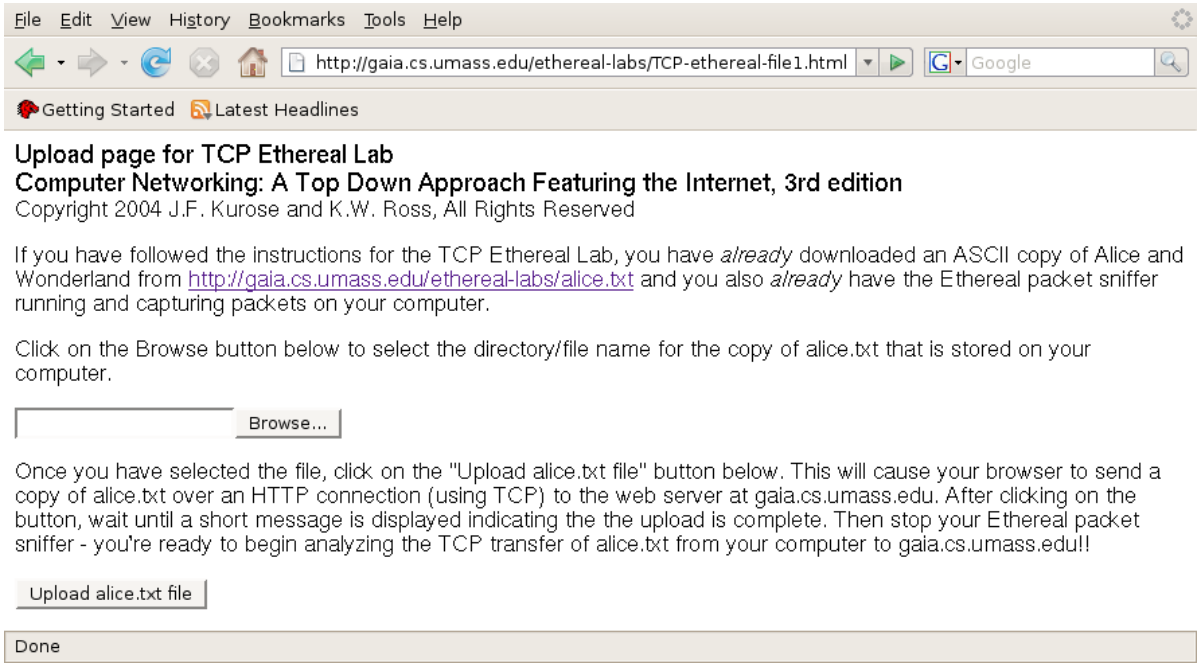


Figura 13.2: Página Web donde podemos subir el fichero `alice.txt`.

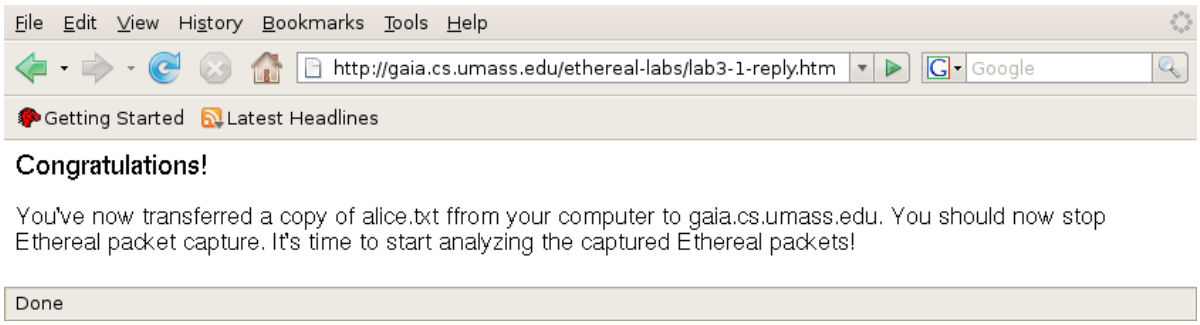


Figura 13.3: Felicitaciones por haber subido con éxito el fichero `alice.txt`.

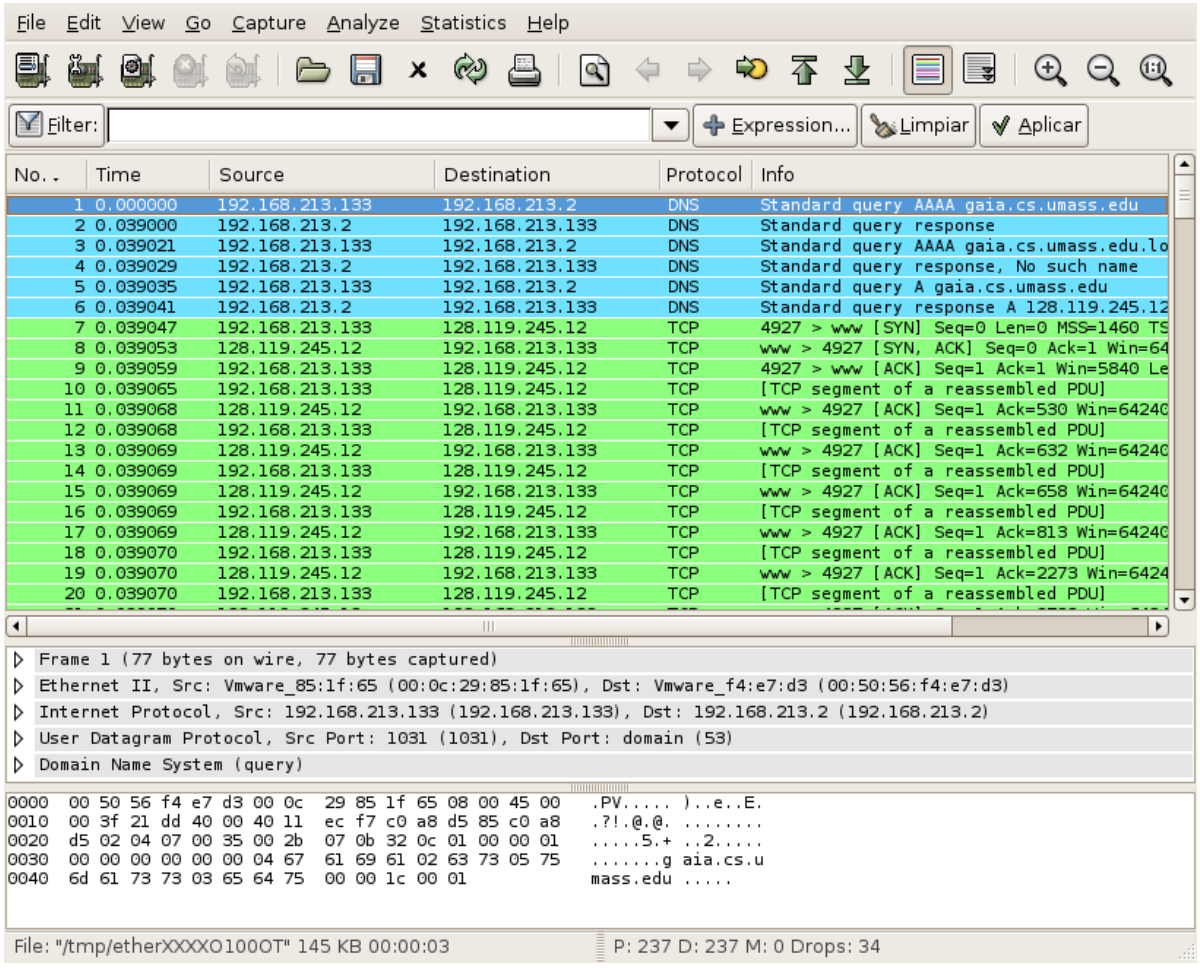


Figura 13.4: Captura ejemplo tras transmitir el fichero `alice.txt`.

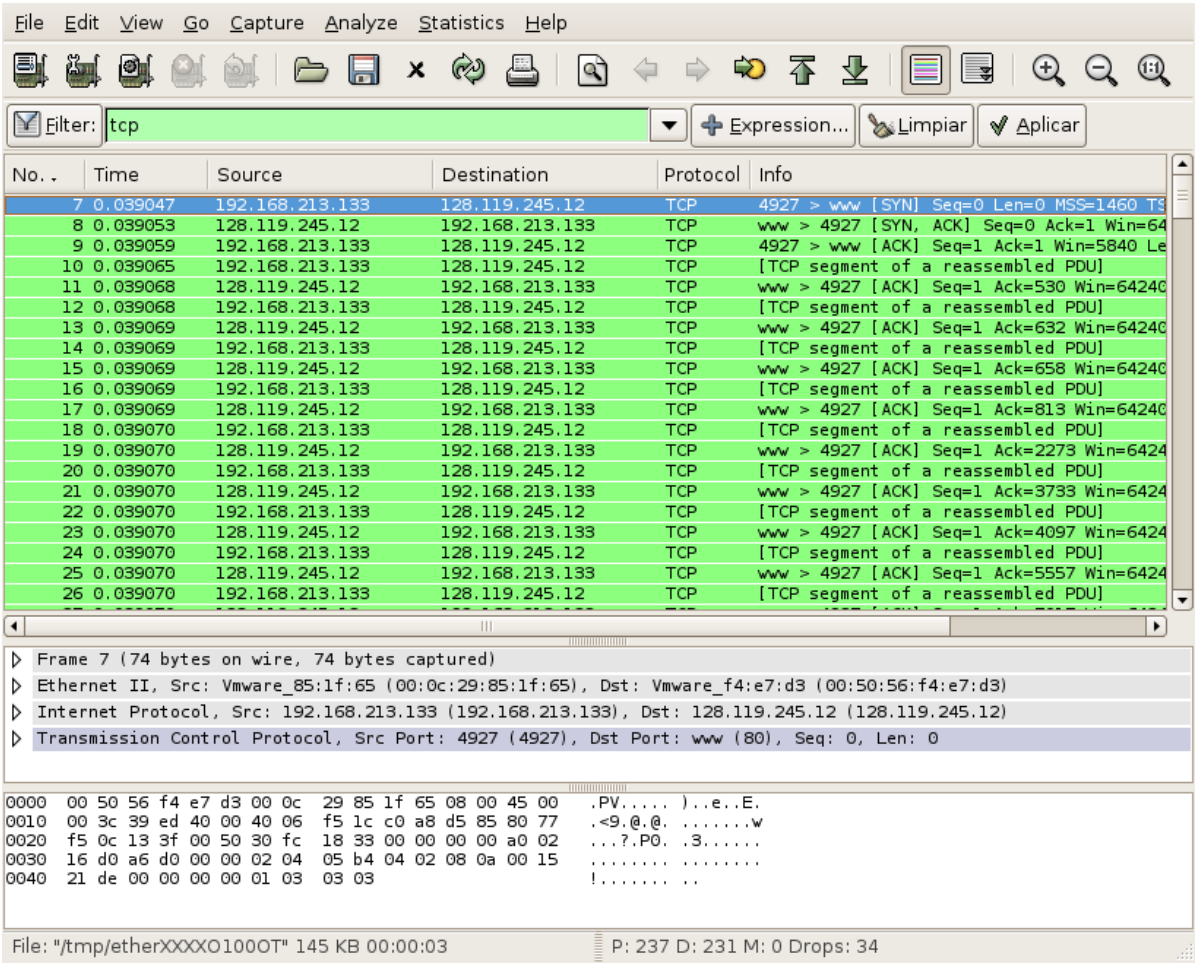


Figura 13.5: La captura del fichero `alice.txt` tras usar el filtro "tcp".

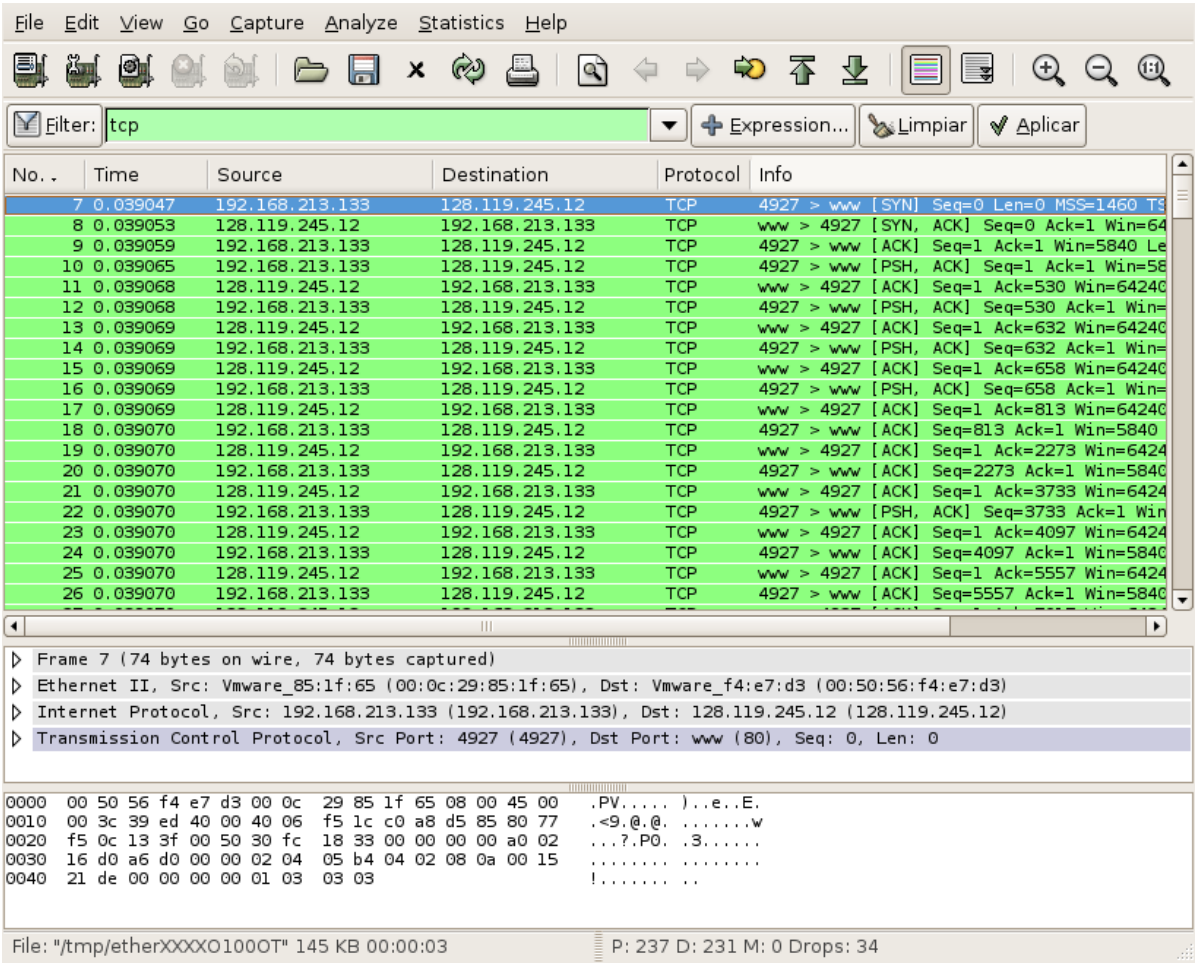


Figura 13.6: La captura del fichero `alice.txt` tras usar el filtro "tcp" y deshabilitar el análisis del HTTP.

1. Deshabilite el análisis del HTTP. Acceda a *Analyze* -> *Enabled Protocols*, deseleccione la caja para HTTP y pulse el botón *OK*. En este instante debería ver los paquetes capturados de forma parecida a como se muestran en la Figura 13.6.

Cuestión 5: ¿Cuál es el número de secuencia usado en el segmento TCP SYN que inicia la conexión entre su host y `gaia.cs.umass.edu`?

Cuestión 6: ¿Qué distingue a un segmento TCP SYN de otro que no lo sea?

Cuestión 7: ¿Cuál es el número de secuencia usado en el segmento TCP SYNACK enviado por `gaia.cs.umass.edu`?

Cuestión 8: ¿Qué relación tienen los números de secuencia de los segmento SYN y SYNACK?

Cuestión 9: ¿Cuál es el número de secuencia del segmento que contiene el comando HTTP POST y el de los siguientes cinco segmentos que van dirigidos desde su host hacia `gaia.cs.umass.edu`? (Busque este segmento buscando en los payload's de los segmentos.)

Cuestión 10: ¿En qué instante de tiempo se han generado cada uno de esos seis segmentos?

Cuestión 11: ¿En qué instante de tiempo se han recibido los correspondientes ACK's?

Cuestión 12: ¿Cuál es el RTT para esos seis segmentos?

Cuestión 13: ¿Cuál es el valor `EstimatedRTT` (véase la página 237 del texto de [8]) tras la recepción de cada ACK? (Asuma que el valor de `EstimatedRTT` es igual al RTT medido para el primer segmento.)

Cuestión 14: ¿Cuál es la longitud de cada uno de estos seis segmentos?

Cuestión 15: ¿Cuál es la cantidad mínima de buffer disponible que el receptor advierte el emisor?

Cuestión 16: ¿Se está produciendo en algún momento un control del flujo? Razone su respuesta.

Cuestión 17: ¿Hay retransmisiones de segmentos? ¿Cómo lo sabe?

Cuestión 18: ¿Cuál es la cantidad máxima de datos reconocidos por el receptor?

Cuestión 19: ¿Se ha producido algún ACK acumulativo?

Cuestión 20: ¿Cuál ha sido el *throughput* (bytes transferidos por unidad de tiempo) de la conexión? Explique el cálculo.

Taller 13.3:

1. Seleccione un segmento que parta desde su host hacia `gaia.cs.umass.edu` en la ventana de segmentos capturados.
2. Acceda al menú *Statistics* -> *TCP Stream Graph* -> *Time-Sequence-Graph(Stevens)*. Usted debería ver algo parecido a lo que se presenta en la Figura 13.7. En ella, cada punto representa un segmento enviado. Lo que vemos concretamente es el número de secuencia del segmento versus el instante de tiempo en el que el segmento ha sido enviado.

Cuestión 21: ¿Indique en qué instantes de tiempo comienza y finaliza la fase de arranque lento?

Cuestión 22: Viendo esta gráfica (Figura 13.7) y sin tener en cuenta el *throughput*, ¿en qué podría haber mejorado la transmisión?

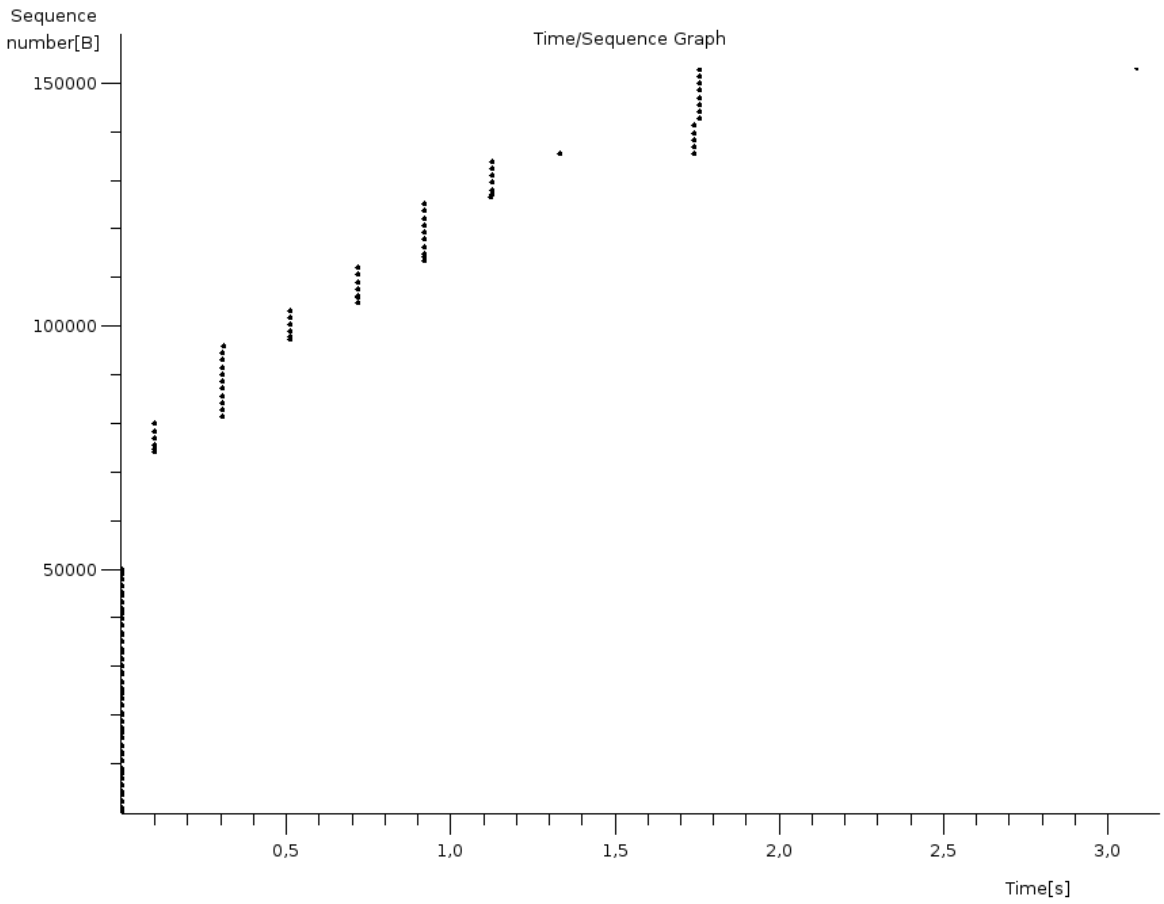


Figura 13.7: Número de secuencia versus instante de tiempo de los segmentos enviados.

Práctica 14

Rastreo del IP

El IP (Internet Protocol) gobierna el trasiego de paquetes por la red. En esta práctica realizaremos una serie de experimentos que nos ayudarán a comprender el funcionamiento de este protocolo tan importante.

14.1. La estructura del paquete

El payload de un paquete IP almacena los datos entregados por el protocolo de la capa de transporte al IP.

> **Taller 14.1:** Diseñe un experimento en el que transmita un único paquete IP desde su host hasta otro diferente. Captúrelo usando un *packet sniffer*.

Cuestión 1: Explique su experimento.

Cuestión 2: ¿Qué versión del IP ha usado?

Cuestión 3: ¿Cuál es la dirección IP del adaptador de red desde el que ha partido dicho paquete?

Cuestión 4: ¿A qué dirección IP ha sido dirigido dicho paquete?

Cuestión 5: ¿Es una dirección IP de una red pública o privada?

Cuestión 6: ¿Qué vale el campo TTL?

Cuestión 7: ¿Cuál ha sido el identificador de protocolo usado en la cabecera IP? ¿Cómo se llama el protocolo asociado?

Cuestión 8: ¿Cuántos bytes contiene la cabecera IP? ¿Cuántos bytes contiene el paquete IP? ¿Cuántos bytes el payload?

Cuestión 9: ¿Cuál ha sido el payload?

14.2. Tiempo de vida de los paquetes

El tiempo de vida de un paquete IP en realidad controla el número de *hops* que el paquete puede realizar antes de que un router lo destruya. Esto se hace para que los paquetes dirigidos a destinos inexistentes no viajen de forma indefinida por la red.

> **Taller 14.2:** Diseñe un experimento en el que un paquete no llege a su destino porque su TTL es demasiado bajo. Capture este paquete y el paquete ICMP *TTL-exceeded* generado usando un *packet sniffer*.

Cuestión 10: Explique su experimento.

Cuestión 11: ¿Qué TTL ha usado?

Cuestión 12: ¿Cuál es la dirección IP del nodo que ha generado el ICMP *TTL-exceeded*?

Cuestión 13: ¿Cuál ha sido el payload del paquete recibido?

14.3. Fragmentación

El IP fragmenta aquellos paquetes que son demasiado grandes para ser transmitidos. Dichos fragmentos son ensamblados en el receptor por su capa de red y así el datagrama o el segmento original es entregado, tal cual, a la capa de transporte.

Taller 14.3: Diseñe un experimento en el que se transmita un mensaje lo suficientemente grande como para que el IP de su host lo fragmente (en más de un datagrama IP).

Cuestión 14: Explique su experimento.

Cuestión 15: ¿Cuántos datagramas IP (fragmentos) ha enviado?

Cuestión 16: ¿Cuántos datagramas IP (fragmentos) recibe el host receptor?

Cuestión 17: ¿Cuántos datagramas UDP o segmentos ha enviado?

Cuestión 18: ¿Cuántos datagramas UDP o segmentos recibe el host receptor?

Cuestión 19: ¿Qué información indica que el datagrama UDP o el segmento ha sido fragmentado por el IP?

Cuestión 20: ¿Qué tamaño tiene el payload de cada fragmento (datagrama IP)?

Cuestión 21: ¿Cómo conoce el IP que el payload de un datagrama IP recibido forma parte (es un subconjunto) del payload un datagrama UDP o de un segmento?

Cuestión 22: Calcule el MTU de la red conectada a su host.

Cuestión 23: ¿Cómo calcularía el MTU de todo el trayecto entre su host y cualquier otro host de Internet?

14.4. NAT

El NAT es el router que nos permite conectarnos a Internet desde una red típicamente privada.

Taller 14.4: Diseñe un experimento en el que compruebe el funcionamiento de su NAT. Dicho dispositivo debería modificar el puerto y la dirección IP origen de todos los paquetes que son enviados por cualquier host de la red privada hacia un host de la red pública.

Cuestión 24: Explique su experimento.

Cuestión 25: ¿Cuál es la dirección IP y el puerto origen de los paquetes cuando estos viajan por la red privada?

Cuestión 26: ¿Y por la red pública? ¿Coinciden?

Cuestión 27: ¿Cuál es la dirección IP y el puerto destino de los paquetes cuando estos viajan por la red privada?

Cuestión 28: ¿Y por la red pública? ¿Coinciden?

Cuestión 29: ¿Modifica el NAT algún otro campo de la cabecera IP (aparte de la dirección IP y el puerto origen del paquete)?

Cuestión 30: Sin necesidad de redireccionar los puertos en el NAT y bajo la suposición de que utiliza el UDP. ¿Qué podría hacer para instalar un servidor en un host de la red privada y que un cliente desde la red pública se conectara a él?

Práctica 15

Rastreo en Ethernet y del ARP

Ethernet es la tecnología de comunicaciones dominante en redes locales de computadoras que están conectadas mediante cables. En esta práctica analizaremos algunos de los aspectos más interesantes de la transmisión de frames Ethernet.

El ARP (Address Resolution Protocol) se utiliza con el mismo objetivo que el DNS aunque a nivel de enlace de datos: determinar las direcciones físicas de los interfaces de red que forman parte de una misma red local. Es decir, con el ARP lo que traducimos son direcciones IP en direcciones físicas.

15.1. La estructura del frame

El frame Ethernet es la estructura de datos que transporta los paquetes generados a nivel de la capa de enlace de datos. A continuación analizaremos algunas de sus características básicas.

Taller 15.1: Diseñe un experimento en el que su host envíe un mensaje a un host que no está en su red local y éste le envíe un mensaje a su host.

Cuestión 1: Exponga su experimento.

Cuestión 2: ¿Qué protocolos a nivel de la capa de aplicación, transporte y red ha usado?

Cuestión 3: ¿Cuántos frames Ethernet han sido necesarios para transportar el mensaje de ida? ¿Cuántos para el mensaje de vuelta?

Cuestión 4: ¿Cuál es el valor del campo *Type* en el/los frame/s Ethernet?

Cuestión 5: ¿Cuál es la dirección física del interface de red del que ha partido el mensaje de ida? Si es incapaz de contestar a esta pregunta, explique por qué.

Cuestión 6: ¿Cuál es la dirección física del interface de red del que ha partido el mensaje de vuelta? Si es incapaz de contestar a esta pregunta, explique por qué.

Cuestión 7: ¿Cuál es la dirección física del interface de red que figura como fuente en el/los frame/s que contiene/n el mensaje de vuelta?

Cuestión 8: ¿A qué nodo se corresponde la anterior dirección física?

Cuestión 9: Basándose en el análisis del frame que realiza el *packet sniffer*, haga un esquema de la estructura de un frame Ethernet (indique qué campos posee, cuál es el tamaño de cada uno de ellos y qué contiene cada campo).

15.2. El ARP

Ahora vamos a estudiar el funcionamiento del ARP, un protocolo que está muy ligado a la tecnología empleada en la capa de enlace de datos.

Taller 15.2: Diseñe un experimento en el que su computadora necesite utilizar el ARP.

Cuestión 10: Explique su experimento.

Cuestión 11: ¿Cuál es la dirección física destino del frame que contiene el mensaje *ARP request*?

Cuestión 12: ¿Cuál es la dirección física origen del frame que contiene el mensaje *ARP request*?

Cuestión 13: ¿A cuántos hosts ha llegado el mensaje *ARP request*?

Cuestión 14: ¿Cuántos hosts han generado un mensaje *ARP response*?

Cuestión 15: ¿Cuál es la dirección física del adaptador de red del que ha partido el mensaje *ARP response*?

Cuestión 16: ¿A qué nodo se corresponde el anterior adaptador de red? ¿Es un nodo de su red local?

Cuestión 17: ¿Cuál ha sido la dirección IP que ha traducido?

Cuestión 18: ¿Se utiliza el ARP normalmente cuando se está comunicando con un host que no está en su red local? Explíquese.

Cuestión 19: Averigue cuánto tiempo cachea su host las entradas en la tabla ARP.

Cuestión 20: ¿Podrían aparecer en dicha tabla traducciones para direcciones que su host nunca ha utilizado? Indique en qué casos.

Apéndice A

Gestión de paquetes en Linux

A la hora de instalar software en Linux tenemos básicamente dos alternativas. La primera consiste en descargar el código fuente del paquete que deseamos y compilarlo. En la segunda alternativa nos bajamos el paquete ya compilado para la versión del sistema operativo que estamos utilizando. En distribuciones como Fedora Core y Debian Linux esta suele ser la opción más frecuente. En otras distribuciones como Gentoo los paquetes se compilan tras las descarga. Veámos cómo se realizan estas tareas en estas tres distribuciones de Linux.

A.1. Debian Linux

En Debian el instalador de paquetes se llama `apt` y lo normal es descargar el paquete ya compilado. Para poderlo usar tenemos que haber accedido al sistema como super-usuario.

Existe mucha más información acerca de la instalación de paquetes usando `apt`. Podremos encontrar gran parte de esta en la la Guía de Referencia Debian (`http://www.debian.org/doc/user-manuals`).

A.1.1. Pasando de stable a testing

Cuando instalamos Debian podemos seleccionar dos versiones: (1) la estable que tiene los paquetes más probados y (2) la de test, con paquetes más nuevos, aunque menos probados. Dependiendo del uso que queramos dar al host deberíamos utilizar una versión u otra. Generalmente, si el host es un servidor seleccionaremos la versión estable que da menos problemas de actualización.

En algún momento puede ser interesante "upgradear" a la versión de pruebas. Esto se puede hacer son los siguientes comandos:

```
root# apt-get update
root# apt-get install libc6 perl libdb2 debconf
root# apt-get install apt apt-utils dselect dpkg

# Editar el fichero /etc/apt/sources.list cambiar "stable" por
# "testing" donde proceda.

root# dselect update  # always do this before upgrade
root# dselect select  # select additional packages
root# dselect install
```

A.1.2. Pasando de testing a unstable

```
# Editar el fichero /etc/apt/sources.list cambiar "testing" por
# "unstable" donde proceda.

root# dselect update  # always do this before upgrade
root# dselect select  # select additional packages
root# dselect install
```

A.1.3. Actualización de la base de datos de paquetes

Antes de realizar una instalación es interesante tener actualizada la base de datos de paquetes en nuestra máquina. Esto se hace escribiendo:

```
root# apt-get update
```

A.1.4. Actualización de los paquetes

Es una buena idea mantener el sistema operativo lo más actualizado posible. Así minimizaremos los agujeros de seguridad y nos aseguraremos de estar utilizando las últimas versiones de los programas. En Debian se actualiza el sistema operativo escribiendo:

```
root# apt-get upgrade
```

A.1.5. Búsqueda de un paquete

Una vez que nos hemos sincronizado con el servidor de apt que hayamos seleccionado (probablemente durante la instalación del sistema operativo) pasaremos a buscar si existe el <paquete> en cuestión. Esto se realiza escribiendo:

```
# Búsqueda sencilla
root# apt-cache search <paquete>

# Búsqueda con información extra
root# apt-cache search --full <paquete>
```

La lista de servidores de paquetes utilizados por apt se encuentra en el fichero ASCII:

```
/etc/apt/sources.list
```

A.1.6. Conocer si un paquete ya está instalado

Para averiguar si un paquete ya está instalado podemos escribir:

```
usuario$ dpkg -l | grep <cadena>
```

donde <cadena> es una cadena de caracteres que pertenecen al nombre del paquete.

A.1.7. Instalación de un paquete

Para instalar un <paquete> escribiremos:

```
root# apt-get install <paquete>
```

A.1.8. Actualización de un paquete

Para actualizar un <paquete> escribiremos:

```
root# apt-get install <paquete>
```

Sí, la misma orden que para instalarlo.

A.1.9. Averiguar los ficheros que ha instalado un paquete

```
root# apt-file update
usuario$ apt-file list <paquete>
```

A.1.10. Averiguar el paquete al que pertence un fichero

```
usuario$ apt-file search <fichero>
```

A.1.11. Encontrar los paquetes de los que depende otro paquete

```
usuario$ apt-cache depends <paquete>
```

A.1.12. Encontrar los paquetes que dependen de un paquete

```
usuario$ apt-cache rdepends <paquete>
```

A.1.13. Borrado de un paquete

Para borrar un paquete, escribir:

```
root# apt-get remove <paquete>
```

Si se desear eliminar además los ficheros de configuración que dependen del paquete, usar:

```
root# apt-get --purge remove <paquete>
```

Esta opción sólo funciona si no existen ficheros nuevos en los directorios que fueran a eliminarse. Si existieran, deberá eliminarlos "a mano".

Finalmente, se pueden desinstalar un conjunto de paquetes escribiendo:

```
root# apt-get remove <paquete>*
```

A.2. Fedora Core Linux

En Fedora Core el instalador de paquetes se llama `yum` y es semejante a Debian. Los paquetes suelen descargarse ya compilados.

A.2.1. Actualización de la base de datos de paquetes

A la hora de usar `yum` no hace falta actualizar la base de datos de paquetes porque de esta tarea se encarga un demonio llamado `yum-updatesd`, que debería estar siempre ejecutándose.

A.2.2. Actualización de los paquetes

```
root# yum update
```

A.2.3. Búsqueda de un paquete

```
root# yum search <paquete>
```

A.2.4. Conocer si un paquete ya está instalado

Escribir:

```
usuario$ rpm --query --all | grep <cadena>
```

donde <cadena> es una cadena de caracteres que pertenecen al nombre del paquete que deseamos saber si está instalado.

A.2.5. Averiguar los ficheros que ha instalado un paquete

```
usuario$ rpm --query --list <nombre_paquete_sin_version>
```

A.2.6. Averiguar el paquete al que pertence un fichero

```
usuario$ rpm --query --file <fichero>
```

A.2.7. Instalación de un paquete

```
root# yum install <paquete>
```

A.2.8. Actualización de un paquete

```
root# yum update <paquete>
```

A.2.9. Encontrar los paquetes de los que depende otro paquete

```
usuario$ ??
```

A.2.10. Borrado de un paquete

```
root# yum remove <paquete>
```

A.3. Gentoo Linux

En Gentoo los paquetes normalmente se descargan en código fuente y se compilan. Esto tiene ciertas ventajas y otros inconvenientes. Las principales ventajas son: (1) que las dependencias entre las versiones de los paquetes es menor y por tanto, la instalación o actualización de un paquete implica recompilar pocas dependencias y (2) el paquete se compila para la arquitectura de nuestra computadora y con las características deseadas. Esto último incrementa el rendimiento de los programas.

El principal problema en Gentoo es que hay paquetes (como Mozilla Firefox) que tardan mucho en compilarse :-(. Por esto, estos paquetes tan pesados también suelen estar disponibles en binario.

A.3.1. Actualización de la base de datos de paquetes

La base de datos de paquetes en Gentoo se denomina "portage". Esta base de datos es un árbol de directorios (y ficheros) que se sincroniza con un servidor mediante `rsync`. La actualización es, debido a la cantidad de información transmitida, un proceso pesado.

Por suerte, los paquetes en Gentoo sólo necesitan ser actualizados cuando aparece una versión nueva de los mismos (no cuando cambia la versión de alguna de sus dependencias, como en Debian). Esto implica que no necesitamos sincronizar portage muy frecuentemente (una vez al mes es más que suficiente para disfrutar de las últimas versiones).

Bien. En Gentoo, portage se actualiza escribiendo:

```
root# emerge --sync
root# emerge portage
```

La lista de servidores de `rsync` y la lista de servidores de paquetes se encuentra en el fichero:

```
/etc/make.conf
```

A.3.2. Actualización de los paquetes

```
root# emerge --update --deep --newuse world
root# etc-update
root# emerge --depclean
root# revdep-rebuild
```

A.3.3. Búsqueda de un paquete

```
root# emerge --search <paquete>
```

A.3.4. Conocer si un paquete ya está instalado

Escribiremos:

```
usuario$ equery list <cadena>
```

donde <cadena> es una cadena de caracteres que pertenecen al nombre del paquete que deseamos saber si está instalado.

A.3.5. Instalación de un paquete

Para instalar un <paquete> escribiremos:

```
root# emerge <paquete>
```

A.3.6. Actualización de un paquete

Para instalar un <paquete> escribiremos:

```
root# emerge --update <paquete>
```

A.3.7. Averiguar los ficheros que ha instalado un paquete

```
usuario$ equery files <paquete>
```

A.3.8. Averiguar el paquete al que pertence un fichero

```
usuario$ equery belongs <fichero>
```

A.3.9. Encontrar los paquetes de los que depende otro paquete

```
usuario$ equery depgraph <paquete>
```

A.3.10. Encontrar los paquetes que dependen de un paquete

```
usuario$ equery depends <paquete>
```

A.3.11. Borrado de un paquete

```
root# emerge --unmerge <paquete>
```

Apéndice B

Administración de cuentas de usuario en Linux

En algún instante puede ser útil crear una cuenta nueva. El comando para hacer esto es:

```
root# adduser <usuario>
```

Para conocer las opciones disponibles para esta utilidad consúltese el manual `man adduser`.
Si lo que desea es eliminar una cuenta, utilice el comando:

```
root# deluser <usuario>
```

Apéndice C

Activación y desactivación de servicios en Linux

Los servicios en Linux (y en UNIX) son proporcionados por los demonios, que no son más que procesos que no dejan nunca de ejecutarse mientras la computadora está encendida. Estos demonios pueden "levantarse" (ejecutarse) o "echarse a bajo" en la sesión actual llamando al script correspondiente con el parámetro adecuado. Típicamente los scripts que levantan y echan abajo los demonios se localizan en el directorio `/etc/init.d`. Si el script que lanza el demonio X se llama `X`, entonces escribiremos:

```
# Levantamos el demonio X
root# /etc/init.d/X start

# Echamos a bajo el demonio X
root# /etc/init.d/X stop

# Forzamos a que el demonio X re-lea los ficheros de configuración
root# /etc/init.d/X restart

# Obteniendo información sobre los parámetros de ejecución del demonio X
root# /etc/init.d/X help
```

Para conseguir que dichos demonios se ejecuten la próxima vez que reiniemos, deben de incluirse los enlaces oportunos en los directorios que controlan la carga de dichos demonios. Estos directorios, desgraciadamente, pueden variar entre distribuciones, así como los comandos y utilidades "gráficas" que los manipulan.

Debian Linux: Los directorios que contiene los enlaces a los demonios a activar y desactivar en cada nivel de ejecución son `/etc/rc.[nivel de ejecución].d`. La utilidad básica es `/usr/sbin/update-rc.d`, aunque también es posible utilizar otros programas más amigables como `/usr/bin/rcconf` y `/usr/bin/services-admin` que habría que instalar previamente.

```
# Mostrar los demonios activos
usuario$ rcconf

# Activar el demonio XXX en los run-levels por defecto
root# update-rc.d XXX defaults

# Desactivar el demonio XXX
root# update-rc.d XXX remove
```

En Debian Linux la instalación de un paquete que contiene un servicio realiza la activación definitiva de dicho servicio. Por tanto, rara vez necesitaremos utilizar la utilidad `update-rc.d`.

Fedora Core Linux: Al igual que en Debian, los directorios que contiene los enlaces a los demonios a activar y desactivar en cada nivel de ejecución son `/etc/rc.[nivel de ejecución].d`. Sin embargo, a diferencia de Debian la utilidad básica es `/sbin/chkconfig`. Finalmente, también existen scripts gráficos como `/usr/sbin/serviceconf`.

```
# Mostrar los demonios activos
usuario$ chkconfig --list

# Activar el demonio XXX en los run-levels por defecto
root# chkconfig --add XXX

# Desactivar el demonio XXX
root# chkconfig --del sshd
```

En Fedora Core Linux la instalación de un paquete que contenga un servicio no lo configura para su ejecución definitiva (probablemente estará desactivado cuando reiniciemos el host).

Gentoo Linux: Los enlaces a los demonios se encuentran en `/etc/runlevels` y la utilidad para manejarlos es `/sbin/rc-update`.

```
# Mostrar los demonios activos
usuario$ /sbin/rc-update show

# Activar el demonio XXX en los run-levels por defecto
root# rc-update add XXX default

# Desactivar el demonio XXX
root# rc-update del XXX default
```

En Gentoo Linux la instalación de un paquete con un determinado servicio no implica su activación definitiva.

Apéndice D

Escaneado de puertos en Linux

Los puertos se utilizan en redes de computadoras para que dos o más procesos puedan comunicarse. En el TCP/IP los puertos se enumeran utilizando un número entero positivo de 16 bits (0 - 65535). Por convenio, los puertos inferiores al 1024 se utilizan para ofrecer servicios conocidos como la Web, Telnet, SSH, etc. [6].

El término "escaneo de puertos" se refiera a la técnica de ir (normalmente secuencialmente) contactando (usando peticiones de conexión TCP o mediante datagramas UDP) con los diferentes puertos de una máquina y ver si hay algún tipo de respuesta [8]. Dependiendo de dichas respuestas puede conocerse qué servicios ofrece dicho host, sistema operativo, modelo del host, etc.

D.1. Cuidado cuando escaneamos una máquina ...

Muchos cortafuegos detectan automáticamente los procesos de escaneo porque son la primera herramienta que utilizan los hackers de redes para acceder ilegalmente a los sistemas. Por tanto, es muy posible que si escanemos un host podamos enfadar a algún administrador que contactará con el nuestro para qujarse. Si nuestra acción no es maliciosa lo más prudente es contactar primero con dicho administrador para pedirle permiso.

D.2. `netstat`

`/bin/netstat` imprime información sobre el sistema de red en el host donde se ejecuta. Esto último implica que tenemos que tener cuenta en la máquina que escaneamos si deseamos usar esta utilidad.

Con `netstat`, dependiendo de los argumentos usados podemos:

Conexiones: Esta es la forma de uso más corriente de `netstat`. Ejemplos:

```
# Mostrando las conexiones TCP establecidas
usuario$ ssh usuario@gogh.ace.ual.es
usuario$ netstat | grep ESTABLISHED
tcp        0      0 192.168.213.133:3841    gogh.ace.ual.es:ssh     ESTABLISHED

# Mostrando los servicios escuchando a través de puertos TCP
usuario$ netstat --listening | grep tcp
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 *:1488                  *:*                     LISTEN
tcp        0      0 *:auth                  *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 *:nessus                *:*                     LISTEN
tcp        0      0 localhost:smtp          *:*                     LISTEN
tcp6       0      0 *:www                   *:*                     LISTEN
tcp6       0      0 *:ssh                   *:*                     LISTEN

# Mostrando los servicios escuchando a través de puertos UDP
usuario$ netstat --listening | grep udp
udp        0      0 *:1024                  *:*
udp        0      0 *:1026                  *:*
```

```
udp        0      0 *:826                   *:*
udp        0      0 *:bootpc                *:*
udp        0      0 *:mdns                  *:*
udp        0      0 *:sunrpc                *:*
udp        0      0 *:ipp                   *:*
udp6       0      0 *:1025                  *:*
```

Taller D.1: Muestre las conexiones TCP establecidas en su host. ¿Se pueden conocer las conexiones UDP establecidas?

La tabla de routing: Ejemplo:

```
# Mostrando la tabla de routing
usuario$ netstat --route
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.213.0   *               255.255.255.0   U         0 0          0 eth0
default         192.168.213.2   0.0.0.0         UG        0 0          0 eth0
```

Taller D.2: Muestre la tabla de routing usando `netstat` y `route`.

Información sobre los interfaces: Ejemplo:

```
# Mostrando la información sobre los interfaces de red
usuario$ netstat --interface
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500 0    191678      0      0      0    87617      0      0      0 BMRU
lo    16436 0     44677      0      0      0    44677      0      0      0 LRU
```

Taller D.3: Con el anterior comando tenemos una buena idea de la calidad de las transmisiones en la red local. ¿Qué campos miraría para conocer si hay algún problema físico (como ruido en la red)?

Estadísticas sobre los protocolos: Ejemplo:

```
# Mostrando las estadísticas de la red por protocolo
usuario$ netstat --statistics
Ip:
    236706 total packets received
    13 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    236693 incoming packets delivered
    132642 requests sent out
    3 outgoing packets dropped
Icmp:
    140 ICMP messages received
    1 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 125
        echo requests: 3
        echo replies: 12
    61 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 58
        echo replies: 3
Tcp:
    37328 active connections openings
```

```
    182 passive connection openings
    30011 failed connection attempts
    50 connection resets received
    4 connections established
    236131 segments received
    131732 segments send out
    371 segments retransmited
    0 bad segments received.
    15052 resets sent
Udp:
    328 packets received
    58 packets to unknown port received.
    0 packet receive errors
    455 packets sent
TcpExt:
    207 TCP sockets finished time wait in fast timer
    128 time wait sockets recycled by time stamp
    268 delayed acks sent
    5 delayed acks further delayed because of locked socket
    Quick ack mode was activated 18 times
    5665 packets directly queued to recvmsg prequeue.
    33578 of bytes directly received from prequeue
    168068 packet headers predicted
    41 packets header predicted and directly queued to user
    906 acknowledgments not containing data received
    7266 predicted acknowledgments
    6 congestion windows recovered after partial ack
    0 TCP data loss events
    334 other TCP timeouts
    4 times receiver scheduled too late for direct processing
    22 connections reset due to unexpected data
    18 connections reset due to early user close
    2 connections aborted due to timeout
```

Taller D.4: Pruebe el anterior comando. ¿Cómo incrementaría el número de paquetes ICMP recibidos?

Conexiones de tipo "mascarada": Las conexiones de tipo "masquerade" son las que se establecen cuando usamos la funcionalidad `ip_masquerade` del kernel con el objetivo de diseñar una NAT box con nuestro Linux. Ejemplo (cuando no se ha habilitado el `ip_masquerade`):

```
# Mostrando las conexiones masquerade. Para que este comando
# muestre información relevante el kernel ha debido ser configurado
# con la funcionalidad ip_masquerade.
usuario$ netstat --masquerade
netstat: no support for 'ip_masquerade' on this system.
```

Pertenencias multicast: Ejemplo:

```
# Mostrando los grupos multicast a los que estamos suscritos
usuario$ netstat --groups
IPv6/IPv4 Group Memberships
Interface       RefCnt Group
--------------- ------ ---------------------
lo              1      ALL-SYSTEMS.MCAST.NET
eth0            1      224.0.0.251
eth0            1      ALL-SYSTEMS.MCAST.NET
lo              1      ip6-allnodes
eth0            1      ff02::1:ff14:bcf6
eth0            1      ip6-allnodes
```

Taller D.5: Encuentre los grupos multicast a los que actualmente está apuntada su computadora.

D.3. Nmap

Nmap (Network MAPper) es una utilidad Open Source (y por tanto, no hay que pagar por usarla) para la exploración de redes y la auditoría de seguridad [7]. Actualmente se puede ejecutar (al menos) bajo los siguientes sistemas operativos: Linux (`http://www.linux.org/`http://www.linux.org/), Microsoft Windows (`http://www.microsoft.com/spain/windows/default.mspx`), FreeBSD (`http://www.freebsd.org/`), OpenBSD (`http://www.openbsd.org/`), Solaris (`http://www.sun.com/software/solaris/`), IRIX (`http://www.sgi.com/products/software/irix/`), Mac OS X (`http://www.apple.com/es/macosx/`), HP-UX (`http://www.hp.com/products1/unix/operating`), NetBSD (`http://www.netbsd.org/`), Sun OS (`http://en.wikipedia.org/wiki/SunOS`) y AmigaOS (`http://www.amiga.com/amigaos/`).

D.3.1. Instalación

Debian Linux:

```
root# apt-get install nmap
```

Fedora Core Linux:

```
root# yum install nmap nmap-frontend
```

Gentoo Linux:

```
root# emerge nmap
```

Taller D.6: Instale Nmap.

D.3.2. Utilización básica

Nmap puede utilizarse desde la línea de comandos (`nmap`) o desde una GUI (`xnmap`). Aquí interactuaremos sólo con la versión de consola y explicaremos algunos ejemplos muy básicos. Nmap es increiblemente rico en opciones y posibilidades que deben consultarse en el manual online (`http://insecure.org/nmap/man/`).

Sondeo de puertos TCP

El sondeo de puertos consiste en ver si pueden establecerse conexiones TCP con los diferentes puertos, y sirve para conocer los servicios activos que se basan en este protocolo. Este escaneo puede realizarse realizando conexiones estándar o un tipo de conexiones que en Nmap llama "SYN sigiloso" que no llega a establecer realmente las conexiones y por lo tanto no le aparecen al usuario de la máquina escaneada cuando "hace" un `netstat`.

El primer tipo (conexión estándar) puede realizarse como usuario normal y es el tipo de sondeo por defecto. El segundo (que utiliza el flag `-sS`) sólo puede realizarlo el usuario `root`. Ejemplos:

```
usuario$ nmap localhost

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-28 10:44 CET
Interesting ports on localhost (127.0.0.1):
Not shown: 1675 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
```

```
22/tcp  open  ssh
25/tcp  open  smtp
53/tcp  open  domain
953/tcp open  rndc

Nmap finished: 1 IP address (1 host up) scanned in 0.152 seconds

# Más info!!!
usuario$ nmap -v localhost

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-28 10:44 CET
Initiating Connect() Scan against localhost (127.0.0.1) [1680 ports] at 10:44
Discovered open port 21/tcp on 127.0.0.1
Discovered open port 25/tcp on 127.0.0.1
Discovered open port 53/tcp on 127.0.0.1
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 953/tcp on 127.0.0.1
The Connect() Scan took 0.04s to scan 1680 total ports.
Host localhost (127.0.0.1) appears to be up ... good.
Interesting ports on localhost (127.0.0.1):
Not shown: 1675 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
25/tcp  open  smtp
53/tcp  open  domain
953/tcp open  rndc

Nmap finished: 1 IP address (1 host up) scanned in 0.154 seconds

# Ahora sólo algunos puertos
usuario$ nmap -v -p 22,53 localhost

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-29 08:40 CET
Initiating Connect() Scan against localhost (127.0.0.1) [2 ports] at 08:40
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 53/tcp on 127.0.0.1
The Connect() Scan took 0.00s to scan 2 total ports.
Host localhost (127.0.0.1) appears to be up ... good.
Interesting ports on localhost (127.0.0.1):
PORT   STATE SERVICE
22/tcp open  ssh
53/tcp open  domain

Nmap finished: 1 IP address (1 host up) scanned in 0.110 seconds

# Ahora un rango de máquinas, sólo puerto 80
usuario$ nmap -v -p 80 193.147.118.128-255

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-29 08:45 CET
Machine 193.147.118.154 MIGHT actually be listening on probe port 80
Machine 193.147.118.131 MIGHT actually be listening on probe port 80
Machine 193.147.118.176 MIGHT actually be listening on probe port 80
Machine 193.147.118.170 MIGHT actually be listening on probe port 80
Machine 193.147.118.174 MIGHT actually be listening on probe port 80
Machine 193.147.118.192 MIGHT actually be listening on probe port 80
Machine 193.147.118.195 MIGHT actually be listening on probe port 80
Machine 193.147.118.196 MIGHT actually be listening on probe port 80
Machine 193.147.118.199 MIGHT actually be listening on probe port 80
```

```
Machine 193.147.118.220 MIGHT actually be listening on probe port 80
Machine 193.147.118.218 MIGHT actually be listening on probe port 80
Machine 193.147.118.217 MIGHT actually be listening on probe port 80
DNS resolution of 18 IPs took 0.00s.
Initiating Connect() Scan against 18 hosts [1 port/host] at 08:45
Discovered open port 80/tcp on 193.147.118.170
Discovered open port 80/tcp on 193.147.118.174
Discovered open port 80/tcp on 193.147.118.176
Discovered open port 80/tcp on 193.147.118.199
Discovered open port 80/tcp on 193.147.118.154
Discovered open port 80/tcp on 193.147.118.196
Discovered open port 80/tcp on 193.147.118.217
Discovered open port 80/tcp on 193.147.118.220
Discovered open port 80/tcp on 193.147.118.192
Discovered open port 80/tcp on 193.147.118.218
Discovered open port 80/tcp on 193.147.118.195
Discovered open port 80/tcp on 193.147.118.131
The Connect() Scan took 0.00s to scan 18 total ports.
Host 193.147.118.131 appears to be up ... good.
Interesting ports on 193.147.118.131:
PORT   STATE SERVICE
80/tcp open  http

Host 193.147.118.145 appears to be up ... good.
Interesting ports on 193.147.118.145:
PORT   STATE  SERVICE
80/tcp closed http

Host tornasol.ual.es (193.147.118.154) appears to be up ... good.
Interesting ports on tornasol.ual.es (193.147.118.154):
PORT   STATE SERVICE
80/tcp open  http

Host 193.147.118.170 appears to be up ... good.
Interesting ports on 193.147.118.170:
PORT   STATE SERVICE
80/tcp open  http

Host invernadero.ual.es (193.147.118.174) appears to be up ... good.
Interesting ports on invernadero.ual.es (193.147.118.174):
PORT   STATE SERVICE
80/tcp open  http

Host sauce.ual.es (193.147.118.176) appears to be up ... good.
Interesting ports on sauce.ual.es (193.147.118.176):
PORT   STATE SERVICE
80/tcp open  http

Host 193.147.118.192 appears to be up ... good.
Interesting ports on 193.147.118.192:
PORT   STATE SERVICE
80/tcp open  http

Host www.dgpaa.ual.es (193.147.118.195) appears to be up ... good.
Interesting ports on www.dgpaa.ual.es (193.147.118.195):
PORT   STATE SERVICE
80/tcp open  http

Host indalo.ual.es (193.147.118.196) appears to be up ... good.
```

```
Interesting ports on indalo.ual.es (193.147.118.196):
PORT   STATE SERVICE
80/tcp open  http

Host acacia.ual.es (193.147.118.199) appears to be up ... good.
Interesting ports on acacia.ual.es (193.147.118.199):
PORT   STATE SERVICE
80/tcp open  http

Host 193.147.118.201 appears to be up ... good.
Interesting ports on 193.147.118.201:
PORT   STATE  SERVICE
80/tcp closed http

Host 193.147.118.206 appears to be up ... good.
Interesting ports on 193.147.118.206:
PORT   STATE  SERVICE
80/tcp closed http

Host aer.ual.es (193.147.118.217) appears to be up ... good.
Interesting ports on aer.ual.es (193.147.118.217):
PORT   STATE SERVICE
80/tcp open  http

Host lsi.ual.es (193.147.118.218) appears to be up ... good.
Interesting ports on lsi.ual.es (193.147.118.218):
PORT   STATE SERVICE
80/tcp open  http

Host 193.147.118.220 appears to be up ... good.
Interesting ports on 193.147.118.220:
PORT   STATE SERVICE
80/tcp open  http

Host libras.ual.es (193.147.118.234) appears to be up ... good.
Interesting ports on libras.ual.es (193.147.118.234):
PORT   STATE  SERVICE
80/tcp closed http

Host desaveal.ual.es (193.147.118.238) appears to be up ... good.
Interesting ports on desaveal.ual.es (193.147.118.238):
PORT   STATE  SERVICE
80/tcp closed http

Host cabezon.ual.es (193.147.118.244) appears to be up ... good.
Interesting ports on cabezon.ual.es (193.147.118.244):
PORT   STATE  SERVICE
80/tcp closed http

Nmap finished: 128 IP addresses (18 hosts up) scanned in 1.317 seconds
```

Sonde del SO

Sólo como root. Nmap envía una colección de paquetes específico al sistema remoto para tratar de adivinar el sistema operativo que ejecuta. Ejemplo:

```
root# nmap -O localhost

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-28 10:48 CET
Interesting ports on localhost (127.0.0.1):
```

```
Not shown: 1675 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
25/tcp  open  smtp
53/tcp  open  domain
953/tcp open  rndc
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.0 - 2.5.20, Linux 2.5.25 - 2.6.8 or Gentoo 1.2\
 Linux 2.4.19 rc1-rc7, Linux 2.6.3 - 2.6.10

Nmap finished: 1 IP address (1 host up) scanned in 2.049 seconds
```

Taller D.7: Pruebe los anteriores comandos. Sea imaginativo y no enfade a nadie :-)

Sondeo de puertos UDP

Hay determinados servicios que sólo se ofrecen a través de UDP. Para descubirlos se utiliza este tipo de escaneo. Un ejemplo (sólo `root`):

```
usuario$ nmap -sU localhost

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-29 08:56 CET
Interesting ports on localhost (127.0.0.1):
Not shown: 1486 closed ports
PORT   STATE         SERVICE
53/udp open|filtered domain

Nmap finished: 1 IP address (1 host up) scanned in 1.288 seconds
```

Descubriendo los hosts encendidos en un rango de IPs

Ahora damos otro ejemplo donde escaneamos un rango de direcciones IPutilizando el mensaje Echo Request del protocolo ICMP. Ejemplo:

```
usuario$ nmap -sP 193.147.118.*

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-29 08:59 CET
Host 193.147.118.1 appears to be up.
Host 193.147.118.2 appears to be up.
Host 193.147.118.3 appears to be up.
Host 193.147.118.5 appears to be up.
Host 193.147.118.21 appears to be up.
Host 193.147.118.24 appears to be up.
Host 193.147.118.25 appears to be up.
Host 193.147.118.26 appears to be up.
Host 193.147.118.27 appears to be up.
Host 193.147.118.28 appears to be up.
Host 193.147.118.29 appears to be up.
Host 193.147.118.30 appears to be up.
Host 193.147.118.38 appears to be up.
Host indalog.ual.es (193.147.118.39) appears to be up.
Host 193.147.118.40 appears to be up.
Host 193.147.118.43 appears to be up.
Host 193.147.118.45 appears to be up.
Host europa.ace.ual.es (193.147.118.46) appears to be up.
Host 193.147.118.47 appears to be up.
Host 193.147.118.48 appears to be up.
Host 193.147.118.49 appears to be up.
```

```
Host vermeer.ace.ual.es (193.147.118.50) appears to be up.
Host iron.ace.ual.es (193.147.118.54) appears to be up.
Host dali.ace.ual.es (193.147.118.56) appears to be up.
Host gogh.ace.ual.es (193.147.118.57) appears to be up.
Host renoir.ace.ual.es (193.147.118.61) appears to be up.
Host caesarg.ace.ual.es (193.147.118.67) appears to be up.
Host 193.147.118.73 appears to be up.
Host davinci.ace.ual.es (193.147.118.77) appears to be up.
Host 193.147.118.80 appears to be up.
Host 193.147.118.81 appears to be up.
Host io.ace.ual.s (193.147.118.89) appears to be up.
Host 193.147.118.92 appears to be up.
Host 193.147.118.131 appears to be up.
Host 193.147.118.145 appears to be up.
Host tornasol.ual.es (193.147.118.154) appears to be up.
Host 193.147.118.170 appears to be up.
Host invernadero.ual.es (193.147.118.174) appears to be up.
Host sauce.ual.es (193.147.118.176) appears to be up.
Host 193.147.118.192 appears to be up.
Host www.dgpaa.ual.es (193.147.118.195) appears to be up.
Host indalo.ual.es (193.147.118.196) appears to be up.
Host acacia.ual.es (193.147.118.199) appears to be up.
Host 193.147.118.201 appears to be up.
Host 193.147.118.206 appears to be up.
Host aer.ual.es (193.147.118.217) appears to be up.
Host lsi.ual.es (193.147.118.218) appears to be up.
Host 193.147.118.220 appears to be up.
Host libras.ual.es (193.147.118.234) appears to be up.
Host desaveal.ual.es (193.147.118.238) appears to be up.
Host cabezon.ual.es (193.147.118.244) appears to be up.
Nmap finished: 256 IP addresses (51 hosts up) scanned in 1.892 seconds
```

Taller D.8: Encuentre los hosts encendidos en su red privada.

D.4. Nessus

Nessus [2] es una herramienta para en análisis de la vulnerabilidad en redes. Ejecuta una colección de "ataques" contra un host con el origen de conocer los posibles agujeros de seguridad en un host que está *encendido y conectado a Internet*. Si Nmap es la herramienta por excelencia para conocer qué servicios pueden ser "atacados", Nessus es la utilidad que permite conocer qué agujeros de seguridad pueden tener dichos servicios.

Nessus es un programa compuesto por un cliente y un servidor y (como es de esperar) ambos programas pueden ejecutarse en hosts diferentes. El servidor realiza todo el trabajo de escaneo que especifica el cliente. Un servidor puede ser usado por varios clientes y generalmente los clientes además tienen la posibilidad de utilizar varios servidores en diferentes sesiones.

D.4.1. Instalación (cliente y servidor)

Debian Linux:

Descargar los ficheros Nessus-<ultima_version>.deb y NessusClient-<ultima_version>.deb de http://www.nessus.org/download/.

```
root# dpkg -i Nessus-<ultima_version>.deb
root# dpkg -i NessusClient-<ultima_version>.deb
```

Fedora Core Linux:

Descargar el fichero Nessus-<ultima_version>.rpm y NessusClient-<ultima_version>.rpm de http://www.nessus.org/download/.

```
root# rpm -Uvh Nessus-<ultima_version>.rpm
root# rpm -Uvh NessusClient-<ultima_version>.rpm
```

Taller D.9: Instale Nessus.

D.4.2. Configuración del servidor

Crear un usuario en el servidor de Nesus:

```
root# /opt/nessus/sbin/nessus-add-first-user
```

Se nos pregutarán tres cosas: (1) un nombre de usuario, (2) la forma de autentificación (elegir "pass"-word) y (3) un password.

Crear un certificado de usuario: Sirve para aseguranos que nos conectamos con el servidor que estamos instalando cuando accedamos a él desde un cliente Nessus.

```
root# /opt/nessus/sbin/nessus-mkcert
```

Registrarnos como usuario de Nessus: Cuando nos hemos descargado el paquete Nessus se nos ha enviado a la dirección de correo especificada un código de activación. Activar la descarga de los plugins utilizando:

```
root# /opt/nessus/sbin/nessus-fetch --register <el_código_de_activación>
```

Actualizar los plugins: Para utilizar los últimos ataques que existen es recomendable mantener actualizada la lista de plugins que Nessus va a utilizar:

```
root# /opt/nessus/sbin/nessus-update-plugins
```

Este paso debería realizarse periódicamente (una vez a la semana más o menos) para disponer de los últimos plugins. Un detalle importante. Nessus puede ser utilizado bajo una licencia comercial (pagando) y bajo una licencia libre. La ventaja de utilizar la versión comercial es que tendremos acceso a los plugins una semana antes que para la versión libre.

Ejecutar el servidor: `root# /etc/init.d/nessus start`

Taller D.10: Configure el servidor Nessus.

D.4.3. Uso del cliente

El cliente Nessus tiene un interfaz de texto (`nessus-text`) y otro gráfico (`nessus`), mucho más fácil de utilizar. Para ejecutarlo:

```
usuario$ /usr/X11R6/bin/NessusClient
```

El cliente puede ser ejecutado en cualquier host, diferente o no del que ejecuta el servidor. Los pasos básicos que deberemos realizar son:

1. Acceder al servidor (usaremos el login y password anteriormente creados).
2. Seleccionar los plugins que deseamos ejecutar.
3. Seleccionar el host "Target" que vamos a escanear.
4. Pinchar en el botón de iniciar el escaneo.

Tras el escaneo obtendremos una lista de servicios disponibles en el host escaneado y comentarios acerca de la vulnerabilidad de dichos servicios.

Taller D.11: Pruebe alguno de los "exploits" que posee Nessus para comprobar la seguridad de alguno de los servicios que ofrece su computadora.

Apéndice E

Filtrado de paquetes en Linux

El filtrado de paquetes es una técnica que consiste, normalmente, en descartar aquellos paquetes que llegan (salen) hasta (desde) nuestro host a un puerto o con un contenido no deseado. En el primer caso los paquetes se filtran atendiendo al puerto al que van dirigidos (por ejemplo, todo lo que no vaya al puerto 80 debería ser ignorado si un servidor sólo debería ofrecer servicio Web). En el segundo el payload de los paquetes es analizado y dependiendo de este pueden ser eliminados (por ejemplo, todo aquel paquete que utilice el protocolo del eMule debería ser destruído). Otra situación muy común puede ser no transmitir paquetes que vengan o vayan desde o hacia un determinado rango de direcciones IP.

El filtrado de paquetes se realiza normlamente en los cortafuegos. Por ejemplo, en la universidad se filtran todos los paquetes de entrada que no provengan del tráfico Web o SSH, y se analiza el contenido de los paquetes para eliminar el tráfico generado por la mayoría de las aplicaciones P2P de compartición de ficheros (como eMule). Sin embargo, existen situaciones donde esto no es suficiente. ¿Qué pasaría si los paquetes no deseados llegan desde otro host situado en la misma red que la nuestra y cuyo tráfico no es filtrado por el cortafuegos?

E.1. En Linux el filtrado de paquetes se realiza a nivel del kernel

Para controlar cómo se filtran los paquetes el kernel utiliza una tabla de filtrado de paquete y la utilidad que permite manipularla se llama `iptables`. Por tanto, el kernel ha debido ser compilado con esta opción o habrá que cargar el módulo correspondiente. Finalmente, la utilidad de la que hablamos deberá estar instalada.

E.1.1. Habilitando el kernel

El kernel se habilita para filtrar paquetes cuando cargamos el módulo `iptable_filter`. Esto se hace escribiendo:

```
root# insmod iptable_filter
```

o modificando el fichero correspondiente para que dicha carga se haga cuando la máquina arranca.

Si por lo que sea no queremos habilitar el kernel cargando el módulo correspondiente sólo nos queda la opción de compilar uno nuevo y usarlo.

E.1.2. Instalación de `iptables`

Debian Linux:

```
root# apt-get install iptables
```

Fedora Core Linux:

```
root# yum install iptables
```

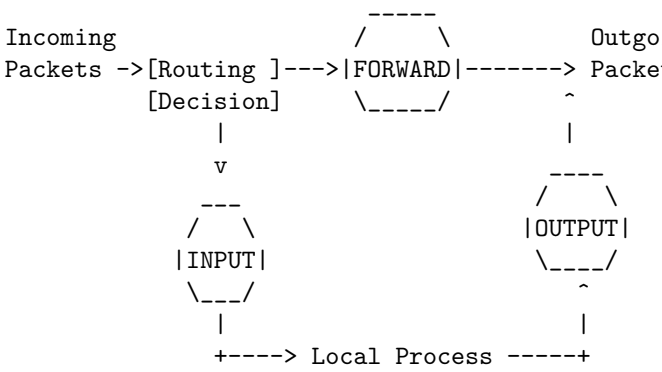
Gentoo Linux:

```
root# emerge iptables
```

E.2. El proceso de filtrado

El kernel maneja tres listas de reglas en su tabla de filtrado de paquetes. Las listas se llaman también *cadenas* (chains, en inglés). Bien, en concreto las cadenas que manejaremos se llaman *input*, *output* and *forward*.

Input controla los paquetes que entran en el host, *output* los que son generados en el host y lo avandonan y *forward* los que vienen desde el exterior pero no van dirigidos a ningún proceso del host. Una figura (`http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-6.html`) aclara el tema:



Así, cada vez que un paquete alcanza una cadena se utilizan una lista de reglas para saber si abandonamos (drop, en inglés) o aceptamos el paquete.

Finalmente indicar que la lista forward se utiliza sólo cuando configuramos nuestro host como un router (o NAT) con cortafuegos.

E.3. Uso de `iptables`

E.3.1. Mostrar la lista de reglas de una cadena

```
root# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

root# iptables --verbose --list INPUT
Chain INPUT (policy ACCEPT 115 packets, 15731 bytes)
pkts bytes target     prot opt in     out     source               destination
```

E.3.2. Crear una nueva cadena

Aparte de las tres cadenas que existen por defecto (INPUT, OUTPUT y FORWARD), el usuario puede crear cadenas nuevas:

```
root# iptables --new new_chain
```

Así, cuando un paquete "acierta" con una regla de una cadena, la acción puede ser que pase a ser procesado por la nueva cadena. Esto se hace para reducir el número de reglas por cadena y que así sean más fáciles de manejar.

Si ninguna de las reglas de la nueva cadena procesa el paquete, el proceso de chequeo continúa por la siguiente regla que referenciaba a la nueva cadena.

E.3.3. Resetear las estadísticas de una cadena

Pone a cero los contadores de la cadena:

```
root# iptables --zero chain_name
```

E.3.4. Cambiar el comportamiento por defecto de una cadena

Por defecto las cadenas `INPUT`, `OUTPUT` y `FORWARD` están en modo `ACCEPT`, significando que dejan pasar los paquetes. Esto puede modificarse cambiando lo que define como "policy" de la cadena. Un ejemplo que evita que podamos transmitir ningún paquete:

```
root# iptables --policy OUTPUT DROP
```

Los posibles modos de una regla o de una cadena son:

ACCEPT: Aceptar el paquete.

DROP: Destruir el paquete.

REJECT: Destruir el paquete pero enviando un paquete ICMP de puerto inalcanzable.

QUEUE: Pasar al paquete al espacio de usuario. Se utiliza para procesar los paquetes que llegan por un determinado proceso, independientemente de que vayan o no dirigidos a él.

RETURN: Retornar (sin chequear más reglas) a la canena que nos dió el control.

E.3.5. Vaciar una cadena

Para eliminar las reglas de una cadena, escribir:

```
root# iptables --flush chain_name
```

E.3.6. Borrar una cadena vacía

Podemos borrar una cadena vacía con:

```
root# iptables --delete-chain chain_name
```

E.3.7. Añadir una nueva regla a una cadena

En el siguiente ejemplo se filtra todo aquel paquete que vaya dirigido a la dirección IP `27.0.0.1`:

```
root# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms

root# iptables --append INPUT --source 127.0.0.1 --protocol icmp --jump DROP
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss

root# iptables --verbose --list INPUT
Chain INPUT (policy ACCEPT 169 packets, 21274 bytes)
pkts bytes target     prot opt in     out     source               destination
1    84    DROP       icmp --  any    any     localhost.localdomain anywhere
```

E.3.8. Borrar una regla de una cadena

Borraremos la regla que acabamos de crear con:

```
root# iptables --delete INPUT 1
```

O también con:

```
root# iptables --delete INPUT --source 127.0.0.1 --protocol icmp --jump DROP
```

Nótese que los argumentos son idénticos a los que utilizamos para crear la regla.

E.3.9. Añadiendo reglas más complejas

Algunos de los parámetros con los que podemos jugar son (véase [12] y [1] para más información):

Las direcciones IP de origen y destino: Por ejemplo, podemos especificar sólo los paquetes que van dirigidos a la red local puedan salir del host:

```
root# iptables --append OUTPUT --destination ! 192.168.213.0/24 --jump DROP
```

El protocolo utilizado: Ejemplo, sólo el tráfico Web de entrada está permitido:

```
# Nótese el flag --syn que ‘‘matches’’ sólo los paquetes TCP
# de establecimiento de conexión
root# iptables --append INPUT --protocol tcp --syn --dport ! http --jump REJECT
```

El puerto utilizado: Sólo aquellos paquetes que vayan al puerto 22 (SSH) serán aceptados:

```
root# iptables --append INPUT --protocol tcp --syn --dport ! 22 --jump ACCEPT
```

El interface de red usado: Esto es útil cuando estamos creando un router o un NAT. Ejemplo, aceptar todo lo que venga de `eth0`:

```
root# iptables --append INPUT --in-interface eth0 --jump ACCEPT
```

E.3.10. Salvando y restaurando las cadenas

Existen dos comandos para hacer esto: `iptables-save` y `iptables-restore`. Ambas leen y escriben la entrada y salida estándares. Ejemplos:

```
root# iptables-save > iptables.txt

root# iptables-restore < iptables.txt
```

Taller E.1: Configure `iptables` para aceptar sólo el tráfico Web.

Apéndice F

Captura de paquetes usando Wireshark

Toda la comunicación que se produce en redes de computadoras se realiza a base de paquetes de datos. Unas veces estos paquetes son generados por nuestra computadora, otras van dirigidos a ésta y en otras ocasiones los paquetes simplemente pasan cerca de ella y es posible capturarlos.

La herramienta que permite capturar los paquetes se llama también un *packet sniffer* (traduciendo de alguna forma: "un olisqueador de paquetes"). Existen muchos sniffers, entre los que podemos destacar a `tcpdump` (`http://www.tcpdump.org`) y Ethereal (`http://www.ethereal.com`) (llamado ahora Wireshark (`http://www.wireshark.org`). El primero tiene una interfaz texto. El segundo gráfica.

F.1. ¿Quién usa un sniffer?

Los sniffers son usados por los hackers, pero también por los programadores de aplicaciones de red y por los administradores de las redes que desean depurar algún problema. Nosotros los usaremos en estas prácticas fundamentalmente por los dos últimos motivos.

F.2. Sniffers y analizadores de paquetes

Muchas veces (como ocurre con Wireshark) el sniffer viene acompañado de un analizador de paquetes (o viceversa). El sniffer se encarga de capturarlos y el analizador de comprender cómo están los datos organizados en los paquetes y presentarlos al usuario de alguna determinada forma. El sniffer puede capturar todos los paquetes que pasan por alguno de los adaptadora de red de la computadora pero no necesita para ello saber cómo se han ido encapsulando los datos en función de los diferentes protocolos que se han utilizado para construir estos paquetes. El analizador sí que debe entender de protocolos y cómo la encapsulación se ha realizado.

F.3. Instalación de Wireshark

Microsoft Windows: Acceder a la Web de Wireshark (`http://www.wireshark.org/download.html`) y descargar el instalador de un mirror.

Debian Linux:

```
root# apt-get install wireshark
```

Fedora Core Linux:

```
root# yum install wireshark
```

Gentoo Linux:

```
root# emerge wireshark
```

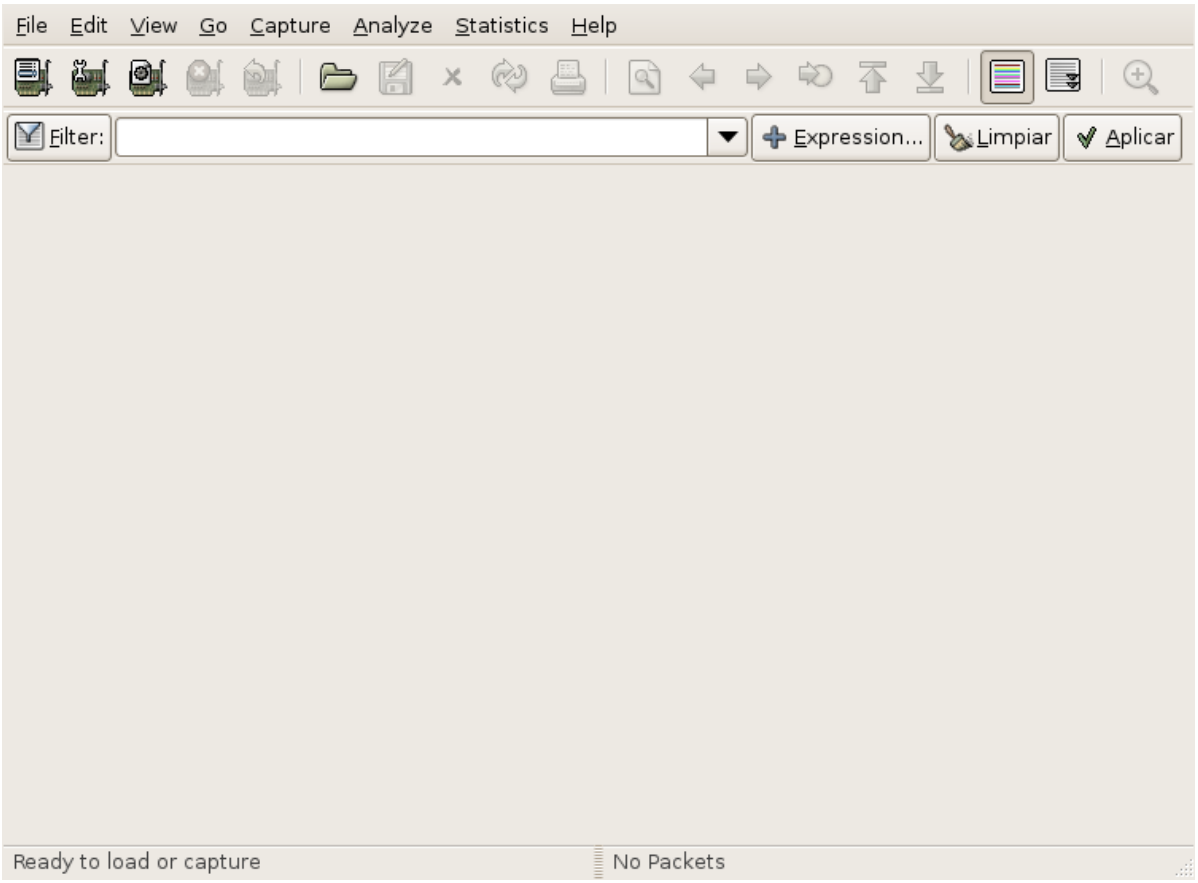


Figura F.1: Interface de usuario de la aplicación Wireshark.

Figura F.2: Selección del interface de red a "sniffear".

F.4. El interfaz gráfico de Wireshark

Descrito de arriba a abajo, el interfaz gráfico de Wireshark (véase la Figura F.1) consta de 6 partes principales:

1. La zona de menús que permite lanzar una captura y controlar cómo se lleva a cabo.
2. La zona de botones con las principales acciones que podemos realizar. Cada uno de estos botones genera una descripcion en texto que aparece cuando dejamos el puntero del ratón un tiempo sin moverlo (tooltip).
3. La entrada para especificar un filtro. Con este elemento podemos controlar los paquetes que aparecen (que son un subconjunto de los que se capturan) en la lista de paquetes capturados.
4. La lista de los paquetes filtrados. En ella aparece por cada paquete una línea diferente. En cada una de ellas aparecen los siguientes campos:
   a) El índice del paquete en la sesión de captura.
   b) El instante en que se ha capturado el paquete.
   c) La dirección IP del interface de red del que ha partido el paquete.
   d) La dirección IP del interface de red al que va dirigido el paquete.
   e) El protocolo de más alto nivel que es usado por el paquete.
   f) Información extra sobre el contenido del paquete.

   Los paquetes pueden ordenarse por cualquiera de estos campos pinchando con el ratón en el encabezado de la columna correspondiente.
5. Los detalles sobre la cabecera del paquete seleccionado en la lista de paquetes. Entre estos detalles figura información sobre el frame ethernet y el datagrama IP que contiene este paquete. La cantidad de información mostrada puede extenderse o minimizarse pinchando en los triángulos que aparecen.
6. El contenido del paquete (incluída la cabecera) en hexadecimal y ASCII.

F.5. Capturando paquetes con Wireshark

Para capturar paquetes hay que realizar los siguientes pasos:

1. Ejecutar Wireshark como `root`. Si ejecutó el sistema gráfico como un usuario diferente (de `root`), ejecute en un shell gráfico el comando:

   ```
   usuario$ xhost +
   ```

   Acceda como `root` al sistema escribiendo:

   ```
   usuario$ su -
   ```

   y una vez que haya accedido, indique que el display usado por el `root` es el mismo que el del usuario:

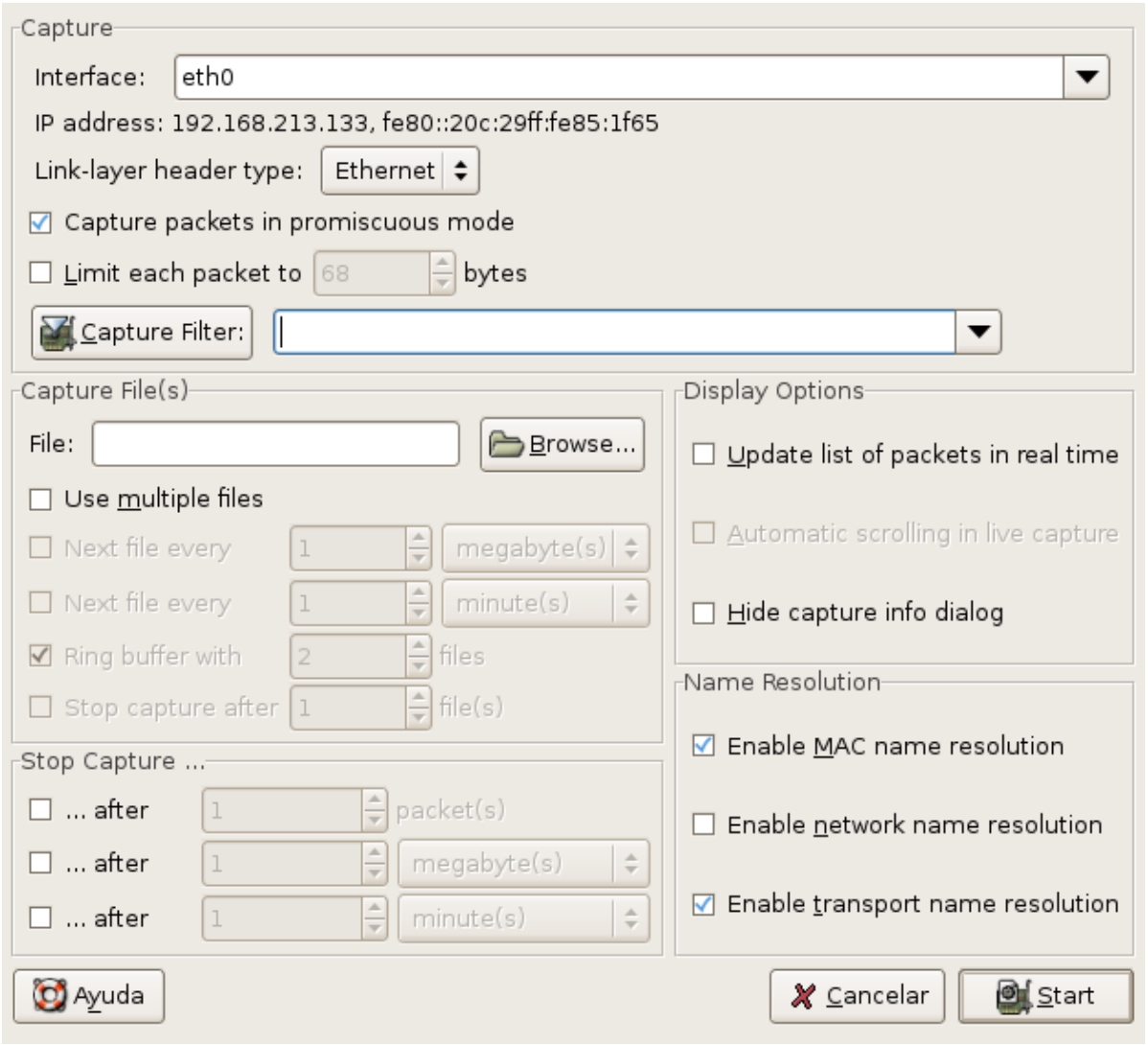


Figura F.3: Control de las opciones de captura.

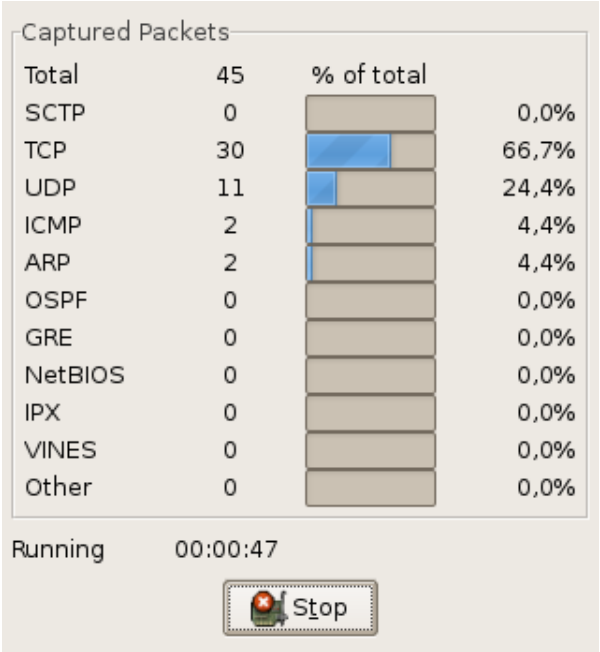


Figura F.4: Estadísticas de la captura.

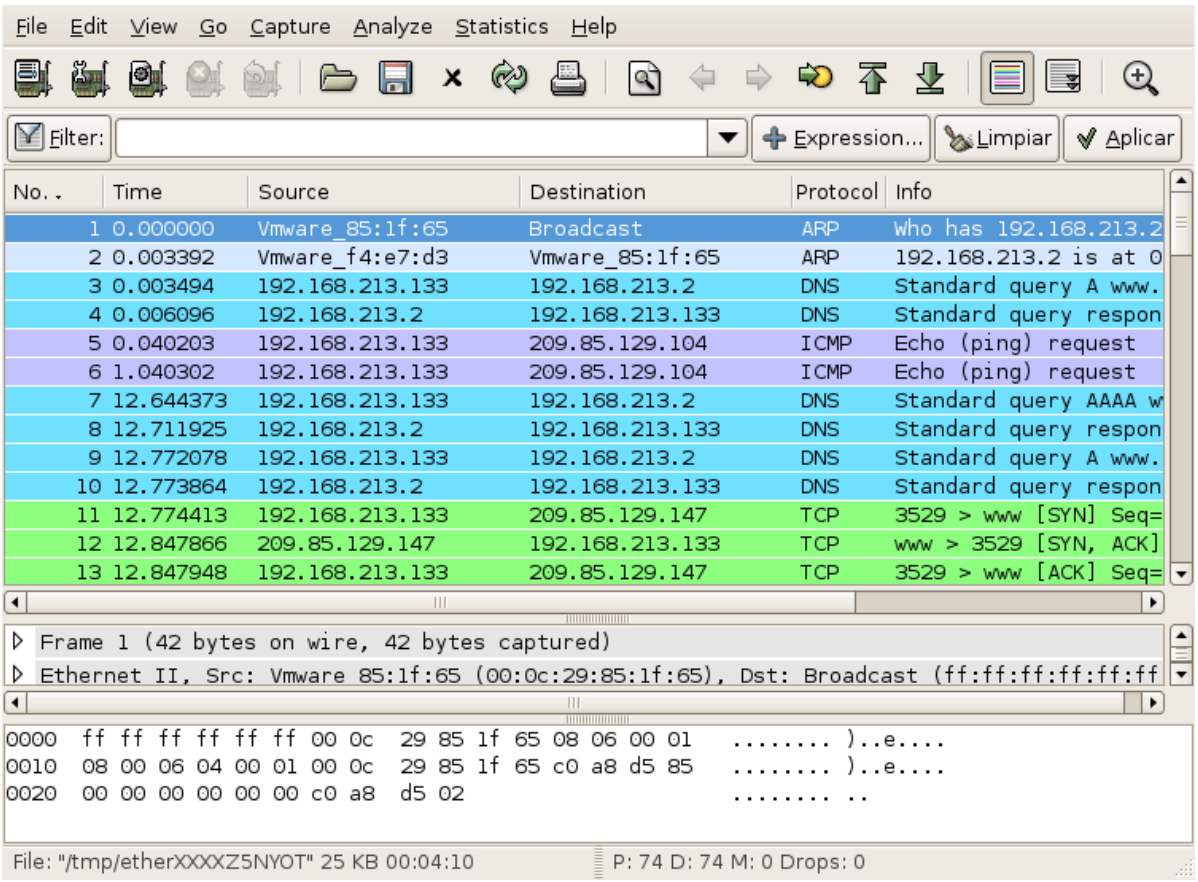


Figura F.5: Wireshark mostrando todos los paquetes capturados.

```
root# export DISPLAY=:0.0
```

2. Seleccionar el interface de red del que desea capturar. Esto puede hacerse pinchando en el botón que está situado bajo el menu *File* o seleccionado el menu *Capture -¿Interfaces*. Debería aparecer una ventana con la lista de adaptadores de red. Seleccione el adaptador que nos conecta con Internet (generalmente `eth0`). Véase la Figura F.2.

3. Definir las opciones de captura pulsando el botón *Options*, si fuera necesario modificar las opciones por defecto. Aparecerá una ventana (Figura F.3) que permite definir los siguientes parámetros:

   a) Una nueva oportunidad para definir el interface de red del que vamos a capturar.

   b) La tecnología física de red que utilizamos (generalmente Ethernet).

   c) El tamaño máximo de los paquetes.

   d) El filtro de captura (muy útil cuando vamos a capturar durante mucho tiempo y sólo estamos interesados en un tipo concreto de paquetes).

   e) El fichero que almacenará la captura su decidimos almacenarla en disco. Es posible definir multiples ficheros.

   f) Los parámetros de fin de la captura.

   g) Los parámetros de visualiación.

4. Pulsar el botón *Start* (este botón aparece también bajo el menú *View*). Aparecerá una ventana (véase la Figura F.4) que indica el número de paquetes capturados para unos cuantos protocolos muy frecuentes. Esta ventana contiene un botón *Stop* que permite detener la captura.

5. Pulsar el botón *Stop*. La ventana con la estadística de la captura se cerrará y aparecerá la lista de paquetes capturados (Figura F.5).

F.6. Filtrado de los paquetes capturados

Generalmente el número de paquetes capturados en mucho mayor que el que nos interesa. Una forma sencilla de trabajar sólo con los paquetes que desamos es usar el campo *Filter:* y a continuación pulsar en el botón *Aplicar*. Por ejemplo, en la Figura F.6 se muestran sólo los paquetes generados por el protocolo HTTP.

F.7. Ordenación de los paquetes capturados

En algunas situaciones puede ser interesante ordenar los paquetes siguiendo otro orden distinto del usado por defecto (el temporal). Esto puede hacerse pulsando con el botón izquierdo del ratón en la columna por la que deseamos ordenar. Por ejemplo, en la Figura F.7 se muestran sólo los paquetes generados por el protocolo HTTP, ordenados por la dirección IP fuente.

F.8. Análisis de los paquetes

Una de las principales utilidades de Wireshark radica en la facilidad de uso de cara a determinar cómo se ha realizado el proceso de encapsulamiento. Para ver cómo se ha ido realizando, en la ventana central aparecen las distintas cabeceras que se han generado. Podemos conocer el contenido concreto de cada una de ellas pulsando con el botón derecho del ratón en el triángulo asociado. Al situarnos sobre él cambiará de blanco a negro y al pulsar, aparecerá el contenido de la cabecera. Por ejemplo, en la Figura F.8 se ha expandido la cabecera HTTP.

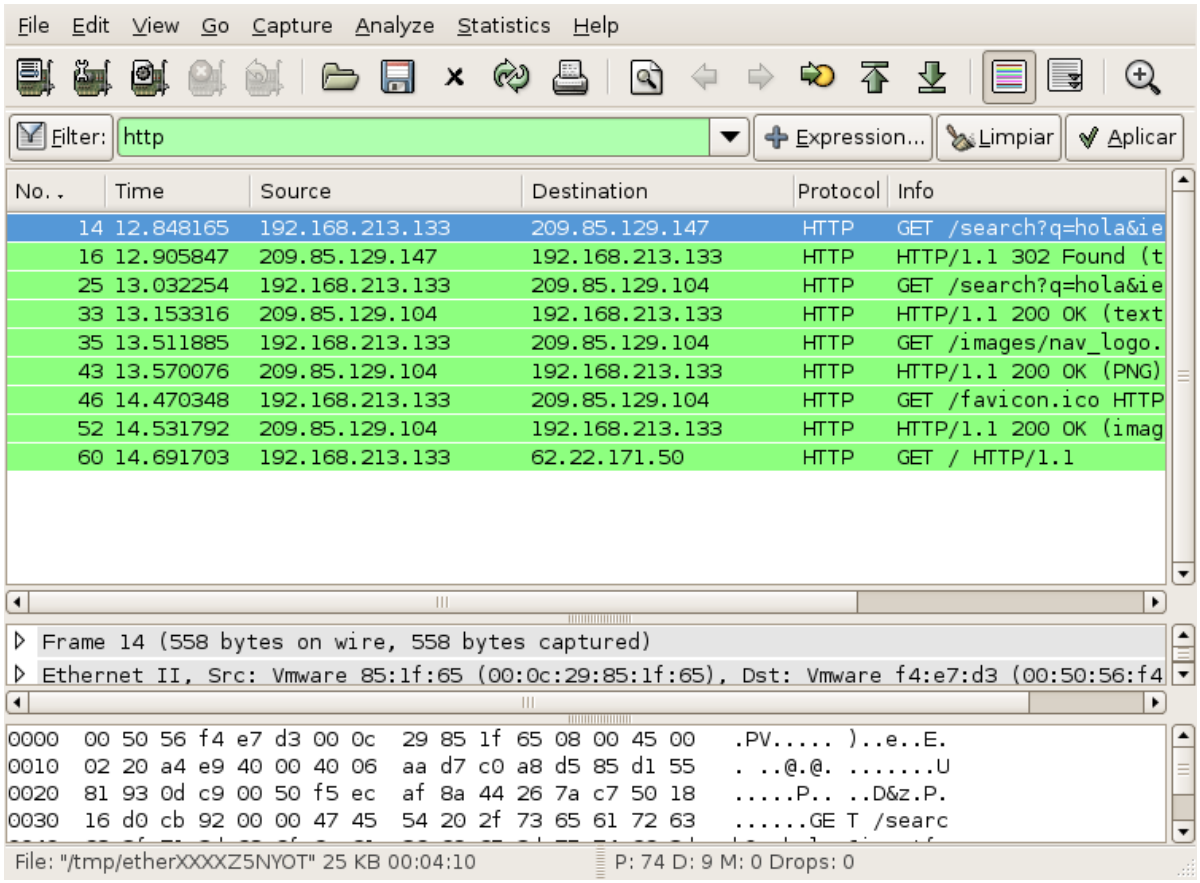


Figura F.6: Wireshark mostrando sólo los paquetes HTTP capturados.

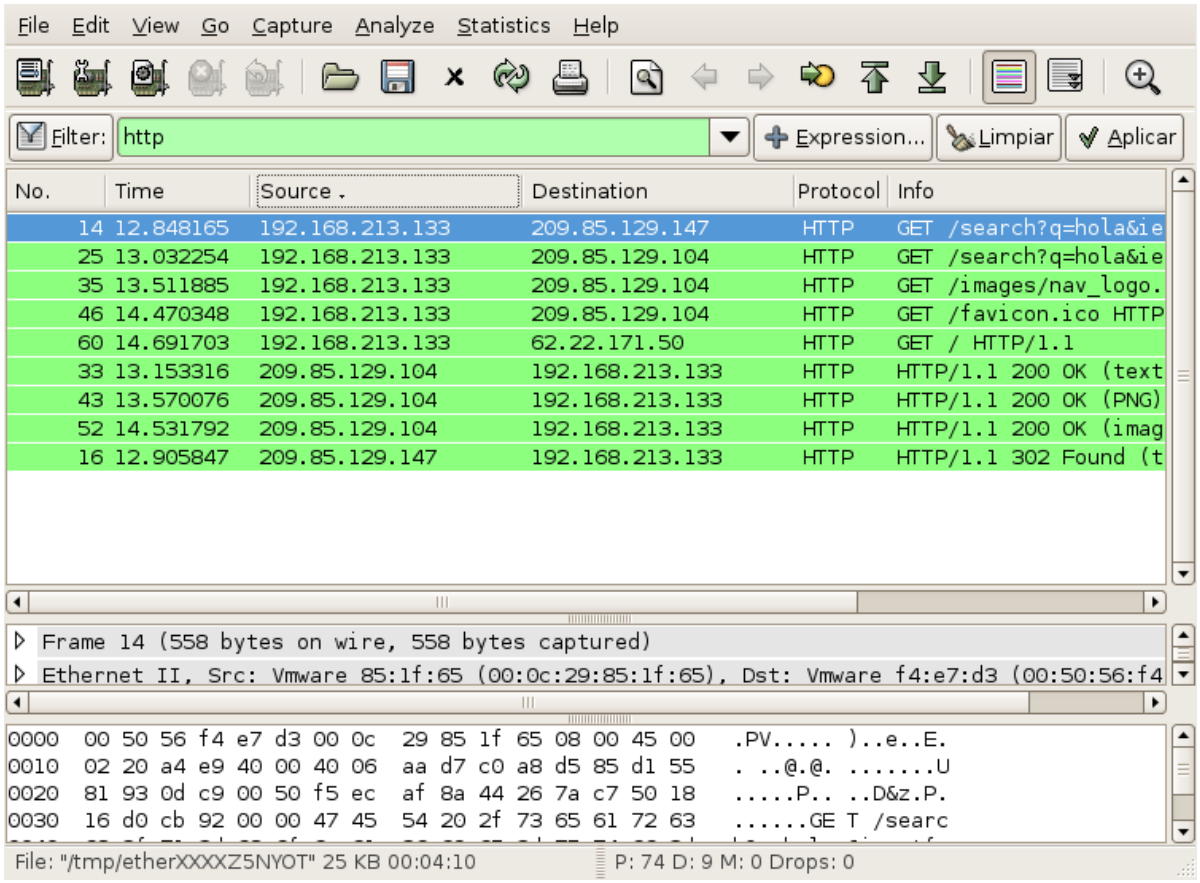


Figura F.7: Wireshark mostrando sólo los paquetes HTTP capturados y ordenados por la dirección IP fuente.

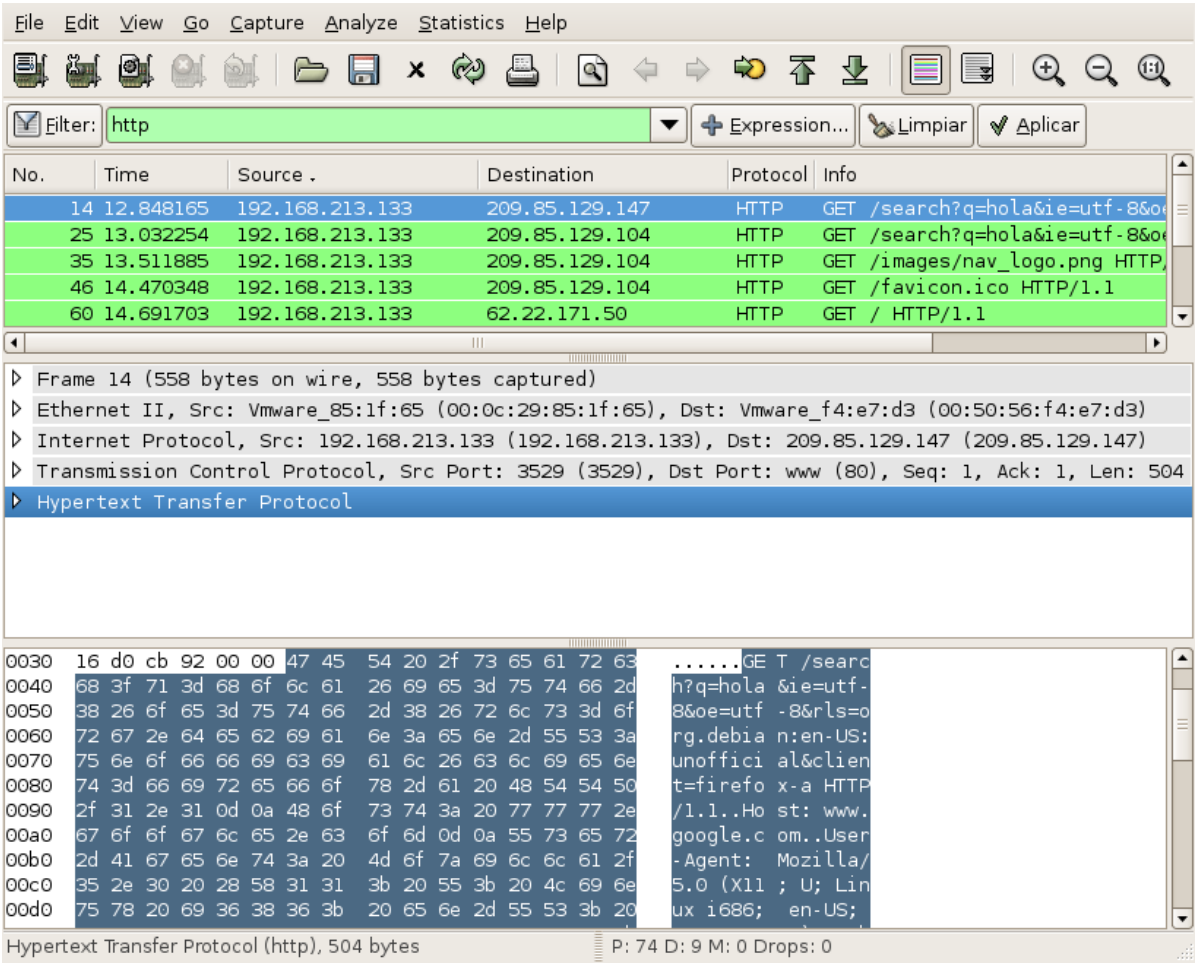


Figura F.8: Expansión de la cabecera HTTP de un paquete capturado.

Apéndice G

Distribución de ficheros usando Bittorrent

Bittorrent es una aplicación para compartir ficheros desarrollada por Bram Cohen. Bittorrent es semejante al Ftp, en el sentido en que hay un servidor de ficheros y un conjunto de clientes que acceden a él para descargarselos. Sin embargo, Bittorrent permite que los clientes hagan a su vez de servidores una vez que una parte de los ficheros ya están en su poder. De esta manera, al existir muchos servidores concurrentes, la distribución de los ficheros se acelera de una forma impresionante. Por este motivo, Bittorrent es actualmente el sistema de compartición de ficheros más utilizado.

G.1. Arquitectura de Bittorrent

En Bittorrent hay dos elementos principales:

1. El tracker. Es estrictamente un servidor, aunque no de ficheros. Se engarga de conocer a un subconjunto de los peers que están compartiendo un recurso (en Bittorrent se pueden compartir ficheros o conjuntos de ficheros, así que usaremos el término recurso para referirnos a ambos casos). Téngase que en cuenta que el número de peers que comparte un recurso no está limitado. El tracker sirve de nodo de acceso inicial a la red de compartición. Cuando un peer quiere comenzar a descargarse un recurso, o a comenzar a compartir un recurso ("publicar" un nuevo fichero, por ejemplo), debe ponerse en contacto con el tracker.

2. El peer. Funciona como cliente y como servidor, dependiendo del caso:

   a) Cliente: Cuando se descarga un recurso y por la razón que sea, no lo reenvía hacia otro peer.

   b) Servidor y cliente: Cuando comparte el recurso que se está descargando con otros peers. Es la situación más común.

   c) Estrictamente como servidor: Cuando ya ha terminado de descargarse el recurso o cuando él es la fuente inicial de recurso. En este caso el peer suele llamarse seed (semilla). Nótese que un seed siempre tiene una copia completa del recurso.

G.2. Funcionamiento básico de Bittorrent

Cuando un seed decide compartir un recurso lo particiona en bloques de un determinado tamaño. Dichos bloques son la unidad mínima de compartición en el sentido en que hasta que un peer no dispone al menos de un bloque, no comienza a actuar como servidor.

Los peers mantienen un número bastante bajo (que suele ser 4) de conexiones con otros peer cuando comparte un recurso. Bittorrent trata siempre de establecer conexiones con aquellos peers que más sirven. Como las conexiones son bidireccionales, aquellos peers que más sirven nos seleccionarán a nosotros si nosotros también somos unos buenos servidores. Por tanto, nos conectaremos con los mejores servidores si nosotros somos un buen servidor. Esta es una de las claves del superior rendimiento que posee Bittorrent frente a otros sistemas de compartición de archivos.

Para distribuir los recursos se utiliza un fichero con la extensión ".bittorrent". En dicho fichero figura el nombre de el/los archivo/s distribuido/s y una URL de la forma:

```
protocol://tracker:port/directory
```

donde `protocol` es el nombre del protocolo usado para realizar las peticiones al servidor (generalmente `http`, aunque también es posible `udp`), `tracker` es la dirección IP o el nombre del tracker, `port` es el puerto en el que escucha el servicio tracker y `directory` es el nombre de un directorio que el tracker utiliza para anunciar los recursos ("torrents").

G.3. Uso de un cliente

Si deseamos descargar un determinado recurso necesitamos dos cosas: (1) el fichero `.torrent` y (2) un cliente Bittorrent.

El fichero `.torrent` es muy pequeño (unos pocos cientos de bytes) y suele distribuirse mediante servidores Web, correos electrónicos, etc. También existen buscadores de ficheros `.torrent` (como por ejemplo, http://www.torrentscan.com/) que podemos utilizar para localizar un determinado recurso a partir del nombre esperado del mismo.

Clientes Bittorrent hay muchos (Azureus http://azureus.sourceforge.net/, μTorrent http://www.utorrent.com/, Transmission http://www.transmissionbt.com/, ...) aunque debemos destacar al original Bittorrent http://bitconjurer.org/BitTorrent/. En este pequeño manual aprenderemos a compartir recursos usando este último. Así, por ejemplo, para descargar un recurso bajo linux (tras instalar el paquete `bittorrent`), escribiremos:

```
peer$ btdownloadcurses recurso.torrent
```

G.4. Uso de un servidor

Cuando deseamos publicar o compartir un nuevo recurso necesitamos dos cosas: (1) un tracker y (2) crear el fichero `.torrent` asociado al recurso. El tracker pondrá en contecto a los diferentes peers y el fichero `.torrent` indicará cómo hacer esto.

G.4.1. Instalación de un tracker

Necesitaremos un host público con el paquete `bittorrent` instalado. Generalmente tras instalar este paquete se lanza el servicio correspondiente (`/etc/init.d/bittorrent`). El fichero de configuración de este servicio lo encontramos en `/etc/default/bittorrent`. Ahí podremos controlar varios aspectos del servidor, como por ejemplo, el puerto de escucha. Resumiendo, para instalar el tracker hacer:

```
tracker# apt-get install bittorrent
```

Que instalará y lanzará el servidor. También es posible lanzar el servidor a mano usando:

```
tracker$ bttrack --port 6969 --dfile dstate
```

`dstate` es el nombre del archivo de log en formato binario y 6969 sería el puerto de escucha.

Finalmente, existe un conjunto de trackers públicos que podemos utilizar si no disponemos acceso a un host público para hacer lo anterior. Algunos ejemplos son:

```
http://open.tracker.thepiratebay.org/announce
http://www.torrent-downloads.to:2710/announce
http://denis.stalker.h3q.com:6969/announce
udp://denis.stalker.h3q.com:6969/announce
http://www.sumotracker.com/announce
```

G.4.2. Generación del fichero `.torrent`

Necesitaremos alguna utilidad para hacer esto. La mayoría de los clientes permiten hacerlo fácilmente. Usando el paquete `bittorrent` escribiremos:

```
seed$ btmakemetafile camino/al/recurso protocol://tracker:port/directory

# Un ejemplo más real:
ssed$ btmakemetafile login_fortunes http://193.147.118.81:6969/announce
```

donde `recurso` es el nombre del fichero o del directorio que queremos compartir, `protocol` es el protocolo a usar (generalmente `http`), `tracker` el la dirección IP o el nombre del servidor que ejecuta el tracker, `port` es el puerto de escucha del tracker y `directory` es el nombre del directorio que el tracker va a utilizar para anunciar dicho recurso.

Este comando generará el fichero `.torrent` que contiene dicha URL, el nombre de el/los fichero/s compartido/s, el tamaño del bloque y otra información que se utiliza saber si los bloques se transmiten correctamente. Dependiendo del tamaño del recurso a compatir y de la potencia de la máquina, `btmakemetafile` empleará más o menos tiempo porque ha de recorrer el recurso completo para generar el fichero `.torrent`. Esto hace además que el contenido del recurso no puede ser variable, es decir, si cambia su contenido debe regenerarse el fichero `.torrent` asociado.

G.4.3. Activación del seed

Para que un recurso sea compartido con éxito al menos un peer debe tener una copia completa de él. Para formar parte de la red de compartición como seed, tras generar el fichero `.torrent` sólo hay que escribir:

```
seed$ btdownloadcurses recurso.torrent
```

en el directorio donde está alojado el recurso `recurso`. Es decir, el mismo comando que usaríamos si no fuéramos un seed.

Apéndice H

Códigos fuente

H.1. add.c

```
/*
 * add.c -- Suma dos señales.
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/add.c
 *
 * Compilar escribiendo:
 * gcc add.c -o add spin.o
 *
 * gse. 2006
 */

#include <stdio.h>
#include "spin.h"

char **_argv;

FILE* open_file(char *file_name) {
  FILE *file_descriptor = fopen(file_name,"rb");
  if(!file_descriptor) {
    fprintf(stderr,"%s: unable to open %s\n",_argv[0],file_name);
    exit(1);
  }
  return file_descriptor;
}

main(int argc, char *argv[]) {
  if(argc<3) {
    fprintf(stderr,"%s signal_1.float signal_2.float > signal_1+2.float\n",
    argv[0]);
  } else {
    _argv = argv;
    FILE *s1,*s2;
    s1 = open_file(argv[1]);
    s2 = open_file(argv[2]);

    int samples = 0;
    for(;;) {
      float sample1, sample2, add;
      fread(&sample1,1,sizeof(float),s1);
      if(feof(s1)) break;
      fread(&sample2,1,sizeof(float),s2);
```

```
      if(feof(s2)) break;
      add = sample1 + sample2;
      fwrite(&add,1,sizeof(float),stdout);
      samples++;
      spin();
    }
    fprintf(stderr,"%s: number of samples = %d\n",argv[0],samples);
  }
  return 0;
}
```

H.2. ascii2float.c

```
/*
 * ascii2float.c -- Genera un fichero de reales float a partir
 *                  de otro en formato ASCII.
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/ascii2float.c
 *
 * Compilar escribiendo:
 * gcc ascii2float.c -o ascii2float spin.o
 *
 * gse. 2007
 */

#include <stdio.h>
#include "spin.h"

int main(int argc, char *argv[]) {
  if(argc>1) {
    fprintf(stderr,"%s < signal.txt > signal.float\n",argv[0]);
    return 1;
  }
  for(;;) {
    float x;
    scanf("%f",&x);
    if(feof(stdin)) break;
    fwrite(&x,sizeof(float),1,stdout);
    spin();
  }
  return 0;
}
```

H.3. demodulator.c

```
/*
 * demodulator.c -- Desmodula una señal (desplaza su espectro).
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/demodulator.c
 *
 * Compilar escribiendo (el paquete fftw debería estar instalado!):
 * gcc demodulator.c -o demodulator spin.o -lfftw3 -lm
 *
 * Más información en: http/www.fftw.org
 *
```

```
 * gse. 2007
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fftw3.h>
#include "spin.h"

main(int argc, char *argv[]) {
  if(argc < 2) {
    fprintf(stderr,
            "%s carrier_frequency modulated_signal.float > unmodulated_signal.float\n"
            ,argv[0]);
  } else {

    FILE *input_file = fopen(argv[2],"rb");
    if(!input_file) {
      fprintf(stderr,
              "%s: unable to open input file \"%s\"\n",
              argv[0],argv[2]);
      exit(1);
    }

    int samples = compute_number_of_samples(input_file);
    fprintf(stderr,"%s: number of samples = %d\n",argv[0],samples);

    double       *signal   = (double       *)fftw_malloc(samples*sizeof(double      ));
    fftw_complex *spectrum = (fftw_complex *)fftw_malloc(samples*sizeof(fftw_complex));
    fftw_complex *tmp      = (fftw_complex *)fftw_malloc(samples*sizeof(fftw_complex));

    /* Creamos un plan para la transformada directa en inversa */
    fftw_plan f = fftw_plan_dft_r2c_1d(samples, signal, spectrum, 0);
    fftw_plan b = fftw_plan_dft_c2r_1d(samples, spectrum, signal, 0);

    read_signal(signal, input_file, samples);

    /* Calculamos la transformada de Fourier */
    fftw_execute(f);

    restore_spectrum(spectrum, tmp, samples, atoi(argv[1]));

    /* Calculamos la transformada de Fourier inversa */
    fftw_execute(b);

    write_signal(signal, stdout, samples);

    /* Destruímos los planes */
    fftw_destroy_plan(f); fftw_destroy_plan(b);

    /* Liberamos memoria */
    free(tmp);
    free(spectrum);
    free(signal);
  }
  return 0;
}

int compute_number_of_samples(FILE *input_file) {
```

```
  fseek(input_file,0,SEEK_END);
  int samples = ftell(input_file)/sizeof(float);
  rewind(input_file);
  return samples;
}

read_signal(double *signal, FILE *input_file, int samples) {
  int i;
  for(i=0; i<samples; i++) {
    float x;
    fread(&x,sizeof(float),1,input_file);
    signal[i] = (double)x;
    spin();
  }
}

write_signal(double *signal, FILE *output_file, int samples) {
  int i;
  for(i=0; i<samples; i++) {
    float sample = signal[i]/samples;
    fwrite(&sample,sizeof(float),1,output_file);
    spin();
  }
}

restore_spectrum(fftw_complex *spectrum, fftw_complex *tmp, int samples, int carrier) {
  int i;
  for(i=0; i<samples/2; i++) {
    tmp[i][0] = 0;
    tmp[i][1] = 0;
    spin();
  }
  for(i=0; i<samples/2 - carrier; i++) {
    tmp[i][0] = spectrum[carrier+i][0];
    tmp[i][1] = spectrum[carrier+i][1];
    spin();
  }
  for(i=0; i<samples/2; i++) {
    spectrum[i][0] = tmp[i][0];
    spectrum[i][1] = tmp[i][1];
    spin();
  }
}
```

H.4. draw_signal.sh

```
#!/bin/bash

#
# Visualiza una señal en formato ASCII almacenada en un fichero
# gse. 2006
#

function end {
#    (echo -e "" >&2)
#    (echo -n "ESC[1;0m" >&2) # ESC debe ser sustituido por el ASCII 27
    exit
}
```

```
function help {
#    (echo -en "ESC[0;33m" >&2) # ESC debe ser sustituido por el ASCII 27
    (echo -e "draw_signal signal.txt [title]" >&2)
    (echo -e "" >&2)
    (echo -e "  signal = ASCII data file" >&2)
    (echo -e "  title = title of the graph" >&2)
    (echo -e "" >&2)
    (echo -e "  Example:" >&2)
    (echo -e "" >&2)
    (echo -e "  draw_signal signal.txt" >&2)
    (echo -e "  draw_signal signal.txt \"Señal ...\"" >&2)
    end
}

if [ -z "$*" ]; then
    help
fi

signal=$1
title=$2

if [ -z $signal ]; then
  (echo -e "Sorry, missing the file name with the signal ..." >&2)
  end
fi

echo "set title \"$title\";\
 plot \"$signal\" title \"\" with linespoints pt 4 ps 0.5,\
 \"$signal\" title \"\" with lines lt 3" | gnuplot -persist
```

H.5. float2ascii.c

```
/*
 * float2ascii.c -- Muestra un fichero de reales.
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/float2ascii.c
 *
 * Compilar escribiendo:
 * gcc float2ascii.c -o float2ascii spin.o
 *
 * gse. 2007
 */

#include <stdio.h>
#include "spin.h"

int main(int argc, char *argv[]) {
  if(argc>1) {
    fprintf(stderr,"%s < signal.float > signal.txt\n",argv[0]);
    return 1;
  }

  for(;;) {
    float x;
    fread(&x,sizeof(float),1,stdin);
    if(feof(stdin)) break;
    printf("%f\n",x);
```

```
    spin();
  }

  return 0;
}
```

H.6. low_pass_filter.c

```
/*
 * low_pass_filter.c -- Filtro paso bajo.
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/low_pass_filter.c
 *
 * Compilar escribiendo (el paquete fftw debería estar instalado!):
 * gcc low_pass_filter.c -o low_pass_filter -lfftw3 -lm
 *
 * Más información en: http/www.fftw.org
 *
 * gse. 2006
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fftw3.h>
#include "spin.h"

main(int argc, char *argv[]) {
  if(argc < 2) {
    fprintf(stderr,
            "%s cut-off_band signal.float > filtered_signal.float\n"
            ,argv[0]);
  } else {

    FILE *input_file = fopen(argv[2],"rb");
    if(!input_file) {
      fprintf(stderr,
              "%s: unable to open input file \"%s\"\n",
              argv[0],argv[2]);
      exit(1);
    }

    int samples = compute_number_of_samples(input_file);
    fprintf(stderr,"%s: number of samples = %d\n",argv[0],samples);

    double       *signal   = (double       *)fftw_malloc(samples*sizeof(double      ));
    fftw_complex *spectrum = (fftw_complex *)fftw_malloc(samples*sizeof(fftw_complex));

    /* Creamos un plan para la transformada directa en inversa */
    fftw_plan f = fftw_plan_dft_r2c_1d(samples, signal, spectrum, 0);
    fftw_plan b = fftw_plan_dft_c2r_1d(samples, spectrum, signal, 0);

    read_signal(signal, input_file, samples);

    /* Calculamos la transformada de Fourier */
    fftw_execute(f);

    filter_signal(spectrum, atof(argv[1]), samples, argv);
```

```
    /* Calculamos la transformada de Fourier inversa */
    fftw_execute(b);

    write_signal(signal, stdout, samples);

    /* Destruímos los planes */
    fftw_destroy_plan(f); fftw_destroy_plan(b);

    /* Liberamos memoria */
    free(spectrum);
    free(signal);
  }
  return 0;
}

int compute_number_of_samples(FILE *input_file) {
  fseek(input_file,0,SEEK_END);
  int samples = ftell(input_file)/sizeof(float);
  rewind(input_file);
  return samples;
}

read_signal(double *signal, FILE *input_file, int samples) {
  int i;
  for(i=0; i<samples; i++) {
    float x;
    fread(&x,sizeof(float),1,input_file);
    signal[i] = (double)x;
    spin();
  }
}

write_signal(double *signal, FILE *output_file, int samples) {
  int i;
  for(i=0; i<samples; i++) {
    float sample = signal[i]/samples;
    fwrite(&sample,sizeof(float),1,output_file);
    spin();
  }
}

filter_signal(fftw_complex *out, double factor, int samples, char **argv) {
  int coefs_erased = 0; /* Coeficientes borrados */
  int i;
  int fmax = samples/2;
  int bandas_eliminadas = fmax*(1.0-factor);
  for(i=fmax-bandas_eliminadas; i<fmax; i++) {
    out[i][0] = out[i][1] = 0.0;
    coefs_erased++;
    spin();
  }
  fprintf(stderr,"%s: number of erased Fourier coefficients = %d\n",
  argv[0],coefs_erased);
  fprintf(stderr,"%s: number of retained Fourier coeffcients = %d\n",
  argv[0],samples/2-coefs_erased);
}
```

cut-off_band: es el índice, expresado como un número real entre 0 y 1, de la banda de frecuencia más baja que es retenida (no eliminada). En general este parámetro se calcula a partir de una frecuencia de

corte (cut-off) f_c medida en Hz. Sea f_s la frecuencia de muestreo (número de muestras por segundo). Según el Teorema del Muestreo Uniforme, la máxima componente de frecuencia de dicha señal no es superior a $f_s/2$. Entonces se cumple que:

$$\texttt{cut-off\_band} = \frac{f_c}{f_s/2}$$

Supongamos, por ejemplo, que la señal digital transporta un bit de información por segundo (esto significa que $f_0 = 1$ Hz en la Figura 2.1). Si hemos tomado 32 muestras/bit, entonces tendremos $f_s = 32$ muestras/segundo y que estamos registrando hasta 16 Hz del espectro de dicha señal. Si queremos eliminar, por ejemplo, la banda de frecuencias que va desde 8 Hz a 16 Hz, es decir, aplicar un filtrado paso bajo:

$$\texttt{cut-off\_band} = \frac{8}{16} = 0{,}5$$

`señal.float`: es un fichero que almacena la señal a filtrar. Puesto que estamos realizando los cálculos en una máquina digital, las señales deben estar digitalizadas (muestreadas usando una frecuncia f_s de muestreo constante y cuantificadas utilizando un determinado número de bits).

`filtered_signal.float`: fichero que almacena la señal filtrada. La frecuencia de muestreo y el número de bits por muestra son conservados. Por tanto, los ficheros `señal_original` y `señal_filtrada` deben tener el mismo tamaño.

H.7. modulator.c

```
/*
 * modulator.c -- Modula una señal (desplaza su espectro).
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/modulator.c
 *
 * Compilar escribiendo (el paquete fftw debería estar instalado!):
 * gcc modulator.c -o modulator spin.o -lfftw3 -lm
 *
 * Más información en: http/www.fftw.org
 *
 * gse. 2007
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fftw3.h>
#include "spin.h"

main(int argc, char *argv[]) {
  if(argc < 2) {
    fprintf(stderr,
            "%s carrier_frequency unmodulated_signal.float > modulated_signal.float\n"
            ,argv[0]);
  } else {

    FILE *input_file = fopen(argv[2],"rb");
    if(!input_file) {
      fprintf(stderr,
              "%s: unable to open input file \"%s\"\n",
              argv[0],argv[2]);
      exit(1);
    }
```

```
    int samples = compute_number_of_samples(input_file);
    fprintf(stderr,"%s: number of samples = %d\n",argv[0],samples);

    double       *signal   = (double       *)fftw_malloc(samples*sizeof(double      ));
    fftw_complex *spectrum = (fftw_complex *)fftw_malloc(samples*sizeof(fftw_complex));
    fftw_complex *tmp      = (fftw_complex *)fftw_malloc(samples*sizeof(fftw_complex));

    /* Creamos un plan para la transformada directa en inversa */
    fftw_plan f = fftw_plan_dft_r2c_1d(samples, signal, spectrum, 0);
    fftw_plan b = fftw_plan_dft_c2r_1d(samples, spectrum, signal, 0);

    read_signal(signal, input_file, samples);

    /* Calculamos la transformada de Fourier */
    fftw_execute(f);

    modulate_spectrum(spectrum, tmp, samples, atoi(argv[1]));

    /* Calculamos la transformada de Fourier inversa */
    fftw_execute(b);

    write_signal(signal, stdout, samples);

    /* Destruímos los planes */
    fftw_destroy_plan(f); fftw_destroy_plan(b);

    /* Liberamos memoria */
    free(tmp);
    free(spectrum);
    free(signal);
  }
  return 0;
}

int compute_number_of_samples(FILE *input_file) {
  fseek(input_file,0,SEEK_END);
  int samples = ftell(input_file)/sizeof(float);
  rewind(input_file);
  return samples;
}

read_signal(double *signal, FILE *input_file, int samples) {
  int i;
  for(i=0; i<samples; i++) {
    float x;
    fread(&x,sizeof(float),1,input_file);
    signal[i] = (double)x;
    spin();
  }
}

write_signal(double *signal, FILE *output_file, int samples) {
  int i;
  for(i=0; i<samples; i++) {
    float sample = signal[i]/samples;
    fwrite(&sample,sizeof(float),1,output_file);
    spin();
  }
}
```

```
modulate_spectrum(fftw_complex *spectrum, fftw_complex *tmp, int samples, int carrier) {
  int i;
  for(i=0; i<samples/2; i++) {
    tmp[i][0] = 0;
    tmp[i][1] = 0;
    spin();
  }
  for(i=0; i<samples/2 - carrier; i++) {
    tmp[carrier+i][0] = spectrum[i][0];
    tmp[carrier+i][1] = spectrum[i][1];
    spin();
  }
  for(i=0; i<samples/2; i++) {
    spectrum[i][0] = tmp[i][0];
    spectrum[i][1] = tmp[i][1];
    spin();
  }
}
```

`carrier_band`: Indica la banda de frecuencia en la que está la portadora (recuérdese que una portadora es una señal sinusoidal pura y por tanto ocupa, en un espectro discreto, sólo una banda de frecuencia). Para calcular este valor tenemos que conocer qué ancho de banda x representa cada coeficiente de Fourier y luego dividir la frecuencia de la portadora w_f entre x, es decir:

$$\texttt{carrier\_band} = \frac{x}{w_f}$$

`unmodulated_signal.float`: El fichero de la señal a modular.

`modulated_signal.float`: El fichero de la señal modulada.

H.8. sampler.c

```
/*
 * sampler.c -- Muestrea una señal digital.
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/sampler.c
 *
 * Compilar escribiendo:
 * gcc sampler.c -o sampler spin.o
 *
 * gse. 2007
 */

#include <stdio.h>
#include <stdlib.h> /* atoi() */
#include "spin.h"

int main(int argc, char *argv[]) {
  if(argc<2) {
    fprintf(stderr,"%s replication_factor < signal.float > signal.float\n",
            argv[0]);
    return 1;
  }
  {
    int output_samples_per_input_sample = atoi(argv[1]);
    for(;;) {
      int i;
```

```
      float input_sample;
      fread(&input_sample,sizeof(float),1,stdin);
      if(feof(stdin)) break;
      for(i=0; i<output_samples_per_input_sample; i++) {
        fwrite(&input_sample,sizeof(float),1,stdout);
      }
      spin();
    }
    return 0;
  }
}
```

H.9. spectrum_analyzer.c

```
/*
 * spectrum_analyzer.c -- Analizador de espectros.
 *
 * Este fichero fuente puede encontrarse en:
 * http://www.ace.ual.es/~vruiz/docencia/redes/practicas/spectrum_analyzer.c
 *
 * Compilar escribiendo (el paquete fftw debería estar instalado!):
 * gcc spectrum_analyzer.c -o spectrum_analyzer spin.o -lfftw3 -lm
 *
 * Más información en: http/www.fftw.org
 *
 * gse. 2007
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fftw3.h>
#include "spin.h"

main(int argc, char *argv[]) {
  if(argc < 2) {
    fprintf(stderr,"%s input_signal.float > spectrum.txt\n",argv[0]);
  } else {
    FILE *input_file = fopen(argv[1],"rb");
    if(!input_file) {
      fprintf(stderr,"%s: unable to open input file \"%s\"\n"
              ,argv[0],argv[2]);
      exit(1);
    }

    int samples = compute_number_of_samples(input_file);
    fprintf(stderr,"%s: number of samples = %d\n",argv[0],samples);

    double       *in  = (double       *)fftw_malloc(samples*sizeof(double      ));
    fftw_complex *out = (fftw_complex *)fftw_malloc(samples*sizeof(fftw_complex));

    /* Creamos un plan para la fftw */
    fftw_plan p = fftw_plan_dft_r2c_1d(samples, in, out, 0);

    read_signal(in, input_file, samples);

    /* Calculamos la transformada de Fourier */
    fftw_execute(p);
```

```
    /* Generamos el espectro de la señal */ {
      int k;
      for(k=0; k<samples/2; k++) {
double power_spectrum = sqrt(out[k][0]*out[k][0] + out[k][1]*out[k][1]);
fprintf(stdout,"%f\n",power_spectrum);
      }
    }

    /* Destruímos el plan */
    fftw_destroy_plan(p);

    /* Liberamos memoria */
    free(out);
    free(in);
  }
  return 0;
}

int compute_number_of_samples(FILE *input_file) {
  fseek(input_file,0,SEEK_END);
  int samples = ftell(input_file)/sizeof(float);
  rewind(input_file);
  return samples;
}

read_signal(double *in, FILE *input_file, int samples) {
  int i;
  for(i=0; i<samples; i++) {
    float x;
    fread(&x,sizeof(float),1,input_file);
    in[i] = (double)x;
    spin();
  }
}
```

Apéndice I

La tabla ASCII imprimible de 7 bits

Dec	Hex	Char
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	’
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
:	:	:
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
:	:	:
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	‘
97	61	a

98	62	b
99	63	c
:	:	:
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	\~{}

Bibliografía

[1] Oskar Andreasson. Iptables tutorial. http://iptables-tutorial.frozentux.net/iptables-tutorial.htm.

[2] Renaud Deraison. *Nessus Vulnerability Scanner*. http://www.nessus.org/about/.

[3] Joshua Drake. *Linux Networking HOWTO*. LDP (Linux Documentation Project), 2000.

[4] R. Droms. *Dynamic Host Configuration Protocol*. http://www.ietf.org/rfc/rfc2131.txt, March 1977.

[5] Behrouz Forouzan. *TCP/IP Protocol Suite*. McGraw-Hill, 2000.

[6] IANA (Internet Assigned Numbers Authority), http://www.iana.org/assignments/port-numbers. *The well known ports number*.

[7] Insecure.Org, http://insecure.org/nmap/. *Nmap*.

[8] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet (3rd Edition)*. Addison Wesley, 2005.

[9] Bhagwandas Pannalal Lathi. *Introducción a la Teoría y Sistemas de Comunicación*. Limusa Noriega Editores, 1994.

[10] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-time signal processing*. Prentice Hall, 1999.

[11] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Señales y Sistemas (2a edición)*. Prentice Hall, 1997.

[12] Rusty Russell. *Linux 2.4 Packet Filtering HOWTO*. http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html.

[13] Claude E. Shannon. *The Mathematical Theory of Communication*. University of Illinois Press, 1963.

[14] SSH Communications Security Corp, http://www.rfc-editor.org/rfc/rfc4251.txt. *RFC 4251. The Secure Shell (SSH) Protocol Architecture*, 2006.

[15] SSH Communications Security Corp, http://www.rfc-editor.org/rfc/rfc4252.txt. *RFC 4252. The Secure Shell (SSH) User Authentication Protocol*, 2006.

[16] SSH Communications Security Corp, http://www.rfc-editor.org/rfc/rfc4253.txt. *RFC 4253. The Secure Shell (SSH) Transport Layer Protocol*, 2006.

[17] SSH Communications Security Corp, http://www.rfc-editor.org/rfc/rfc4254.txt. *RFC 4254. The Secure Shell (SSH) Connection Protocol*, 2006.

[18] William Stallings. *Comunicaciones y Redes de Computadores (7a Edición)*. Prentice Hall, 2004.