Redes1 - Lab

Jesús Blanco, Cleto Martín, José Luis Segura

25 de junio de 2025

Índice general

This documentation is also available in English

En esta documentación podrás encontrar el material necesario para relizar las prácticas de la asignatura Redes de Computadores 1 para el curso 2024/2025. Las *prácticas* se irán publicando conforme avancen las sesiones de laboratorio de la asignatura.

Esta documentación también está disponible en PDF.

Profesores

Jesús Blanco A1 - Miércoles 18:30h

A2 - Jueves 18:30h

D2 - Jueves 17:00h

Jesús Barba B1 - Martes 10:00h

Inocente Sánchez B2 - Viernes 10:00h

D1 - Lunes 11:30h

Cristina Bolaños C1 - Jueves 8:30h

C2 - Lunes 13:00h

Dinámica general

Las sesiones de prácticas están diseñadas para reforzar y ampliar los conocimientos adquiridos en las clases de teoría. Muchos de los materiales y cuestiones que se verán durante las sesiones requerirán de práctica y estudio posterior.

Las primeras sesiones de prácticas irán dirigidas a la familiarización con el entorno y que *no contarán* para la nota. Más adelante, nos centraremos en las cuestiones propias de las redes de computadores, las cuales *sí contarán* para la nota de laboratorio y que requerirán de la resolución de ejercicios (durante la sesión de prácticas).

El entorno de prácticas estará basado en el sistema operativo GNU/Linux. Para saber más al respecto, échale un vistazo a la sección de nuestra *máquina virtual*.

CAPÍTULO 1

Máquina Virtual

Las prácticas de laboratorio se van a realizar sobre un sistema operativo llamado Debian que es una de las más famosas y antiguas distribuciones de GNU/Linux y que se utiliza en gran cantidad de entornos y aplicaciones, mayoritariamente en infraestructuras de red.

> **Nota**
>
> Es posible utilizar cualquier otra distribución de GNU/Linux como Ubuntu o Fedora. Te *recomendamos* utilizar la máquina virtual que te proporcionamos de forma que puedas seguir los ejemplos en la misma línea que los explicamos.

Para que sea más sencilla su distribución e instalación, hemos creado una máquina virtual para VirtualBox de forma que sea posible ejecutar una máquina completa con nuestra Debian configurada para las prácticas sin necesidad de instalarlo nativamente.

Esta imagen está disponible en los *ordenadores de los laboratorios*, por lo que los utilizaremos para para las sesiones de prácticas. También puedes instalarlo en *tu propio PC*, por lo que puedes tener el mismo entorno para practicar en casa o utilizarlo en el laboratorio si tu ordenador asignado no funciona por algún motivo. Si ya tienes alguna distribución de GNU/Linux instalada en tu PC, es muy probable que puedas utilizarla directamente.

1.1 Ordenadores de los laboratorios

Como norma general, los ordenadores del laboratorio son los que se deben utilizar durante las sesiones de prácticas.

Para arrancar la máquina virtual, haz lo siguiente:

- Enciende el ordenador y espera a que arranque el sistema operativo.
- Una vez arrancado, aparecerá un menú con diferentes máquina virtuales para arrancar.
- Selecciona, **en este orden**:
  1. `Vol O`
  2. `Red de Laboratorio`

3. `Ejecutar`

- Espera unos segundos y la máquina virtual comenzará a arrancar.
- Una vez arrancada, pedirá el nombre de usuario y contraseña.

> **Nota**
>
> Usuario: `alumno`
>
> Contraseña: `alumno`

1.2 Instalación en tu PC

Con estas instrucciones podrás arrancar la máquina virtual de prácticas en tu PC sin necesidad de hacer una instalación nativa. Están dirigidas principalmente a usuarios con Windows o MacOS que no disponen de una distribución de GNU/Linux instalada de alguna forma en su PC.

1.2.1 Requisitos

Para poder usar la máquina virtual de prácticas tendrás que tener:

- Tener unos **10GB** de espacio libre en disco.
- Tener VirtualBox instalado.
- Tener el soporte de **virtualización activado**. Es posible que tu PC ya lo tenga pero si te falla la carga de la máquina virtual, es muy probable que tengas que activarlo.

  Se activa normalmente en la BIOS de tu PC y el modo de hacerlo depende en gran parte del fabricante del mismo. Te recomendamos que busques el modelo de tu PC y cómo activar el soporte de virtualización para activarlo (o asegurarte que ya lo tienes activo).

Opcionalmente, y si quieres utilizar tu PC durante las sesiones de laboratorio, tu PC necesitará tener una interfaz de red *cableada*. En caso de que tu PC sólo tenga interfaz inalámbrica, puedes comprar una interfaz de red con conexión USB.

1.2.2 Procedimiento

1. Descargar el archivo OVA que contiene la máquina virtual.
2. Abrir VirtualBox.
3. Pulsar en `Archivo` -> `Importar`
4. Seleccionar el archivo OVA descargado anteriormente en el campo de `Archivo` y pulsar `Siguiente`.
5. En la siguiente pantalla, pulsar `Importar`.
6. Tras unos segundos, aparecerá en el panel izquierdo la nueva máquina virtual llamada `redes1`.

> **Importante**
>
> Antes de lanzar por primera vez la máquina virtual, se debe configurar la interfaz de red correctamente:
>
> 1. Selecciona la máquina virtual en el panel izquierdo y pulsa `Configurar`.

2. En el apartado de `Red`, en la pestaña de `Adaptador 1`, cambia el valor `NAT` por `Puente`. Y en la opciones avanzadas cambia la dirección MAC por una generada aleatoria pulsando el botón de refrescar.
3. Finaliza pulsando `Aceptar`.

Este procedimiento solo hay que hacerlo antes de ejecutar por primera vez la máquina virtual.

La máquina ya estaría lista para ser utilizada. Para arrancarla, sólo tienes que hacer doble click sobre ella en el panel izquierdo y comenzará su ejecución.

1.3 Configuración del lenguaje

La máquina virtual está configurada por defecto en *español*. Pero se puede cambiar el idioma a *inglés* ejecutando el siguiente comando en una terminal:

```
$ sudo localectl set-locale en_GB.UTF-8
```

Tambien, se puede configurar la distribución del teclado en la máquina virtual para adecuarla a tu teclado real. La distribución de teclado en la máquina vitual está configurada en español, por lo que mantener si se trabaja con un teclado español. Se puede cambiar la máquina virtual para que utilice una distribución de teclado inglés con los siguientes comandos:

```
$ sudo localectl set-keymap en
$ sudo localectl set-x11-keymap en
```

Se pueden emplear otras distribuciones de teclado. Simplemente usa la que desees en lugar de `en`.

Finalmente, reinicia la máquina virtual.

CAPÍTULO 2

Prácticas

Las prácticas se irán publicando conforme avancen las sesiones de laboratorio de la asignatura. Aquí tienes un listado de las actualmente disponibles:

2.1 Práctica 1 - Línea de Comandos I

El terminal es el software del sistema operativo que permite introducir datos en el ordenador y obtener la salida de los diferentes programas.

En toda instalación de GNU/Linux (y en general de cualquier sistema operativo) existe una aplicación de terminal o un emulador gráfico de la misma.

En nuestro sistema operativo de referencia, podemos utilizar `LXTerminal`, disponible en el menú de *Aplicaciones* -> *Herramientas del sistema*.

2.1.1 ¿Qué es un comando?

Un comando es un programa que puede ser ejecutado desde el terminal, que acepta una serie de argumentos y produce una salida específica o bien provoca un cambio determinado en el sistema.

Estructura de un comando

Un comando consta de su propio nombre, argumentos y, de manera opcional, opciones:

```
$ command [--option=value] [-f] [argument1 argument2 ...]
```

- `command` sería el nombre del comando. Más adelante veremos muchos ejemplos.
- `--option` y `-f` serían opciones. Normalmente las opciones tienen 2 versiones, la larga, que suele anteponer dos guiones, y la corta, que sólo antepone uno. Las opciones modifican el modo normal de funcionamiento de un comando.
- `argument1` y `argument2` son los argumentos. En muchas ocasiones no es necesario especificarlos porque toman valores por defecto.

Habitualmente, todos los comandos tienen la opción `--help` y `-h`, que permiten obtener la ayuda del comando, donde nos explican las diferentes opciones y argumentos que acepta dicho comando.

Leer el manual

Aunque como se ha explicado anteriormente, todos los comandos pueden proporcional ayuda sobre sus propias opciones y argumentos, pero en algunas ocasiones se necesita mucha más información, ya que algunos comandos pueden necesitar ficheros de configuración, utilizar variables de entorno, producir ficheros de salida y un largo etcétera.

Existe un comando específico para solicitar más información sobre otros comandos: `man`.

El manual del terminal permite acceder a una completa página de ayuda sobre el comando solicitado, con las diferentes opciones, argumentos o el formato de los mismos. También si el programa utiliza variables de entorno, información de cómo reportar bugs o la licencia del mismo.

La navegación por las páginas del manual puede resultar poco intuitiva al principio. Aquí algunos trucos:

- `Flecha Arriba`, `Flecha Abajo`, `Av. Pág`, `Re. Pág` permiten subir y bajar por la página.
- `q` termina la ejecución del comando `man`.
- `/` permite buscar una cadena en el documento actual. Una vez en modo búsqueda, se puede usar `n` o `N` para buscar hacia abajo o hacia arriba en el documento, respectivamente.

Por ejemplo, abre la página del manual del comando `date` y busca cómo se especifica el formato de salida del comando:

1. Ejecuta `man date`.
2. Pulsa la tecla `/`.
3. Escribe la cadena a buscar `formato`.
4. Pulsa `n` hasta encontrar la sección donde se explica el formato.

> **Nota**
>
> Estos atajos de teclado explicados para navegar por las páginas del manual también son válidos para navegar por cualquier fichero utilizando el comando `less` que se verá más adelante.

2.1.2 Familiarizándonos con el sistema de archivos

El sistema de archivos es la parte del sistema operativo que permite interactuar con los dipositivos de almacenamiento de nuestra máquina.

El primer directorio que debemos conocer en GNU/Linux es el raíz o *root*: `/`. En ese directorio suele montarse un dispositivo de almacenamiento, normalmente una partición de un disco duro. Dentro de ese directorio suelen existir los siguientes:

- `/boot`: contiene la configuración de arranque, así como las imágenes del kernel.
- `/dev`: contiene toda una estructura virtual de directorios y ficheros que representan a los diferentes dispositivos presentes en la máquina.
- `/etc`: la configuración de los servicios instalados en el sistema.
- `/home`: contiene los directorios personales de los usuarios. En la máquina virtual proporcionada puedes encontrar `/home/alumno`, que es donde trabajarás.
- `/media` y `/mnt`: rutas donde suelen montarse, según la distribución, los dispositivos de caracter temporal, como memorias USB o discos duros externos.
- `/opt`: ruta donde se instalan aplicaciones de terceros.
- `/proc`: estructura virtual de directorios y ficheros con información relativa a los programas en ejecución.
- `/root`: directorio personal del usuario `root`, que es el administrador del sistema.
- `/run`: estructura virtual de directorios y ficheros con datos temporales de procesos en ejecución.
- `/srv`: directorio donde se almacenan ficheros que serán servidos a otros sistemas (en desuso).
- `/sys`: estructura virtual con archivos utilizados por el propio sistema.
- `/tmp`: directorio temporal. Dependiendo de la configuración, puede ser volatil (se pierde tras cada reinicio) o persistente.
- `/usr`: directorio donde se instalan todos los programas del sistema operativo.
- `/var`: ruta donde se almacenan todos los ficheros variables, como las bases de datos, ficheros de log, cachés de ficheros…

> **Nota**
>
> Con «estructura virtual» se hace referencia a toda una jerarquía de ficheros y directorios que son creados en el sistema a nivel lógico, pero que no representan a directorios ni ficheros almacenados en ningún disco duro u otro almacenamiento.

Trabajando con el sistema de archivos

En muchas ocasiones necesitamos consultar el contenido de archivos, modificarlos, visitar otros directorios, ejecutar comandos en diferentes rutas… Para ello debemos conocer los comandos básicos para hacer consultas al sistema de archivos, cambiar el directorio de trabajo, crear, modificar y borrar archivos, copiarlos o moverlos a diferentes directorios.

- `pwd`: muestra el directorio de trabajo actual.
- `ls`: lista los ficheros y directorios. Si pasamos una ruta, nos mostrará el listado correspondiente a dicha ruta.
- `cat`: muestra el contenido de un archivo.

- `cd RUTA`: permite cambiar el directorio de trabajo actual. Si no pasamos una ruta, se configurará el directorio personal del usuario. Si en lugar de una ruta usamos el carácter `-`, nos permite volver al directorio de trabajo anterior.
- `mkdir DIRECTORIO`: crea un nuevo directorio.
- `rmdir DIRECTORIO`: elimina un directorio (siempre que esté vacío).
- `rm RUTA`: elimina un archivo.
  - Con `-r` permite eliminar un directorio con sus subdirectorios y archivos recursivamente.
- `touch ARCHIVO`: crea un archivo vacío o actualiza su fecha de acceso si ya existe.
- `cp ORIGEN DESTINO`: copia un archivo a otra ruta o directorio.
- `mv ORIGEN DESTINO`: mueve un archivo a otra ruta o directorio (puede usarse para renombrar ficheros).

Rutas absolutas y relativas

Cuando hablamos de rutas del sistema de archivos, podemos referirnos a ellas de manera absoluta, dando la ruta completa hasta el fichero o directorio, o de forma relativa, donde sólo damos una parte de la ruta, relativa al directorio actual.

¿Y cuál es el directorio actual? Cuando estamos en un terminal, podemos consultarlo usando el comando `pwd`. Cuando abrimos un terminal, el directorio de trabajo inicialmente es el directorio personal (o «home») del usuario (`/home/alumno`, por ejemplo).

Siempre que se especifique una ruta cuyo primer carácter es una `/` se trata de una ruta absoluta. Cuando empieza por cualquier otro carácter, se tratará de una ruta relativa al directorio actual.

También hay algunos especificadores de directorio especiales:

- `~`: hace referencia el directorio personal del usuario.
- `.`: hace referencia al directorio actual de trabajo.
- `..`: hace referencia al directorio «anterior» o «padre» del actual.

Por ejemplo, si abrimos el terminal, nuestro directorio de trabajo será `/home/alumno`. La ruta absoluta `/var/log` y las relativas `../../var/log` o `~/../../var/log` son equivalentes y hacen referencia al mismo directorio.

> **Nota**
>
> Presionando el `Tab`, la terminal completará automáticamente el nombre de tu comando o ruta. En caso de que haya más de una opción, presionando dos veces el `Tab` se muestran todas las opciones posibles.

Permisos

En todo sistema operativo, los ficheros del sistema de archivos tienen asignados una serie de permisos que le indican al sistema qué usuarios pueden realizar las diferentes operaciones sobre cada fichero o directorio.

En GNU/Linux, los permisos se dividen en 3: lectura (**r**ead), escritura (**w**rite) y ejecución (e**x**ecution). A su vez, cada permiso se puede definir de forma diferente para 3 «roles» diferentes de usuario: el usuario propietario del archivo, usuarios miembros del grupo propietario del archivo y el resto de usuarios (no son el propietario ni miembros del grupo propietario).

Para consultar los permisos de un fichero, puedes usar el comando `ls -l` (consulta en el manual el significado completo de su salida).

Ejecuta en tu terminal `ls -l /etc/hosts`:

```
$ ls -l /etc/hosts
-rw-r--r--. 1 root root 185 Dec 16 17:00 /etc/hosts
```

En este caso, el primer carácter nos indica que no se trata de ningún tipo especial de fichero; los 3 caracteres siguientes (`rw-`) nos indican los permisos asignados al usuario propietario del archivo; los 3 siguientes (`r--`), los permisos del grupo propietario y, los 3 siguientes (también `r--`) los permisos para otros usuarios. Tras eso, podemos ver también el nombre del usuario y el grupo propietario (usuario `root` y grupo `root`).

Resumiendo, este fichero únicamente tiene permisos de lectura y escritura para el usuario `root` y de lectura para todos los demás usuarios, tanto si son del grupo `root` como si no.

La manera de modificar los permisos de un fichero es mediante el comando `chmod`:

```
$ chmod MODE PATH
```

Este comando nos obliga a pasar, primero, el modo que queremos configurar para el fichero, y después, su ruta.

El modo se puede definir a través de caracteres o a través de su versión octal, que no es si no una representación numérica de los 3 grupos de permisos que queremos configurar para el fichero.

En la siguiente imagen tienes una explicación de cómo se traducen los permisos entre el modo textual y octal:

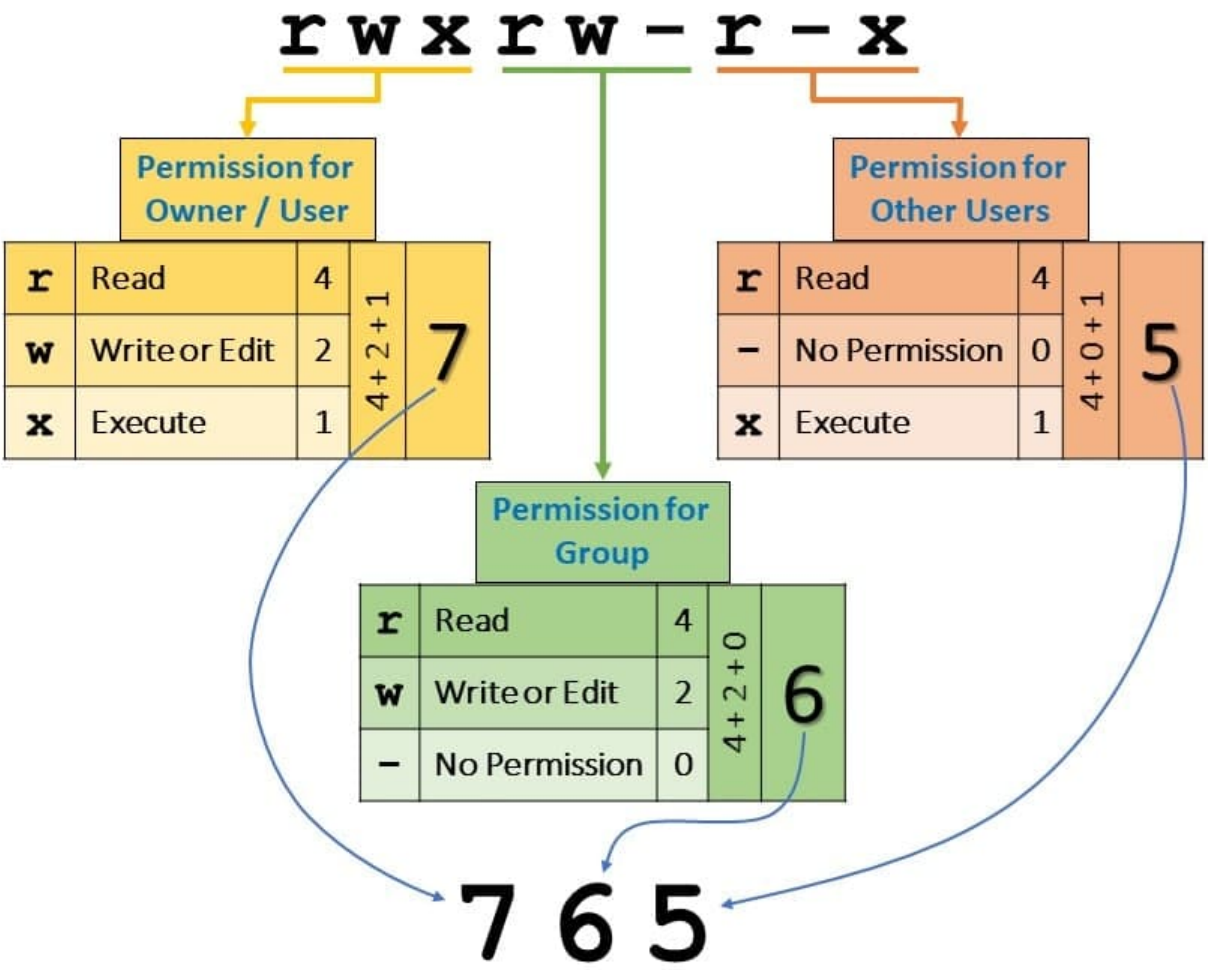

Figura 2.1: Conversión de permisos

2.1.3 Ejercicio

Para asentar lo visto durante esta sesión, vamos a realizar una serie de ejercicios prácticos usando un terminal, la línea de comandos y algunos de los comandos vistos anteriormente:

1. Comprueba si en el directorio `/` de tu máquina hay algún fichero o directorio adicional a los mencionados en esta sesión.

   #### Solución

   Aparecen los siguientes elementos:

   - `bin`
   - `initrd.img` e `initrd.img.old`
   - `lib`, `lib32` y `lib64`
   - `lost+found`
   - `sbin`
   - `vmlinuz` y `vmlinux.old`

   La mayoría son enlaces simbólicos que se mantienen por compatiblilidad con versiones más antiguas. `initrd.img` y `vmlinuz` son enlaces a las imágenes de arranque y del kernel. `lost+found` es el directorio donde se recuperan archivos corruptos cuando se hacen comprobaciones en cualquier sistema de ficheros de tipo "ext".

2. Busca en la ayuda o el manual del comando `ls` cómo mostrar los detalles de cada archivo y comprueba el tamaño del fichero `/etc/fstab`.

   #### Solución

   Usar `man ls` o `ls --help` para encontrar que la opción adecuada es `ls -l` y ejecutar `ls -l /etc/fstab`

   ```
   $ ls -l /etc/fstab
   -rw-r--r-- 1 root root 826 feb  8 11:45 /etc/fstab
   ```

3. Crea un directorio `licenses` dentro de tu directorio de usuario.

   #### Solución

   ```
   mkdir licenses
   ```

4. Busca en la ayuda o el manual del comando `cp` como hacer una copia de un directorio de forma recursiva y copia el directorio `/usr/share/common-licenses` al directorio creado en el paso anterior. ¿Has obtenido el resultado esperado?

Solución

Buscar en el manual de `cp` o en `cp --help` el modo recursivo y usarlo con `cp -r`:

```
$ cp -r /usr/share/common-licenses licenses
```

Al hacerlo así, se creará un directorio `licenses/common-licenses`. Razonar por qué ocurre esto, cuál era el resultado esperado y cómo conseguirlo.

5. Usando `ls` con las opciones necesarias, compara los permisos y propietarios de los directorios `/usr/share/common-licenses` y el recién creado `licenses/common-licenses` en tu directorio de usuario. ¿Qué diferencias encuentras? ¿A qué crees que se deben?

Solución

Ejecutar `ls -l` en los 2 directorios indicados y comprobar que la principal diferencia es el propietario de los archivos y las fechas.

6. Cambia el directorio de trabajo a `licenses` y, desde ahí, crea un directorio `GPLs` en tu directorio de usuario.

Solución

`cd licenses` y `mkdir ~/GPLs` o `mkdir ../GPLs`

7. Copia los 4 ficheros que empiezan por `GPL` del directorio `licenses/common-licenses` al directorio `GPLs`.

Solución

```
cp common-licenses/GPL* ../GPLs/
```

8. Cambia al directorio `GPLs` y modifica los permisos para que sólo pueda leer o modificar los ficheros el usuario (es decir, quitar permisos de lectura y escritura al resto).

Solución

`cd ~/GPLs` y `chmod 600 *` o `chmod go-rwx *`

9. Borra el directorio `licenses` con todos sus subdirectorios y ficheros.

Solución

`cd` y `rm -fr licenses`. Intentarlo sin hacer antes el `cd`, para que se vea que se elimina y que `pwd` lo sigue mostrando como directorio de trabajo, pero que no podemos hacer nada dentro.

10. Borra el directorio `GPLs` con todos sus subdirectorios y ficheros para dejar limpio el directorio de usuario.

Solución

`cd` y `rm -fr GPLs`

> **Nota**
>
> Para limpiar el contenido de la terminal emplear el comando `clear` o pulsar la combinación de teclas `Ctrl l`.

2.2 Práctica 2 - Línea de Comandos II

Continuamos con el trabajo que hemos hecho en la *práctica 1*, profundizando un poco más en el manejo de la línea de órdenes. En esta ocasión veremos cómo agrupar comandos para crear *scripts* y así automatizar tareas. También cómo utilizar unas técnicas muy potentes para agrupar funcionalidad entre comandos como son las redirecciones y las tuberías. Finalmente, veremos como instalar nuevos programas utilizando el gestor de paquetes de Debian.

2.2.1 Scripts

Una de las características más importantes de la línea de comandos es que todo lo que se hace sobre ella se puede almacenar en un archivo y ejecutarlo tantas veces como se quiera. La propia línea de órdenes acepta como entrada todo un lenguaje de programación (con sus sentencias *if*, bucles *for* etc.), lo que permite automatizar tareas a todos los niveles. A estos programas que contienen un conjunto de órdenes que se ejecuta de una vez se les conoce como *scripts*.

Vamos a ver cómo crear *scripts* muy sencillos desde el terminal.

Editando archivos con `nano`

Para crear un archivo nuevo, simplemente tienes que pasar la ruta dónde quieras almacenarlo. Por ejemplo:

```
$ nano my-file.txt
```

Puedes escribir texto y, cuando quieras, puedes salvarlo a disco utilizando `Ctrl o`. Pedirá confirmación sobre el nombre de archivo, que será el mismo que has dado, y aceptas con `Enter`. Con ello, el archivo quedará almacenado.

Para salir del editor, sólo tienes que usar `Ctrl x`. Si no guardaste, te pedirá si quieres salir sin guardar o guardar los cambios actuales.

> **Nota**
>
> Con `Ctrl g` puedes ver todos los atajos de teclado que permite `nano`. En particular, son interesantes `Ctrl k` y `Ctrl u` que permiten cortar y pegar una línea, respectivamente.

Editando archivos con `gedit`

También hay editores de texto plano gráficos como `gedit`. Puedes lanzarlo desde el terminal simplemente utilizando el comando `gedit`.

> **Advertencia**
>
> Aunque los editores gráficos son más fáciles de usar, te recomendamos que te habitúes a utilizar editores como `nano` ya que son editores que están presentes en muchos hardware de red como routers de los cuales no dispondrás de interfaz gráfica. Además, suele ser mucho más rápido editar pequeños cambios con este tipo de herramientas que con las alternativas gráficas.

Ejercicio: crea un script

Ahora que sabes cómo crear un archivo de texto, puedes crear scripts y ejecutarlos. Por ejemplo, crea el archivo `/home/alumno/my-script.sh` con el siguiente contenido:

```
#!/bin/bash

echo "-- Deployer v1.0 --"

# First, create all directories
mkdir dir1 dir2 dir3

# Then, copy myself into these directories
cp /home/alumno/my-script.sh dir1
cp /home/alumno/my-script.sh dir2
cp /home/alumno/my-script.sh dir3

echo "-- All done! --"
```

El script es bastante sencillo de entender. Algunas cuestiones nuevas son:

- La línea `#!/bin/bash` que siempre se pone al principio se conoce como *shebang*. Esa primera línea tiene que empezar con `#!` seguida del intérprete que quieras utilizar. En este ejemplo, usamos Bash.
- El comando `echo` sirve para imprimir cadenas de caracteres por la salida estándar.
- Las líneas que empiezan con `#` se consideran comentarios, por lo que el intérprete las ignora.

Para ejecutar el nuevo script puedes hacer:

```
$ bash /home/alumno/my-script.sh
```

También es muy común darle permisos de ejecución:

```
$ chmod +x /home/alumno/my-script.sh
```

y, posteriormente, ya se puede lanzar únicamente dado la ruta al script:

```
$ /home/alumno/my-script.sh
```

O si estás en `/home/alumno`:

```
$ ./my-script.sh
```

> **Nota**
>
> Como ejercicio, crea un script que borre todo lo que este script genera.

2.2.2 Buscando con `grep` y `find`

Una de las tareas más recurrentes es buscar en los archivos y directorios. `grep` es un comando muy potente para buscar patrones en contenido de archivos. `find`, por su parte, es un comando para encontrar archivos y directorios cuyo nombre satisfaga un patrón determinado.

La estructura principal de `grep` es:

```
$ grep [OPTIONS] PATTERN [FILE FILE ...]
```

El argumento `PATTERN` es la cadena que quieras buscar. Esta cadena puede ser simplemente una palabra o incluso una *expresión regular* la cual nos permite definir patrones genéricos. Veamos algunos ejemplos sencillos:

```
$ grep auto /etc/network/interfaces
$ grep -i AuTo /etc/network/interfaces
$ grep -i AuTo --color /etc/network/interfaces
$ grep -i AuTo --color -H /etc/network/interfaces
$ grep -i AuTo --color -RH /etc/network
```

Por su parte, la estructura principal de `find` es:

```
$ find [STARTING-POINT] [OPTIONS]
```

Donde `STARTING-POINT` es la ruta desde donde se tiene que empezar a buscar archivos. En `OPTIONS` tienes todo tipo de filtros por lo que puedes afinar nuestra búsqueda (por nombre, por tamaño, por tipo, etc.). Estos son algunos ejemplos:

```
$ find
$ find /usr/bin
$ find /usr/bin -type f
$ find /usr/bin -type f -perm 755
$ find /usr/bin -type f -perm 755 -size +100k
$ find /usr/bin -type f -perm 755 -size +100k -mtime +100
```

2.2.3 Redirecciones

Como hemos dicho, los comandos son programas. Y los programas, al ejecutarse, crean *procesos* de ejecución. En un sistema GNU/Linux, un proceso tiene 3 puntos por los cuales puede comunicarse con el exterior. Por defecto, a todo proceso se le asigna estos *descriptores de archivo*:

- `stdin` o entrada estándar, por donde el comando puede recibir información. Por defecto, la entrada estándar es el teclado.
- `stdout` o salida estándar, por donde el comando produce su salida. Por defecto, la salida estándar es la propia línea de órdenes.
- `stderr` o salida de error estándar, por donde el comando puede emitir mensajes de error o de depuración. Por defecto, también es la propia línea de comandos.

> **Nota**

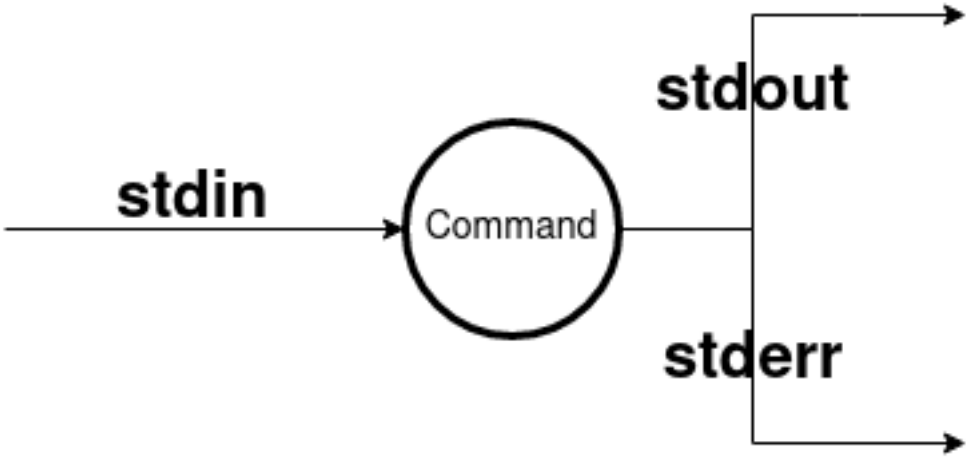

Figura 2.2: Descriptores de archivo por defecto de un comando.

La salida estándar y la de error se separan, fundamentalmente, para distinguir qué partes deben ser tomadas como puramente salidas del comando de los mensajes de error. Esto es muy importante cuando utilizamos redirección o *tuberías*.

En GNU/Linux, es posible cambiar a los sitios que apuntan estos descriptores de archivo de los comandos. Para ello, se utilizan los siguientes operadores:

- `cmd > file.txt`: redirige la salida del comando `cmd` en el fichero `file.txt`.
  - Si el fichero no existe, se crea.
  - Si el fichero existe, se reescribe completamente.

```
$ cat /etc/fstab > my-fstab.txt
$ echo "Hello!" > greetings
$ echo "World!" > greetings
```

Nota

Existe un fichero especial a donde va a parar todo lo que no se quiere almacenar: `/dev/null`. Es muy común redirigir la salida de comandos a este fichero para ignorar la misma.

- `cmd >> file.txt`: redirige la salida del comando `cmd` en el fichero `file.txt`.
  - Si el fichero no existe, se crea.
  - Si el fichero existe, la salida se añade al final del archivo.

```
$ cat /etc/fstab > my-fstab.txt
$ echo "# this goes at the end!" >> my-fstab.txt
```

- `cmd < file.txt`: redirige la entrada estándar del comando `cmd` de forma que `file.txt` sea su entrada y no el teclado.

```
$ cat < /etc/fstab
```

Ejercicios

1. Almacena en el archivo `libs.txt` los archivos (de forma recursiva) que terminen por `.so` desde el directorio `/usr/lib`.

   ##### Solución

   ```
   $ find /usr/lib -type f -name *.so > libs.txt
   $ cat libs.txt
   ```

2. Añade al final del mismo archivo aquellos que terminen en `.a`.

   ##### Solución

   ```
   $ find /usr/lib -type f -name *.a >> libs.txt
   $ cat libs.txt
   ```

> **Nota**
>
> Por simplicidad, no se menciona cómo manipular la salida de error, pero también es posible redirigirla como los otros dos.

2.2.4 Tuberías

Las tuberías son, probablemente, la característica más atractiva de un entorno de línea de órdenes en GNU/Linux. Proporciona la capacidad de realizar tareas no triviales de una forma sencilla, simple y clara.

El concepto parte de lo que se ha visto en *redireccionamiento*, donde se vio que todos los comandos tenían una entrada y una salida que se podía manipular. El operador de tubería (denotado con la barra vertical `|`) conecta la salida de un comando con la entrada de otro. De esta forma, se pueden unir comandos que realizan diferentes tareas para realizar, en global, una tarea más compleja.

> **Nota**
>
> La filosofía UNIX está detrás de las tuberías: se proporcionan muchos programas que hacen cosas específicas para luego componerlos entre ellos y crear funcionalidad avanzada.

Por ejemplo:

```
$ find /usr/lib -name *.so -type f | less
$ cat /etc/passwd | grep 100 --color
$ grep -Hiw network /usr/share/common-licenses/* | cut -f 1 -d ":" | sort -u
```

Ejercicios

- Descarga el fichero de Don Quijote en texto plano y cuenta el número de palabras que contiene. Puedes usar el comando `wc` para contarlas.
- ¿Sabrías hacer todo el proceso, incluso descargar el archivo, con varios comandos en cadena?

Solución

```
$ wget -q https://uclm-esi.github.io/redes1-lab/assets/quijote.txt -O - | wc -w
```

2.2.5 Permisos de administrador

Como ya habrás comprobado, hay muchos elementos del sistema de archivos a los cuales no puedes acceder por faltar de permisos. Por ejemplo:

```
$ cat /var/log/messages
cat: /var/log/messages: Permission denied
```

Como ya vimos en la sección de *permisos*, es necesario que tengas los permisos adecuados para acceder a un archivo para poder realizar la operación de escritura, lectura o ejecución. Sin embargo, también hay otro tipo de acciones que, por seguridad, están reservadas a usuarios que tengan permisos de administración. Algunas de estas operaciones son:

- Añadir usuarios o grupos.
- Cambiar las claves de otros usuarios.
- Instalar programas nuevos en el sistema.
- Acceso a bajo nivel de las interfaces de red.

En nuestra máquina virtual, el usuario `alumno` viene configurado de forma que pueda ejecutar comandos como usuario administrador cuando lo necesite. Para ello, se usa el comando `sudo` y, a continuación, el comando que se quiera ejecutar. Por ejemplo:

```
$ sudo cat /var/log/messages
```

2.2.6 Sistemas de paquetes

Si hay algún aspecto que caracteriza a una distribución de GNU/Linux es el sistema de paquetes que utiliza. Una distribucicón se basa en recolectar software que hay disponible como software libre o código abierto, agruparlo y confirgurarlo de forma que los usuarios puedan acceder a ellos de una forma rápida y sencilla. El sistema de paquetes es el componente con el que los usuarios pueden gestionar los programas que instalan.

En Debian, los diferentes sistemas de paquetes se basan en un formato muy determinado: los *paquetes Debian*. Estos paquetes, empaquetados en archivos `.deb`, contienen normalmente software ya compilado y listo para usar, además de información extra como dependencias de otros programas, autor del programa, etc.

dpkg

Los paquetes Debian pueden instalarse directamente con el comando `dpkg`. Sin embargo, no puedes instalarlos como usuario normal, sino que tienes que tener permisos de administrador:

```
$ sudo dpkg -i <path/to/file.deb>
```

> **Advertencia**
>
> Si el paquete tiene dependencias `dpkg` *no* las resolverá y no tratará de instalarlas. Para hacer eso, debes usar `apt` como veremos después.

También se puede eliminar un paquete como:

```
$ sudo dpkg -r <package-name>
```

> **Nota**
>
> El nombre de paquete es *diferente* al nombre del fichero `.deb`. Por ejemplo: el paquete `myapp-1.0.deb` seguramente tenga el nombre de paquete como `myapp`.

Y algunas opciones interesantes para listar los paquetes del sistema y los ficheros que contine un paquete:

```
$ dpkg -l
$ dpkg -L <package-name>
```

apt

En un nivel superior de gestión de paquetes se encuentra `apt`. Esta herramienta, que utiliza `dpkg` por debajo, proporciona una mayor funcionalidad ya que:

- Resuelve dependencias entre paquetes, instalando todo lo que sea necesario.
- Accede a repositorios remotos. `apt` puede conectarse a un «store» de aplicaciones para descargarlas e instalarlas automáticamente.
- Búsqueda de paquetes en base a diferentes criterios (contenido, descripción, tipo, etc.)

Las formas de utilizar `apt` más comunes son:

```
$ sudo apt update
$ sudo apt search <pattern>
$ sudo apt show <package-name>
$ sudo apt install <package-name>
$ sudo apt remove <package-name>
$ sudo apt purge <package-name>
$ sudo apt upgrade
$ sudo apt full-upgrade
```

Ejercicios

- Descarga el paquete `redes1` e instálalo usando `dpkg`. ¿Qué archivos tiene? ¿Puedes usar alguno? Finalmente, bórralo usando `dpkg`.

Solución

```
$ dpkg -L redes1
```

Se puede ejecutar `redes1` y sale un mensaje.

- Instala los paquetes `sl` y `tree` utilizando `apt`. ¿Qué es lo que hacen según su descripción? Desinstala los paquetes completamente.

Solución

```
$ sudo apt update
$ sudo apt install tree sl
$ apt show tree
$ apt show sl
$ sudo apt purge tree sl
```

2.3 Práctica 3 - Cliente y servidor

En esta sesión vamos a profundizar en los conceptos de cliente y servidor desde un punto de vista práctico y, al mismo tiempo, aprender una utilidad llamada *netcat* (comando `nc`): una herramienta básica que se utiliza, entre otros usos, para diagnosticar problemas en redes.

El modelo de comunicación cliente-servidor se describe gráficamente en la siguiente figura:

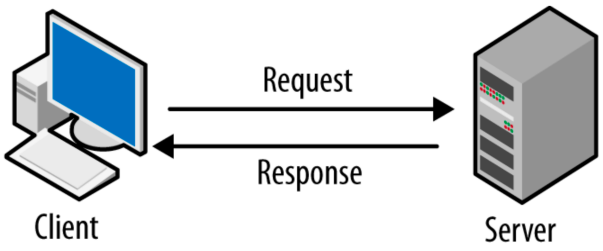

Figura 2.3: Esquema básico de comunicación entre un cliente y un servidor

El cliente realiza peticiones al servidor y el servidor responde a estas peticiones. Algunas consideraciones a tener en cuenta:

- El servidor tiene un comportamiento *reactivo*, es decir, espera a que le lleguen peticiones para actuar en la comunicación.
- El cliente tiene un comportamiento *proactivo*, es decir, realiza las peticiones al servidor cuando lo necesita y es quien *inicia* la comunicación.
- Un nodo puede tener los 2 roles a la vez. Es posible que, dentro de un entorno más complejo, un mismo nodo pueda ser el cliente de un servidor y servidor de otros nodos diferentes. La separación de estos dos roles es *lógica*, no física.

2.3.1 *netcat*

Para crear conexiones simples entre nodos utilizando el modelo cliente-servidor una de las herramientas más sencillas es *netcat*. En el terminal, se usa con el comando `nc`. Si no lo tienes instalado, debes instalar el paquete `netcat`:

```
$ sudo apt install netcat
```

2.3.2 Casos prácticos

Los siguientes casos prácticos están pensados para realizarse en el laboratorio por parejas (cada miembro de la pareja con su PC de laboratorio).

> **Nota**
>
> También sería posible practicar desde casa utilizando 2 máquinas virtuales lanzadas a la vez. Cada máquina virtual tendrá su IP y, por tanto, una puede actuar de servidor y otra de cliente. Sólo tienes que tener 2 máquinas virtuales importadas.

Un miembro de la pareja actuará de cliente y el otro actuará de servidor. Os recomendamos que se intercambien los roles en diferentes ejercicios para que ambos miembros hagáis de cliente y de servidor.

Para poder realizar la comunicación entre ellos es necesario conocer las direcciones IP de ambos. Podemos conocer las direcciones utilizando el comando `ip`:

```
$ ip addr
```

También es posible obtener las direcciones IP utilizando el comando `ifconfig` aunque es preferible el primer método:

```
$ sudo ifconfig
```

Estos comandos nos muestran las direcciones IP de las que dispone nuestra máquina. Tiene la forma de 4 bytes separados por puntos. Por ejemplo: `192.168.1.120`. Las direcciones IP van asociadas a las *interfaces de red* de las que disponga el nodo. Como nuestra máquina virtual sólo tiene una interfaz de red (simulada) sólo veremos una IP válida.

> **Nota**
>
> Ambos comandos muestran una interfaz de red especial llamada `lo` cuya IP es siempre `127.0.0.1`. Esta interfaz corresponde a la red local de la propia máquina y, de momento, la debemos ignorar.

Una vez tomada nota de qué IP corresponde con el cliente y cuál con el servidor, puedes comenzar con los siguientes ejercicios prácticos de comunicación entre ambos.

Chat entre cliente y servidor

Servidor

Teclea en el servidor el comando:

```
$ nc -l -p 5400
```

El servidor se pone a la escucha con la opción `-l` (*listen*) en el puerto `5400`. Por defecto lo hace en TCP, aunque con la opción `-u` se podría hacer para UDP.

Cliente

Una vez el servidor esté a la escucha, teclea en el cliente:

```
$ nc <server-IP> 5400
```

Substituye `<server-IP>` por la IP del servidor a la que quieres conectarte. Esto hará que `nc` abra una conexión TCP con el servidor.

Una vez conectados, el cliente se comunica con el servidor como si compartieran teclado y pantalla: cualquier texto que introduzca cualquiera de ellos será visto por ambos cada vez que se confirme con `Enter`.

Para abortar la comunicación basta con pulsar `Ctrl c` sobre cualquiera de los dos. Esto lo que hace es abortar el programa y cerrar cualquier comunicación que esté en proceso.

Enviar texto y almacenarlo

Servidor

Teclea en el servidor el comando:

```
$ nc -l -p 5400 > output.txt
```

Al igual que antes, el servidor se pone a la escucha en el puerto `5400`. Ahora lo que reciba se redirigirá al fichero `output.txt`. En cuanto se conecte el cliente y empiece a enviar datos, el fichero irá adquiriendo contenido.

Una vez cerrada la conexión se puede ver el contenido del archivo con `cat`. No olvides borrar el archivo con `rm`.

Cliente

Cuando el servidor esté escuchando, teclea en el cliente el comando:

```
$ nc <server-IP> 5400
```

Escribe una frase de prueba como: `You'll be free, hackers, you'll be free.`

La tecla `Enter` no cierra la conexión, simplemente introduce un salto de línea. Cuando se acabe de introducir por teclado el texto, puedes cerrar la conexión con `Ctrl c`.

Ver el contenido recibido por el servidor

Servidor

Teclea en el servidor el comando:

```
$ nc -l -p 5400
```

El servidor se pone a la escucha en el puerto `5400`. Ahora lo que reciba se mostrará por pantalla.

Cliente

Crea un archivo llamado `file.txt` con un contenido cualquiera, por ejemplo: `At our call, hackers, at our call.`. Puedes utilizar `nano` para crearlo.

```
$ nc <server-IP> 5400 < file.txt
```

No olvides borrar el archivo con `rm` al finalizar.

Envío de fichero por parte del cliente (subida)

Servidor

Teclea en el servidor el comando:

```
$ nc -l -p 5400 > book.pdf
```

El servidor se pone a la escucha en el puerto `5400` y lo que reciba se almacenará en el fichero `book.pdf`.

Una vez finalizado, no olvides borrar el archivo con `rm`.

Cliente

Descarga el libro de prácticas de la asignatura con `wget`. Substituye `en` por `es` si quieres la versión en español:

```
$ wget https://uclm-esi.github.io/redes1-lab/en/redes1-lab-en.pdf
```

Ahora lo puedes enviar al servidor usando:

```
$ nc <server-IP> 5400 < redes1-lab-en.pdf
```

Una vez finalizado, no olvides borrar el PDF con `rm`.

Nota

Se puede ver el contenido del PDF abriendo el archivo con el comando `evince`, que abrirá un programa gráfico para visualizar el PDF que se pase por argumentos.

Debes cerrar el PDF para poder retomar el control del terminal, que se queda bloqueado mientras tanto. También puedes utilizar `Ctrl c` para finalizar el programa desde el terminal.

El nombre del archivo con que se guarda el PDF a cada lado de la comunicación puede ser el mismo o diferente. Cada extremo de la comunicación le asigna un nombre.

Almacenar datos que vienen del servidor

Servidor

Teclea en el servidor el comando:

```
$ nc -l -p 5400
```

El servidor se pone a la escucha en el puerto `5400`, pero también puede recibir entrada por teclado. Esta entrada la enviará al cliente cuando se conecte.

Vamos a probar esto. Teclea una frase como por ejemplo: `We'll kick out those dirty licenses`.

Una vez se conecte el cliente, este irá recibiendo el texto que has introducido. Si sigues introduciendo texto el cliente lo seguirá recibiendo.

Cliente

Espera a que el servidor haya introducido todo el texto. Cuando esté listo, ejecuta:

```
$ nc <server-IP> 5400 > file.txt
```

El fichero `file.txt` se actualizará con los datos que el servidor vaya enviando. Puedes abrir otro terminal y con `cat` ver el contenido del archivo.

No olvides borrar el archivo generado con `rm`.

Ver un fichero del servidor desde el cliente

Servidor

Crea un archivo llamado `file.txt` con un contenido cualquiera, por ejemplo: `Join us now and share the software`. Puedes utilizar `nano` para crearlo.

```
$ nc -l -p 5400 < file.txt
```

El servidor se pone a la escucha en el puerto `5400` y en cuanto se conecte el cliente recibirá el contenido del archivo.

No olvides borrar el archivo creado con `rm` al finalizar.

Cliente

Cuando el servidor esté listo, ejecuta:

```
$ nc <server-IP> 5400
```

Aparecerá por pantalla el contenido del archivo que el servidor ha puesto a disposición.

Envío de fichero por parte del servidor (descarga)

Servidor

Descarga el libro prácticas de la asignatura con `wget`. Substituye `en` por `es` si quieres la versión en español:

```
$ wget https://uclm-esi.github.io/redes1-lab/en/redes1-lab-en.pdf
```

Esto creará el archivo `redes1-lab-en.pdf` en el directorio desde donde se lanzó `wget`. Ahora comienza el servidor con:

```
$ nc -l -p 5400 < redes1-lab-en.pdf
```

El servidor se pone a la escucha en el puerto `5400` y en cuanto se conecte el cliente recibirá el PDF.

No olvides borrar el archivo PDF con `rm` al finalizar.

Cliente

Cuando el servidor esté listo, ejecuta:

```
$ nc <server-IP> 5400 > book.pdf
```

Cuando se conecte, el servidor enviará el PDF y el cliente lo almacenará en el archivo *book.pdf*.

2.4 Práctica 4 - Captura de tráfico de red

Cuando un ordenador envía o recibe tráfico a la red, lo hace a través de sus interfaces de red. Dependiendo de la configuración de la red y de nuestro propio equipo, puedes capturar dicho tráfico de red para poder estudiarlo, analizarlo o incluso guardarlo en un fichero para utilizarlo más tarde.

Para poder realizar esta captura de tráfico, necesitamos un software específico. En esta práctica vamos a aprender a utilizar el programa `tshark` que nos permitirá realizar las funciones mencionadas.

También existe la posibilidad de usar `wireshark`, una versión sobre entorno gráfico de `tshark`. Todo lo que se va explicar para uno, sirve también para el otro.

2.4.1 Preparación del entorno

Configuración de la red

En primer lugar, debes asegurarte de tener conexión de red. Para ello, abre el terminal y ejecuta lo siguiente:

```
$ ip addr
```

En la salida deben aparecer todas las interfaces de red de tu equipo con sus direcciones MAC y sus direcciones IP si las tuvieran.

Dependiendo de la red en la que te encuentres, la IP tendrá un valor diferente. Recuerda que siempre debe tener la forma de 4 bytes (valores del 0 al 255) separados por puntos, por ejemplo `192.168.1.120`.

En principio, si usas la *máquina virtual*, tanto en los ordenadores de los laboratorios como en tu PC, no deberías tener problema a la hora de realizar las capturas de tráfico.

Instalación del software de captura

Como se mencionó anteriormente, los programas que usaremos para capturar tráfico son `tshark` y `wireshark`.

Para comprobar si tienes estos programas instalados, utiliza:

```
$ command -v tshark
$ command -v wireshark
```

Si al ejecutar los comandos anteriores no apareciera la ruta de alguno de los programas, significa que no está instalado, así que instálalo usando:

```
$ sudo apt install tshark wireshark
```

2.4.2 Conociendo `tshark`

`tshark` es una herramienta de terminal que permite capturar el tráfico de red y guardarlo en un fichero o mostrarlo en la salida estándar. Para realizar una captura de tráfico, lo primero que debemos hacer es averiguar en qué interfaz o interfaces de red del equipo queremos realizar la captura.

Para ello, utilizaremos tanto el comando `ip addr`. Con él podemos ver las direcciones IP de las interfaces de red, pudiendo averiguar en cuál de ellas tenemos la conexión configurada.

> **Nota**
>
> Si utilizas la máquina virtual, deberías ver 2 interfaces: una llamada `lo` o interfaz *loopback*, que da conectividad IP con la propia máquina, y una segunda interfaz cuyo nombre dependerá del entorno, que será la que nos de conectividad hacia el exterior de la máquina. Debes fijarte en la segunda para realizar capturas de tráfico.

Una vez sepas los nombres de la interfaz o interfaces de red que quieras utilizar, puedes usar el comando `tshark -D` para ver la correspondencia entre el índice a utilizar y el nombre de la interfaz:

```
$ tshark -D
Running as user "root" and group "root". This could be dangerous.
1. enp0s3
2. any
3. lo (Loopback)
4. ciscodump (Cisco remote capture)
5. dpauxmon (DisplayPort AUX channel monitor capture)
6. sdjournal (systemd Journal Export)
7. sshdump (SSH remote capture)
8. udpdump (UDP Listener remote capture)
```

Para las prácticas normalmente utilizaremos la interfaz de red de nuestra máquina virtual, así que selecciona el índice correspondiente a la interfaz listada con `ip addr` (en este caso, el **1**).

Nuestra primera captura

Para lanzar la primera captura, abre tu terminal y ejecuta:

```
$ tshark -i 1
```

Este comando empezará a capturar tráfico en la interfaz número `1` (en este caso la interfaz de red de la máquina virtual como vimos antes). El comportamiento por defecto es mostrar un breve resumen de los paquetes capturados en la salida del programa. Pulsa `Ctrl c` para detener la captura.

```
Capturing on 'enp0s3'
 ** (tshark:594982) 00:11:55.871189 [Main MESSAGE] -- Capture started.
 ** (tshark:594982) 00:11:55.871276 [Main MESSAGE] -- File: "/var/tmp/wireshark_
↪wlp0s20f35RkW4P.pcapng"
    1 0.000000000 192.168.1.39 → 255.255.255.255 UDP 909 54712 → 29810 Len=863
    2 3.684236270 fe80::9a97:d1ff:fe77:9368 → ff02::1      ICMPv6 82 Router␣
↪Advertisement from 98:97:d1:77:93:68
    3 4.094190431 192.168.1.91 → 255.255.255.255 UDP 214 49153 → 6667 Len=172
    4 4.094193127 192.168.1.91 → 255.255.255.255 UDP 218 49153 → 6667 Len=172
    5 2.247868733 MitraSta_77:93:68 →              ARP 62 Who has 192.168.1.95? Tell␣
↪192.168.1.1
    6 2.247899195 IntelCor_d1:62:2b →              ARP 44 192.168.1.95 is at␣
↪c8:09:a8:d1:62:2b
```

(continúe en la próxima página)

(proviene de la página anterior)

```
    7 5.119982488 192.168.1.39 → 255.255.255.255 UDP 905 54712 → 29810 Len=863
    8 5.119985457 192.168.1.39 → 255.255.255.255 UDP 909 54712 → 29810 Len=863
    9 9.009472416 192.168.1.91 → 255.255.255.255 UDP 214 49153 → 6667 Len=172
   10 9.009534174 192.168.1.91 → 255.255.255.255 UDP 218 49153 → 6667 Len=172
```

Tras unos segundos verás una salida similar a la anterior. En ella, aparte de un primer mensaje confirmando que la captura se está realizando en la interfaz `enp0s3` y un par de mensajes de log del propio `tshark`, aparece un resumen de cada paquete capturado por línea, indicando:

1. Número de paquete (ordinal) en la captura.
2. Tiempo transcurrido relativo con respecto al primer paquete capturado.
3. Direcciones IP origen y destino del paquete (o dirección MAC si es tráfico de una capa inferior).
4. Protocolo (TCP, UDP, ICMP, DNS…).
5. Tamaño del datagrama IP o Ethernet
6. Puerto origen y destino (sólo en caso de TCP y UDP).
7. Descripción corta del paquete.

Guardando una captura en un fichero

Cuando realizamos una captura en vivo, se corre el riesgo de no poder inspeccionar con el detalle deseado alguno de los paquetes capturados. Para evitar este comportamiento, es muy aconsejable hacer que `tshark` guarde la captura de tráfico en un fichero en disco:

```
$ tshark -i 1 -w /home/alumno/capture1.pcapng
```

De este modo, el programa no mostrará ninguna información en la salida, pero guardará toda la captura en la ruta que hayas pasado como argumento. Esto nos permite poder inspeccionar la captura de tráfico tantas veces como necesitemos, abrirla con otros programas, como el citado `wireshark` e incluso «inyectar» la captura de nuevo, simulando un tráfico ya pasado.

Para mostrar el contenido del fichero guardado con `tshark`, usa la siguiente opción:

```
$ tshark -r /home/alumno/capture1.pcapng
```

2.4.3 Filtrando las capturas de `tshark`

`tshark` permite definir 2 tipos de filtros:

- *Filtro de captura*: hace que sólo los paquetes que cumplan con el filtro definido sean mostrados y/o guardados en la captura.
- *Filtro de visualización*: muestra únicamente los paquetes que cumplan con la condición definida por el filtro, pero el resto se seguirán guardando en la captura.

Filtros de captura

Este tipo de filtros nos permite aligerar el tamaño de la captura resultante. Es adecuado cuando tenemos una idea precisa de qué paquetes queremos poder analizar en la captura.

Para realizar un filtrado de captura debemos utilizar la siguiente opción:

```
$ tshark -f <capture filter>
```

La sintaxis de los filtros de captura es la denominada BPF. Con ella, podemos hacer filtros como los siguientes:

Filtro	Resultado obtenido
icmp	Tráfico ICMP
host 192.168.1.1	Tráfico hacia o desde la IP 192.168.1.1
tcp portrange 1-1024	Paquetes TCP recibidos o enviados en cualquier puerto del rango 1 a 1024
tcp and not http	Paquetes TCP, pero que no contengan tráfico HTTP

Aunque los filtros de captura son muy útiles, sobre todo para aligerar el tamaño de las capturas, realmente son menos versátiles que los filtros de visualización que veremos en el siguiente apartado.

Ejercicio: filtra la captura de paquetes

1. En un terminal, inicia una captura de tráfico que sólo capture los paquetes ICMP. Haz que la captura se guarde en un fichero llamado `icmp_cap_filter.pcapng`.
2. Abre un segundo terminal y realiza un `ping` con 10 peticiones a `8.8.8.8`. Utiliza la opción `-c 10` para que el comando `ping` se detenga tras enviar las 10 peticiones.
3. Cuando `ping` haya finalizado, detén la captura con `Ctrl c`.

Filtros de visualización

Los filtros de visualización permiten que `tshark` capture todo el tráfico (o todo el que haya sido definido en los *filtros de captura*) y que sólo visualicemos aquellos paquetes que deseemos. Esto es mucho más flexible, ya que usando una misma captura podemos inspeccionar de forma sencilla diferentes tipos de tráfico usando diferentes filtros de visualización.

Además, al contrario que los filtros de captura que sólo pueden basarse en los propios campos que contienen los paquetes, los filtros de visualización pueden tomar en consideración información de contexto que `tshark` puede extraer del análisis de la traza completa.

Por ejemplo, con cualquiera de los 2 tipos de filtro podríamos encontrar sólo los paquetes ICMP, pero con el de visualización, podríamos solicitar ver sólo aquellos paquetes ICMP que no hayan obtenido una respuesta.

Aunque la sintaxis de los filtros es muy parecida a la de los filtros de captura vistos anteriormente, no cumplen exactamente las mismas reglas.

Para definir un filtro de visualización:

```
$ tshark -Y <display filter>
```

Por ejemplo, para ver sólo los paquetes ICMP enviados desde la IP `192.168.1.11`, usamos el siguiente filtro:

```
$ tshark -Y 'ip.src == 192.168.1.11 and icmp'
```

> **Nota**
>
> Se recomienda utilizar comillas sencillas para delimitar el filtro desde comandos de terminal para evitar que el intérprete de comandos divida el filtro o se puedan expandir variables o comodines de terminal.

Para los protocolos de capas 2 a la 4, `tshark` permite filtrar por la dirección de envío, recepción o cualquiera de ellas:

Protocolo	Origen	Destino	Cualquiera
Ethernet	eth.src	eth.dst	eth.addr
IP	ip.src	ip.dst	ip.addr
TCP	tcp.srcport	tcp.dstport	tcp.port
UDP	udp.srcport	udp.dstport	udp.port

Cualquiera de esos campos puede ser comparado con los operadores de igualdad o desigualdad (== o `!=`).

Ejercicio: filtros de visualización

1. En un terminal nuevo, deja lista para ejecutar el comando para descargar el archivo phone_new.gif, usando por ejemplo `wget`, pero sin ejecutar el comando por ahora.

   ```
   $ wget http://www.esi.uclm.es/www/isanchez/phone_new.gif
   ```

2. Arranca una captura todo el tráfico con `tshark` sin utilizar ningún filtro de captura. Guarda dicha captura en un fichero llamado `http_get_gif.pcapng`.
3. Ejecuta el comando que preparaste en el paso 1.
4. Detén `tshark` para finalizar la captura con `Ctrl c`.
5. Carga la captura de nuevo en `tshark` y aplica un filtro de visualización que muestre sólo los paquetes del protocolo DNS.

   ```
   $ tshark -r /home/alumno/http_get_gif.pcapng -Y 'dns'
   ```

6. ¿A qué IP (servidor DNS) se ha conectado tu máquina para hacer la petición DNS sobre el dominio `www.esi.uclm.es`?
7. Carga la captura de nuevo en `tshark` y aplica un filtro de visualización que muestre sólo los paquetes del protocolo HTTP.

   ```
   $ tshark -r /home/alumno/http_get_gif.pcapng -Y 'http'
   ```

8. ¿Cuál es la IP del servidor que hospeda la imagen?

> **Nota**
>
> En los filtros de visualización, los protocolos deben escribirse en minúsculas.

2.4.4 Modificar la salida de `tshark`

En ocasiones, la salida que proporciona el comando `tshark` no incluye algún campo que nos puede resultar relevante. Por supuesto, el comando nos permite modificar su salida de manera que nos muestre aquellos datos relevantes que necesitemos.

Para modificar los campos mostrados por defecto, deberemos activar la opción `-T`, indicando en qué formato queremos mostrar la salida (por ahora utiliza `fields`), y con `-e` indicaremos las columnas que queremos mostrar. Podemos usar tantas veces `-e` como necesitemos:

```
$ tshark -r cap1.pcapng -T fields -e ip.addr -e tcp -e udp -e _ws.col.Info
```

El comando anterior nos mostraría las direcciones IP de origen y destino, información relevante sobre TCP o UDP, según el protocolo de transporte utilizado por cada paquete, y por último una columna especial que contiene el resumen breve del contenido del paquete.

> **Nota**
>
> Para ver una lista completa de todos los campos que `tshark` permite utilizar con `-e`, puedes listarlos con `tshark -G fields`. La lista es bastante grande, así que utiliza los comandos aprendidos en las primeras prácticas para filtrar o paginar el contenido de la salida.

Ejercicio:

1. Utiliza la captura realizada en el bloque anterior de ejercicios llamada `http_get_gif.pcapng`.
2. Carga la captura con `tshark` y, filtrando sólo los paquetes DNS, modifica la salida para que se muestre la información relativa al protocolos de transporte utilizado.

   ```
   $ tshark -r /home/alumno/http_get_gif.pcapng -Y 'dns' -T fields -e udp
   ```

3. ¿Cuáles han sido los puertos de origen y destino de las peticiones DNS?.
4. Carga de nuevo la captura anterior, pero esta vez filtra sólo los paquetes HTTP. Modifica la salida para que se muestre información relativa al protocolo de transporte utilizado.

   ```
   $ tshark -r /home/alumno/http_get_gif.pcapng -Y 'http' -T fields -e tcp
   ```

5. ¿Qué puertos TCP ha utilizado la petición HTTP?
6. Ejecuta los siguientes comandos. Di qué observas. ¿A qué crees que es debido?

   ```
   $ tshark -r /home/alumno/http_get_gif.pcapng -Y 'dns' -T fields -e tcp
   $ tshark -r /home/alumno/http_get_gif.pcapng -Y 'http' -T fields -e udp
   ```

7. Averigua el código de respuesta de la petición HTTP que descargó el fichero *gif* del ejemplo:

   ```
   $ tshark -r /home/alumno/http_get_gif.pcapng -Y 'http' -T fields -e _ws.col.Info
   ```

2.4.5 Usando `wireshark`

Hasta ahora hemos visto el uso de `tshark`, que es una herramienta de línea de comandos. Su uso está recomendado para realizar capturas con filtros sencillos o para analizar paquetes a niveles muy básicos, como direcciones físicas, de red o puertos, así como para filtrar por protocolos de manera rápida.

Su principal ventaja es que nos permite redirigir la salida, aprovechándonos de tuberías a otros comandos para filtrar información, colorearla, o guardarla en un formato legible sin tener que aprender otras herramientas.

Sin embargo, para realizar análisis más profundos, `tshark` puede ser demasiado complejo, por lo que para esas situaciones está bien utilizar `wireshark` en su lugar.

Arrancando `wireshark`

Al igual que `tshark`, su versión gráfica también nos permite realizar capturas y visualizarlas: podemos abrir ficheros `pcapng` generados con `tshark` sin ningún problema y también generar ficheros para poderlos analizar con `tshark` o el propio `wireshark` más adelante.

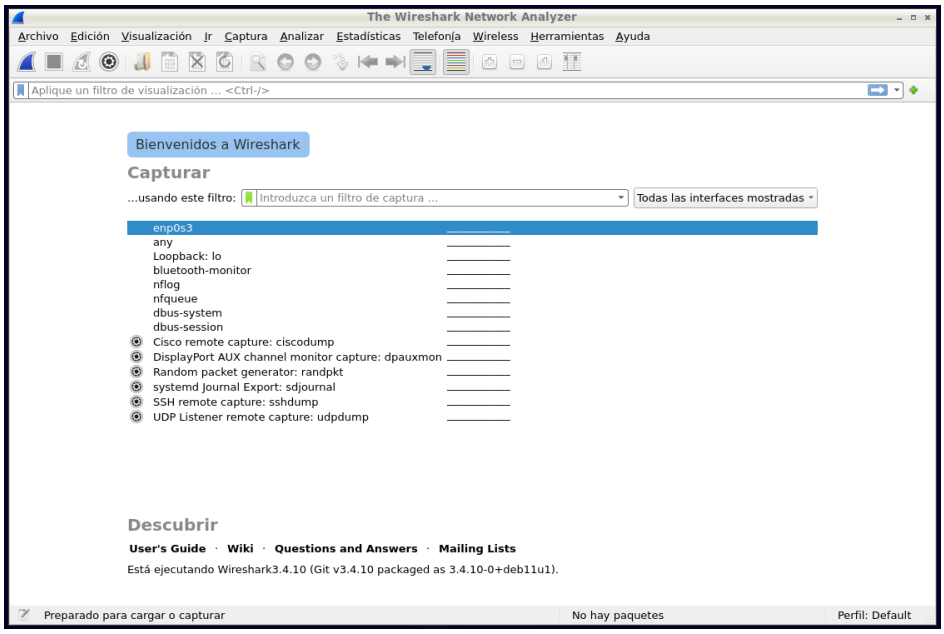

Figura 2.4: Captura de pantalla de la ventana de inicio de Wireshark

Desde la pantalla inicial podemos seleccionar tanto iniciar una captura, definiendo un filtro de captura utilizando la misma sintaxis vista anteriormente, como abrir un fichero de captura previamente guardado desde el menú `Archivo`.

Pulsando en el botón azul podemos iniciar una captura de tráfico, y en la caja de entrada de texto bajo `Capturar` podemos definir un filtro de captura.

Pantalla de análisis

Mientras `wireshark` está realizando una captura o bien hayamos cargado una desde un fichero, podrás ver una ventana como la siguiente:

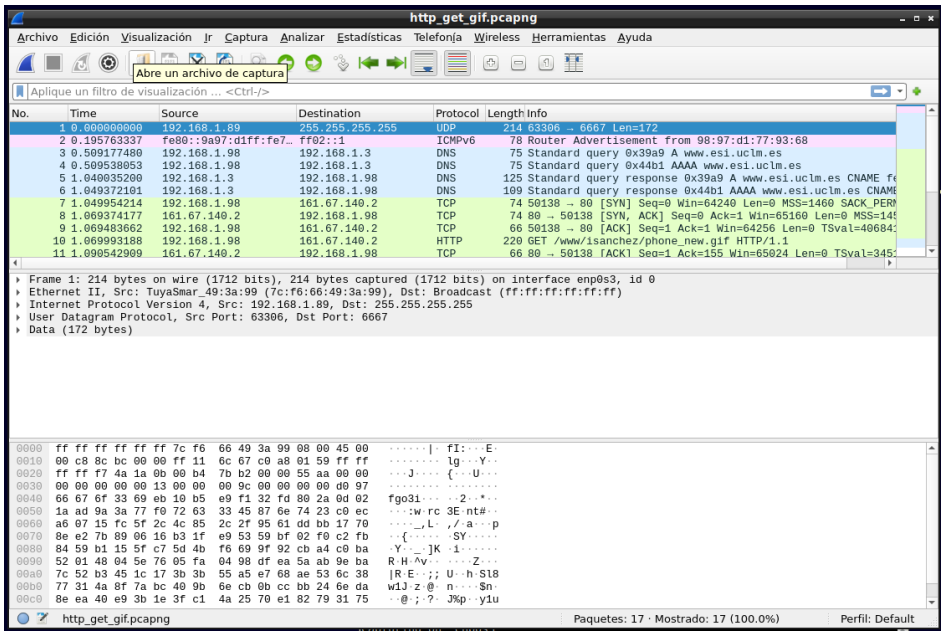

Figura 2.5: Captura de pantalla de la ventana de analisis de capturas en Wireshark

Dicha ventana está dividida en 4 partes importantes, de arriba a abajo:

1. Entrada de *filtro de visualización*: nos permite definir un filtro de visualización, de la misma manera que con `tshark -Y`.
2. Lista de *paquetes*: aparecen los paquetes de la captura y una serie de información básica de los mismos, una vez aplicado el filtro definido, si es que lo hubiera.
3. *Desglose* del paquete seleccionado: cuando tengamos un paquete seleccionado, aquí podremos ver información interna de cada capa de protocolos.
4. *Contenido* hexadecimal y ASCII de los paquetes: nos muestra la información al nivel más bajo posible, pudiendo ver en todo momento la correspondencia entre el valor hexadecimal de cada byte del paquete y su representación ASCII si la tuviera.

Ejercicio: analizando una captura con `wireshark`

1. Al igual que en el ejercicio anterior, abre la captura `http_get_gif.pcapng` pero en esta ocasión, con `wireshark`.
2. Encuentra los mismos valores que se solicitaron en el ejercicio anterior.
3. Encuentra en la captura la petición HTTP en la que se solicita la descarga del fichero GIF del servidor y localiza las cabeceras del protocolo HTTP que han sido utilizadas como por ejemplo el `user agent`.
4. Repite lo mismo, pero con la respuesta HTTP.

2.5 Práctica 5 - Análisis de la capa de aplicación

El objetivo de esta práctica es realizar la captura del tráfico de red correspondiente al intercambio de mensajes en los que intervienen protocolos a nivel de aplicación, su análisis y la identificación de los diferentes campos de los mensajes. Concretamente, en esta sesión se verán los protocolos DNS y HTTP.

2.5.1 Evaluación

Esta práctica es evaluable (**0.5 puntos** sobre los 3.0 puntos totales de las prácticas), por lo que deberás acceder al cuestionario correspondiente en Campus Virtual y responder a las preguntas. Las preguntas que hay que responder estarán disponibles mientras se realiza la sesión de prácticas y hay que entregar antes de la finalización de la misma. Cada respuesta errónea *se penalizará con un 1/3* del valor asignado a cada pregunta.

2.5.2 Preparación

Para la realización de la práctica es necesario que la conexión de red esté configurada correctamente:

1. Se suponen conocidos aspectos ya vistos en prácticas anteriores, por ejemplo, el comando para conocer la propia IP y el manejo de Wireshark.
2. Prepara tu directorio de trabajo:
   - En `/home/alumno` crea un directorio llamado `p5`. Asumiendo que tu directorio actual es `/home/alumno`, el comando sería:

     ```
     $ mkdir p5
     ```

   - Si ya existe, bórralo y créalo de nuevo

     ```
     $ rm -rf p5
     $ mkdir p5
     ```

   - Ve al directorio y realiza la práctica desde allí:

     ```
     $ cd p5
     ```

   - Comprueba que estás en el directorio adecuado con:

     ```
     $ pwd
     ```

     Debe devolver:

     ```
     /home/alumno/p5
     ```

3. Asegúrate que la interfaz de red de la máquina virtual tiene una dirección IP del tipo `172.24.21x.xxx`.
4. Asegúrate que puedes salir a Internet desde la máquina virtual. Por ejemplo, puedes ejecutar:

   ```
   $ ping -c 3 8.8.8.8
   ```

   Esto debe proporcionar un resultado similar a:

   ```
   3 packets transmitted, 3 received, 0% packet loss, time XXXms
   ```

5. Abre un navegador *dentro de la máquina virtual* y ten a mano estas instrucciones así como el formulario de Campus Virtual. Así te será más fácil seguirlas y descargar los archivos necesarios.

> **Importante**
>
> **No almacenes tu contraseña de Campus Virtual en la máquina virtual.**

2.5.3 DNS

dns1.pcapng

Esta captura se ha generado utilizando el siguiente comando:

```
$ host www.uclm.es
```

Este comando resuelve el nombre `www.uclm.es` en la dirección o direcciones IP a la que corresponde dicho nombre utilizando el protocolo DNS.

Descarga la captura `dns1.pcapng` y realiza un análisis con `wireshark` o `tshark`. Responde a las preguntas del cuestionario relacionadas con esta captura.

dns2.pcapng

Descarga la captura `dns2.pcapng` y analízala. Después, responde a las preguntas asociadas a la misma. Esta captura se ha generado utilizando el siguiente comando:

```
$ host redes1.com
```

2.5.4 HTTP

http1.pcapng

En este ejercicio se utilizará el comando `curl`. Se trata de una herramienta que permite realizar peticiones HTTP. La respuesta a cada petición la muestra por pantalla. Por defecto, el método que utiliza es `GET` que es el caso de este ejemplo. Se puede utilizar otro método a través de la opción `-X`.

En esta ocasión, debes crear una captura llamada `http1.pcapng` que contendrá tráfico HTTP. Para ello:

1. Arranca `wireshark`.
2. Ten a mano un terminal abierto.
3. Comienza a capturar tráfico en `wireshark`.
4. Ejecuta lo siguiente en el terminal:

   ```
   $ curl http://httpbin.org/get
   ```

   Debes tener una salida similar a la siguiente:


```
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.74.0",
    "X-Amzn-Trace-Id": "Root=1-621ff61f-71e17fc20c02922e7e26a8df"
  },
  "origin": "182.200.203.150",
```


(continúe en la próxima página)

(proviene de la página anterior)

```
  "url": "http://httpbin.org/get"
}
```

5. Para de capturar tráfico en `wireshark`.
6. Guarda la captura como `http1.pcapng`. Asegúrate que lo guardas en `/home/alumno/p5`.
7. Como filtro de visualización utiliza `http`.
8. Se deben mostrar 2 paquetes HTTP únicamente. Si no obtienes esta salida, debes repetir la captura.

Con los 2 paquetes HTTP mostrados ya puedes completar la parte del cuestionario asignada a este ejercicio.

http2.pcapng

Para generar el tráfico en este ejercicio se empleará el comando `wget` que sirve, entre otras muchas funcionalidades, para descargar ficheros utilizando HTTP. A diferencia de `curl`, este comando guarda en ficheros las respuestas del servidor, además de implementar funcionalidades de más alto nivel como seguir las redirecciones HTTP de forma automática. De hecho, esto mismo es lo que se hará en este ejemplo.

En definitiva, `wget` es una herramienta de más alto nivel que `curl` y permite descargar ficheros por HTTP a como lo haría un navegador convencional pero sin necesidad de entorno gráfico.

Utilizando el procedimiento anterior, ahora debes crear la captura llamada `http2.pcapng` de la siguiente forma:

1. Arranca `wireshark`.
2. Ten a mano un terminal abierto.
3. Comienza a capturar tráfico en `wireshark`.
4. Ejecuta lo siguiente en el terminal:

   ```
   $ wget http://www.uclm.es
   ```

   Debes tener una salida similar a la siguiente:

```
--2022-03-07 09:31:00--  http://www.uclm.es/
Resolving www.uclm.es (www.uclm.es)... 51.105.185.204
Connecting to www.uclm.es (www.uclm.es)|51.105.185.204|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://www.uclm.es/ [following]
--2022-03-07 09:31:00--  https://www.uclm.es/
Connecting to www.uclm.es (www.uclm.es)|51.105.185.204|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 113381 (111K) [text/html]
Saving to: 'index.html'

index.html              100%[=============================>] 110.72K  --.-KB/s  ␣
 ↪ in 0.03s

2022-03-07 09:31:00 (3.57 MB/s) - 'index.html' saved [113381/113381]
```

5. Para de capturar tráfico en `wireshark`.
6. Guarda la captura como `http2.pcapng`. Asegúrate que lo guardas en `/home/alumno/p5`.
7. Como filtro de visualización utiliza `ip.addr == 51.105.185.204`.
8. Debes ver diferentes tipos de tráfico. Además `wget` se ha debido de descargar un fichero llamado `index.html`. Si falta algo de esto, debes repetir la captura.

Con todos los requisitos cumplidos ya puedes analizar el tráfico y responder a la parte del cuestionario asignada a este ejercicio.

2.5.5 Finalización

Finalmente, borra el directorio generado `/home/alumno/p5` con:

```
$ rm -rf /home/alumno/p5
```

2.6 Práctica 6 - Capa de transporte I

El objetivo de esta práctica es estudiar las diferencias entre los protocolos de transporte UDP y TCP, tanto en los aspectos de rendimiento como de confiabilidad. Para ello se utilizará la herramienta `nc` para la transferencia de algunos ficheros, así como la herramienta `wireshark` para realizar las capturas correspondientes a las transmisiones.

Más específicamente, durante esta sesión vas a enviar varios ficheros entre dos procesos de tu equipo a través de la interfaz de *loopback* (`lo`). Se utilizarán diferentes ficheros de texto y de sonido, utilizando los protocolos de transporte TCP y UDP.

2.6.1 Evaluación

Esta práctica es evaluable (**0.5 puntos** sobre los 3.0 puntos totales de las prácticas), por lo que deberás acceder al cuestionario correspondiente en Campus Virtual. El cuestionario se cerrará al acabar la sesión. Cada respuesta errónea *se penalizará con un 1/3* del valor asignado a cada pregunta.

2.6.2 Preparación

Para la realización de la práctica es necesario que la conexión de red esté configurada correctamente:

1. Se suponen conocidos aspectos ya vistos en prácticas anteriores, por ejemplo, el comando para conocer la propia IP y el manejo de Wireshark y Tshark.
2. Prepara tu directorio de trabajo:
   - En `/home/alumno` crea un directorio llamado `p6`. Asumiendo que tu directorio actual es `/home/alumno`, el comando sería:

     ```
     $ mkdir p6
     ```
   - Si ya existe, bórralo y créalo de nuevo:

     ```
     $ rm -rf p6
     $ mkdir p6
     ```
   - Ve al directorio y realiza la práctica desde allí:

     ```
     $ cd p6
     ```
   - Comprueba que estás en el directorio adecuado con:

     ```
     $ pwd
     ```

     Debe devolver:

     ```
     /home/alumno/p6
     ```
3. Crea un directorio para ejecutar desde allí los clientes y otro para los servidores:

```
$ mkdir cliente servidor
```

4. Ten preparadas dos terminales diferentes; en una de ellas, sitúate en el directorio del cliente y en la otra, en el del servidor:

```
$ cd ~/p6/cliente
$ pwd
```

```
$ cd ~/p6/servidor
$ pwd
```

2.6.3 UDP

Para el análisis del funcionamiento del protocolo UDP se va a transmitir un fichero de texto entre 2 programas dentro del mismo equipo a través de la interfaz de red local o *loopback*. De manera simultánea ejecutarás el servidor, el cliente y realizarás la captura de los paquetes UDP con la herramienta que desees, Wireshark o Tshark.

udp-uuid.pcapng

1. En primer lugar, prepara el fichero que enviará el cliente hacia el servidor. Dicho fichero debe contener un «identificador único universal» (UUID). Este tipo de identificador son 32 caracteres hexadecimales agrupados en 5 grupos de diferente tamaño, utilizando el guión como separador. Para generarlo, ejecuta lo siguiente en el terminal del cliente:

   ```
   $ cat /proc/sys/kernel/random/uuid > uuid.txt
   ```

   El fichero debe tener un tamaño de 37 bytes (36 bytes del UUID y el carácter de nueva línea). Puedes comprobarlo con:

   ```
   $ ls -l uuid.txt
   -rw-rw-r--. 1 alumno alumno 37 Mar 20 16:58 uuid.txt
   ```

2. Lanza la captura con `wireshark`. Si lo deseas, guárdalo con el nombre de `udp-uuid.pcapng` para poderlo analizar posteriormente.

   > **Nota**
   >
   > Puedes indicar que se escuche sólo en la interaz «lo» (*loopback*) y utilizar un filtro de captura «udp», de modo que sólo se capture tráfico que viaje usando UDP como protocolo de transporte.

3. En el terminal del servidor, ejecuta el siguiente comando:

   ```
   $ nc -l -u -p 8888 > received-uuid.txt
   ```

   Los datos que reciba `nc` se redirigirán al fichero `received-uuid.txt`.

4. En el terminal del cliente, ejecuta el siguiente comando:

   ```
   $ nc -q 2 -u 127.0.0.1 8888 < uuid.txt
   ```

   En este ejemplo utilizamos `127.0.0.1`, que es la dirección IP asignada a la interfaz local o *loopback*. De ese modo nos comunicaremos con otro proceso dentro de nuestra máquina a través de la pila de protocolos de red.

5. Termina la ejecución del servidor pulsando `Ctrl c` en la ventana correspondiente.

6. Detén la captura. Comprueba que en ella aparece tráfico.

7. Responde a las preguntas asociadas a esta captura.

udp-quijote.pcapng

1. En este ejercicio vamos a utilizar una captura preexistente: descarga la captura udp-quijote.pcapng. En ella, el cliente envía el fichero de texto `quijote.txt` mediante la interfaz local o *loopback*, utilizando el protocolo de transporte UDP.
2. Abre la captura con `wireshark`.
3. Responde a las preguntas asociadas a esta captura.

2.6.4 TCP

En esta parte vamos a realizar el trabajo equivalente de la sección anterior, pero en esta ocasión utilizando TCP como protocolo de transporte y utilizando un fichero de texto y otro de sonido en lugar de los ficheros de texto.

tcp-quijote.pcapng

1. Para este ejercicio vamos a utilizar una captura preexistente: descarga la captura tcp-quijote.pcapng. En ella se replica la transmisión del ejercicio anterior, pero utilizando el protocolo de transporte TCP.
2. Abre la captura con `wireshark`.
3. Responde a las preguntas asociadas a esta captura.

tcp-mp3.pcapng

1. Descarga el audio MP3 en la carpeta del cliente. El fichero se debe llamar `p6audio.mp3`:

   ```
   $ wget https://uclm-esi.github.io/redes1-lab/assets/p6audio.mp3
   ```

2. Lanza la captura con `wireshark`. Si lo deseas, guárdalo con el nombre de `tcp-mp3.pcapng` para poderlo analizar posteriormente.

   > **Nota**
   >
   > Puedes indicar que se escuche sólo en la interaz «lo» (*loopback*) y utilizar un filtro de captura «tcp», de modo que sólo se capture tráfico que viaje usando TCP como protocolo de transporte.

3. En el terminal del servidor, ejecuta el siguiente comando:

   ```
   $ nc -l -p 8888 > audio_tcp_received.mp3
   ```

4. En el terminal del cliente, ejecuta el siguiente comando:

   ```
   $ nc -q 2 127.0.0.1 8888 < p6audio.mp3
   ```

   En este ejemplo utilizamos de nuevo `127.0.0.1` para que se utilice la interfaz local o *loopback*.

5. Detén la captura. Comprueba que en ella aparecen tanto los paquetes que envía el cliente como los que envía el servidor

   > **Nota**
   >
   > Fíjate en el puerto origen y destino para saber qué paquetes pertenecen a cada proceso.

6. Responde a las preguntas asociadas a esta captura utilizando como filtro de visualización `tcp.port == 8888`.

2.6.5 Finalización

Finalmente, borra el directorio generado `/home/alumno/p6` con:

```
$ rm -rf /home/alumno/p6
```

2.7 Práctica 7 - Capa de transporte II

El objetivo de esta práctica es estudiar las diferencias entre los protocolos de transporte UDP y TCP, tanto en los aspectos de rendimiento como de confiabilidad. Para ello se utilizará la herramienta `nc` para la transferencia de algunos ficheros, así como la herramienta Wireshark para realizar las capturas correspondientes a las transmisiones.

Más específicamente, durante esta sesión vas a enviar varios ficheros desde tu equipo (el cliente) a un *servidor remoto* (el servidor), que los recibirá y devolverá un eco de los mismos. Los ficheros serán dos, uno de texto y otro que será una imagen. Se van a utilizar en ambos casos los dos protocolos de transporte estudiados: UDP y TCP.

Importante

La IP del servidor remoto se proporcionará durante la sesión.

2.7.1 Evaluación

Esta práctica es evaluable (**0.5 puntos** sobre los 3.0 puntos totales de las prácticas), por lo que deberás acceder al cuestionario correspondiente en Campus Virtual. El cuestionario se cerrará al acabar la sesión. Cada respuesta errónea *se penalizará con un 1/3* del valor asignado a cada pregunta.

2.7.2 Preparación

Para la realización de la práctica es necesario que la conexión de red esté configurada correctamente:

1. Se suponen conocidos aspectos ya vistos en prácticas anteriores, por ejemplo, el comando para conocer la propia IP y el manejo de Wireshark.
2. Prepara tu directorio de trabajo:
   - En `/home/alumno` crea un directorio llamado `p7`. Asumiendo que tu directorio actual es `/home/alumno`, el comando sería:

     ```
     $ mkdir p7
     ```

   - Si ya existe, bórralo y créalo de nuevo:

     ```
     $ rm -rf p7
     $ mkdir p7
     ```

   - Ve al directorio y realiza la práctica desde allí:

     ```
     $ cd p7
     ```

   - Comprueba que estás en el directorio adecuado con:

     ```
     $ pwd
     ```

Debe devolver:

```
/home/alumno/p7
```

3. Asegúrate que la interfaz de red de la máquina virtual tiene una dirección IP del tipo `172.24.21x.xxx`.
4. Debes comprobar que puedes llegar al servidor remoto:

   ```
   $ ping -c 3 <IP-servidor-remoto>
   ```

   Esto debe proporcionar un resultado similar a:

   ```
   3 packets transmitted, 3 received, 0% packet loss, time XXXms
   ```

5. Abre un navegador *dentro de la máquina virtual* y ten a mano estas instrucciones así como el formulario de Campus Virtual. Así te será más fácil seguirlas y descargar los archivos necesarios.

> **Importante**
>
> **No almacenes tu contraseña de Campus Virtual en la máquina virtual.**

2.7.3 UDP

Para el análisis del funcionamiento del protocolo UDP se van a transmitir dos ficheros entre tu equipo y un servidor remoto. El primero será un pequeño fichero de texto y el segundo corresponderá a una imagen en formato BMP. Tendrás que realizar la captura del tráfico con Wireshark que generan ambas transmisiones y responder a las preguntas.

> **Importante**
>
> Antes de comenzar las capturas de esta sección, debemos desactivar el protocolo `WOL` para evitar que Wireshark muestre algunos paquetes UDP como tales, ya que su contenido es similar. Selecciona desde el menú principal `Analizar` y después `Protocolos activados`. Busca `WOL` y desactívalo, si aún no lo está.

udp-me.pcapng

1. Descarga el archivo de texto `me.txt` con:

   ```
   $ wget https://uclm-esi.github.io/redes1-lab/assets/me.txt
   ```

   Comprueba que lo tienes descargado correctamente con:

   ```
   $ ls -l me.txt
   ```

   Deben salir exactamente `41` caracteres (40 y el carácter de nueva línea) como tamaño de archivo.
2. Prepara en un terminal la orden en la línea de comandos (no pulses `Enter` todavía):

   ```
   $ nc -u <IP-servidor-remoto> 5020 < me.txt > me-udp.txt
   ```

3. Ahora ve a Wireshark y comienza a capturar tráfico.
4. Pulsa `Enter` en la línea de comandos con la orden que has preparado.
5. Finaliza la orden tecleando `Ctrl c`.
6. Detén la captura y la guardas con el nombre `udp-me.pcapng`.

7. Responde a las preguntas asociadas a esta captura utilizando como filtro de visualización `ip.addr == <IP-servidor-remoto>`.

udp-bmp.pcapng

1. Descarga la imagen BMP. El fichero se debe llamar `p7barras.bmp`:

   ```
   $ wget https://uclm-esi.github.io/redes1-lab/assets/p7barras.bmp
   ```

2. Prepara en un terminal la orden en la línea de comandos (no pulses `Enter` todavía):

   ```
   $ nc -u <IP-servidor-remoto> 5020 < p7barras.bmp > p7barras-udp.bmp
   ```

3. Ve de nuevo a Wireshark y comienza a capturar tráfico de nuevo.
4. Pulsa `Enter` en la línea de comandos con la orden que has preparado.
5. Finaliza la orden tecleando `Ctrl c`.
6. Detén la captura y la guardas con el nombre `udp-bmp.pcapng`.
7. Responde a las preguntas asociadas a esta captura utilizando como filtro de visualización `ip.addr == <IP-servidor-remoto>`.

2.7.4 TCP

Esta parte es totalmente equivalente a la realizada anteriormente, pero en TCP. La única diferencia es que en las órdenes dadas en el terminal se suprime la opción `-u`, ya que por defecto el protocolo usado es TCP. También vamos a cambiar el número de puerto, que ahora será el `5010`. Además, vamos a usar la opción `-q2` que obliga a terminar la conexión pasados 2 segundos.

> **Importante**
>
> Antes de comenzar las capturas de esta sección, debemos desactivar el protocolo `IPSICTL` para evitar que Wireshark muestre algunos paquetes TCP como tales, ya que su contenido es similar. Selecciona desde el menú principal `Analizar` y después `Protocolos activados`. Busca `IPSICTL` y desactívalo, si aún no lo está.

tcp-me.pcapng

1. Prepara en una terminal la orden en la línea de comandos (no pulses `Enter` todavía):

   ```
   $ nc -q2 <IP-servidor-remoto> 5010 < me.txt > me-tcp.txt
   ```

2. En Wireshark, comienza a capturar.
3. Pulsa `Enter` en la línea de comandos con la orden que has preparado y espera a que finalice el proceso.
4. Detén la captura y la guardas con el nombre `tcp-me.pcapng`.
5. Responde a las preguntas asociadas a esta captura utilizando como filtro de visualización `ip.addr == <IP-servidor-remoto>`.

tcp-bmp.pcapng

1. Prepara en una terminal la orden en la línea de comandos (no pulses `Enter` todavía):

   ```
   $ nc -q2 <IP-servidor-remoto> 5010 < p7barras.bmp > p7barras-tcp.bmp
   ```

2. En Wireshark, comienza a capturar.
3. Pulsa `Enter` en la línea de comandos con la orden que has preparado y espera a que finalice el proceso.
4. Detén la captura y la guardas con el nombre `tcp-bmp.pcapng`.
5. Responde a las preguntas asociadas a esta captura utilizando como filtro de visualización `ip.addr == <IP-servidor-remoto>`.

2.7.5 Finalización

Finalmente, borra el directorio generado `/home/alumno/p7` con:

```
$ rm -rf /home/alumno/p7
```

2.8 Práctica 8 - Capa de red I

El objetivo de esta práctica es conocer el direccionamiento dentro de una red local, utilizarlo para descubrir servicios disponibles en los nodos de la red y aspectos relacionados con la fragmentación. Se utilizará la herramienta `Wireshark` para realizar las capturas correspondientes a las ransmisiones.

2.8.1 Evaluación

Esta práctica es evaluable (**0.5 puntos** sobre los 3.0 puntos totales de las prácticas), por lo que deberás acceder al cuestionario correspondiente en Campus Virtual. El cuestionario se cerrará al acabar la sesión. Cada respuesta errónea *se penalizará con un 1/3* del valor asignado a cada pregunta.

2.8.2 Preparación

Para la realización de la práctica es necesario que la conexión de red esté configurada correctamente:

1. Se suponen conocidos aspectos ya vistos en prácticas anteriores, por ejemplo, el comando para conocer la propia IP y el manejo de Wireshark y Tshark.
2. Prepara tu directorio de trabajo. Como siempre, en `/home/alumno` crea un directorio llamado `p8`. Puedes hacerlo con estos comandos:

   ```
   $ rm -rf p8
   $ mkdir p8
   $ cd p8
   $ pwd
   ```

   Debe devolver:

   ```
   /home/alumno/p8
   ```

2.8.3 Direccionamiento de la red local

La primera parte de la práctica consiste en entender la configuración de direccionamiento proporcionado por el protocolo DHCP. Asegúrate de que la configuración de tu interfaz de red está correctamente configurada para ello.

Dirección de red y dirección de host

1. Usando el comando `ip addr`, obtén tu dirección IP y la máscara de red, así como la interfaz de red que está conectada a Internet.
2. Aplica la máscara sobre la dirección IP para obtener la dirección de red y la dirección de broadcast.
3. Responde a esta parte del cuestionario.

Scan de puertos a hosts locales

Usando el comando `nmap` se puede rastrear por la red local los hosts que tienen puertos accesibles.

> **Nota**
>
> Recuerda que si no tienes `nmap`, puedes instalarlo con `apt`.

Realiza ahora un scan sobre todas las máquinas de tu subred sobre el puerto 22, que es el utilizado por `ssh`, un protocolo de nivel de aplicación que permite abrir terminales de forma remota.

1. Para realizar el scan la sintaxis debe ser la siguiente:

   ```
   $ nmap -p 22 <rango de IPs>
   ```

   El `<rango de IPs>` debe especificarse como la dirección de red *calculada anteriormente* y la máscara en formato CIDR. Por ejemplo:

   ```
   $ nmap -p 22 192.168.1.0/24
   ```

   > **Advertencia**
   >
   > El comando puede tardar varios segundos en ejecutarse. Ten paciencia.

2. Puedes usar la opción `-A` con el mismo comando anterior para averiguar (si es posible) el sistema operativo de los hosts. Por ejemplo:

   ```
   $ nmap -A -p 22 192.168.1.0/24
   ```

3. Responde a esta parte del cuestionario.

ping

Se trata de una de las herramientas más utilizadas en el diagnóstico de red ya que permite verificar si es posible alcanzar, a nivel de red, una máquina remota.

1. Realiza una captura con Wireshark haciendo un ping al servidor DNS de Google:

   ```
   $ ping -c 2 8.8.8.8
   ```

   > **Nota**
   >
   > Este es muy buen comando para utilizar si un día tienes problemas de conexión en casa.

2. Como resultado deberías observar 2 pares de peticiones y respuestas (significado de `-c 2`) hacia el servidor DNS de Google.
3. Responde a esta parte del cuestionario.

2.8.4 Fragmentación

El protocolo IP es también capaz de manejar de forma automática la fragmentación de datagramas en aquellos casos en los que la MTU de las redes por las que debe atravesar el mensaje es menor que el propio tamaño del mensaje.

ping.pcapng

1. Averigua cuál es la MTU de tu red local utilizando el comando `ip addr` y buscando el valor del parámetro del mismo nombre asociado a la interfaz de red que utilizas para conectarte a Internet.
2. Inicia Wireshark y comienza la captura del tráfico hacia Internet.
3. Ejecuta el siguiente comando ping que realiza una única petición y fija un tamaño de payload del mensaje ICMP a 800 bytes:

   ```
   $ ping -c1 -s 800 <IP>
   ```

   Reemplaza `<IP>` por la IP proporcionada durante la sesión.
4. Detén la captura y almacénala como `ping.pcapng`.
5. Filtra por `ip.addr == <IP>` para asegurar que ves el tráfico adecuado.
6. Contesta a las preguntas asociadas.

fragmentacion.pcapng

1. Modifica la MTU de tu red para reducirla a 700 bytes mediante el siguiente comando:

   ```
   $ sudo ip link set <nombre_interfaz> mtu 700
   ```

2. Comienza la captura del tráfico hacia Internet.
3. Repite ahora el mismo ping con las mismas opciones del apartado anterior.
4. Deten la captura y almacénala como `fragmentacion.pcapng`.
5. Contesta a las preguntas asociadas.

2.8.5 Finalización

Finalmente, borra el directorio generado `/home/alumno/p8` con:

```
$ rm -rf /home/alumno/p8
```

2.9 Práctica 9 - Capa de red II

El objetivo de esta práctica es descubrir la jerarquía de redes locales a través del protocolo ICMP. Para ello se utilizará la herramienta `wireshark` para realizar las capturas correspondientes a las transmisiones.

2.9.1 Evaluación

Esta práctica es evaluable (**0.5 puntos** sobre los 3.0 puntos totales de las prácticas), por lo que deberás acceder al cuestionario correspondiente en Campus Virtual. El cuestionario se cerrará al acabar sesión. Cada respuesta errónea se penalizará con un 1/3 del valor asignado a cada pregunta.

2.9.2 Preparación

Para la realización de la práctica es necesario que la conexión de red esté configurada correctamente:

1. Se suponen conocidos aspectos ya vistos en prácticas anteriores, por ejemplo, el comando para conocer la propia IP y el manejo de Wireshark y Tshark.
2. Prepara tu directorio de trabajo. Como siempre, en `/home/alumno` crea un directorio llamado `p9`. Puedes hacerlo con estos comandos:

   ```
   $ rm -rf p9
   $ mkdir p9
   $ cd p9
   $ pwd
   ```

   Debe devolver:

   ```
   /home/alumno/p9
   ```

2.9.3 `traceroute`

> **Nota**
>
> Utiliza `apt` si necesitas instalar `traceroute`.

La utilidad `traceroute` de GNU/Linux proporciona información acerca de la ruta que toman los paquetes desde el host origen hasta alcanzar el host de destino. Para ello, utiliza el campo de la cabecera IP TTL (*Time To Live*), el cual se codifica como un contador de saltos (por defecto con valor `64`) e indica el número máximo de saltos que un paquete puede dar en Internet antes de ser descartado. En cada salto el campo TTL se decrementa en una unidad, de manera que cuando este contador llega a `0` el paquete se descarta y el host que lo descarta informa al host origen enviando un mensaje ICMP con su IP y una estimación del tiempo *Round-Trip delay Time* (RTT).

El funcionamiento de traceroute es el siguiente: envía un paquete `TTL=1` que será descartado en el primer salto. Dicho host devuelve su IP y su RTT. A continuación, envía un segundo paquete IP con `TTL=2` que será descartado en el segundo

salto, el cual devolverá la misma información. El proceso se repite sucesivamente con `TTL=3`, `4`… hasta que el paquete alcanza su destino. Si no se devuelve respuesta de algún salto en un tiempo máximo de 5 segundos (por defecto), se muestra un asterisco `*`. Esto ocurre cuando firewalls u otros dispositivos de seguridad bloquean tráfico.

Pruébalo con:

```
$ traceroute -n 8.8.8.8
```

y obtendrás una salida *similar* a la siguiente:

```
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  172.24.212.2  0.686 ms  0.593 ms  0.683 ms
 2  172.16.160.42  1.463 ms  1.476 ms  1.349 ms
 3  * * *
 4  130.206.212.1  4.242 ms  4.210 ms  4.140 ms
 5  130.206.245.25  4.280 ms  5.958 ms  5.907 ms
 6  * * *
 7  74.125.242.161  4.093 ms 74.125.242.177  5.193 ms 74.125.242.161  4.012 ms
 8  74.125.253.197  5.144 ms 209.85.247.245  4.494 ms 142.250.213.127  4.107 ms
 9  8.8.8.8  4.076 ms  4.350 ms  4.154 ms
```

La primera columna se corresponde con el número de salto, la segunda proporciona el nombre de host y la IP del salto alcanzado, y las tres columnas siguientes se corresponden con tres valores RTT, que es el tiempo de ida y vuelta del paquete desde el host origen hasta ese punto. Hay tres valores porque `traceroute`, por defecto, manda tres datagramas de señalización diferentes. Si alguna de las tres pruebas RTT no devuelve respuesta, se muestra en su tiempo RTT un asterisco `*`. Los `* * *` indican, por tanto, que no se ha recibido ninguna respuesta a las pruebas RTT para ese salto en concreto.

`traceroute` permite, por tanto, determinar el ruta IP que puede seguir un paquete enviado desde su origen hasta su destino, así como el estado de la ruta en cada punto gracias a los valores de tiempo RTT que ofrece.

`traceroute` a `google.es`

1. Arranca `wireshark` e inicia el proceso de captura en la interfaz conectada a Internet.
2. Ejecuta:

   ```
   $ sudo traceroute -n -I -N 1 -q 1 google.es
   ```

   Cuyas opciones son:

   - `-n`: evita que `traceroute` muestre el nombre del host de forma que sólo muestre las direcciones IP.
   - `-I`: fuerza a `traceroute` a utilizar ICMP.
   - `-N`: el número de datagramas de prueba que se envían silmultáneamente (por defecto es 16).
   - `-q`: el número de datagramas de prueba que se envían por salto (por defecto 3).
3. Una vez terminado el comando anterior, para la captura. Deberías observar las peticiones ICMP con sus correspondientes mensajes de error de respuesta. Si quieres, salva la captura como `google-es.pcapng`.
4. Con el filtro de visualización puesto a `icmp`, contesta a las preguntas del formulario asociadas a esta parte.

`traceroute` a `microsoft.es`

1. Realiza el mismo ejercicio que antes, pero ahora a `microsoft.es`. En este caso, ejecuta:

   ```
   $ sudo traceroute -n -I -N 1 -q 1 -m 10 microsoft.es
   ```

   La opción `-m` permite definir el valor máximo del campo TTL que se va a utilizar. En este caso, al llegar a `10` se abortará la ejecución.

2. Contesta a las preguntas del formulario asociadas a esta parte.

2.9.4 `mtr`

`mtr` es una herramienta que funciona sobre los mismos principios que `traceroute` pero está diseñada para medir la calidad de las rutas en cada punto.

1. Instala `mtr` con:

   ```
   $ sudo apt install mtr-tiny
   ```

2. Ejecuta el siguiente ejemplo:

   ```
   $ mtr -n -c 3 -r 8.8.8.8
   ```

   Que hará una prueba hacia `8.8.8.8`, con las siguientes opciones:

   - `-n`: evita que muestre el nombre del host de forma que sólo muestre las direcciones IP.
   - `-c`: limita la prueba a 3 peticiones a cada punto de la ruta.
   - `-r`: muestra la salida al final del proceso en vez de entrar en modo interactivo.

3. La salida de este comando muestra los siguientes valores por cada punto de la ruta:

   - `Loss%`: porcentaje de datagramas perdidos.
   - `Snt`: número de datagramas enviados.
   - `Last`: RTT del último datagrama en ms
   - `Avg`: media del RTT en ms.
   - `Best`: el mejor RTT registrado en ms.
   - `Wrst`: el peor RTT registrado en ms.
   - `StDev`: desviación estándar de los RTT.

   > **Nota**
   >
   > Los datagramas perdidos ya se marcan como `???`.

4. Con la salida obtenida y todo lo aprendido hasta ahora, contesta a las preguntas asociadas.

2.9.5 `ip route`

Con el comando `ip route` se pueden ver y manipular las rutas por las que los datagramas IP serán enviados cuando salen desde tu host.

Por ejemplo, ejecuta:

```
$ ip route
```

Y obtendrás una salida similar a la siguiente:

```
default via 192.168.1.1 dev ens3 proto dhcp metric 600
192.168.1.0/24 dev ens3 proto kernel scope link src 192.168.1.232 metric 600
```

Cada línea representa una ruta configurada en tu host:

- La primera línea muestra la ruta por defecto. Esta ruta es la que tomarán los datagramas IP cuya dirección destino no esté incluída en ninguna de las otras reglas. En este caso, estos datagramas se retransmitirán al host `192.168.1.1` utilizando la interfaz `ens3`. Este host es la *puerta de enlace* o *gateway* y es la ruta que se utiliza para los datagramas que van a Internet.
- La segunda línea será para aquellos datagramas cuya dirección destino está dentro de la red `192.168.1.0/24`, en cuyo caso se enviará por la interfaz `ens3` y será entrega directa (`scope link`), por lo que no hay que reenviarlo a ningún router intermedio. Esta es la ruta que tomarán los paquetes dirigidos a nuestra propia LAN.

Puedes comprobar la ruta que tomará un datagrama con una IP destino que proporciones al comando `ip`. Por ejemplo:

```
$ ip route get 8.8.8.8
```

Mostrará algo similar a:

```
8.8.8.8 via 192.168.1.1 dev ens3 src 192.168.1.232 uid 1000
    cache
```

Indicando que un datagrama IP con destino a `8.8.8.8` se reenviará a `192.168.1.1`.

Rutas

Antes de ejecutar los comandos, **lee todo el ejercicio al completo** ya que te será de gran ayuda para entender mejor las preguntas.

1. Ejecuta:

   ```
   $ ip route
   ```

2. Responde a las preguntas asociadas del cuestionario.

Cambio de rutas

Antes de cada paso, **lee todo el ejercicio al completo**.

1. Vamos a eliminar la ruta al *gateway*. Ejecuta:

   ```
   $ sudo ip route del default
   ```

2. Ahora lo vamos a sustituir por otro:

   ```
   $ sudo ip route add default via 172.24.212.48
   ```

3. Contesta a las preguntas de esta parte del cuestionario.

> **Peligro**
>
> Es posible que experimentes problemas de conexión al formulario en este momento. Si es así y no puedes contestar a las preguntas, puedes anotar tus respuestas, continuar al paso siguiente y contestarlas una vez hayas completado dicho paso.

4. Una vez contestado, vuelve a poner la configuración inicial con el siguiente comando:

   ```
   $ sudo ip route replace default via <IP-original-gateway>
   ```

2.9.6 Finalización

Finalmente, borra el directorio generado `/home/alumno/p9` con:

```
$ rm -rf /home/alumno/p9
```

2.10 Práctica 10 - Capa de enlace

El objetivo de esta práctica es entender el encapsulamiento y las cabeceras usadas en protocolos de diferentes capas, desde la de enlace hasta transporte. Para ello se utilizarán herramientas de generación de tráfico sintético. También se abordará el modo en el que el protocolo Ethernet lleva a cabo la entrega de tramas a través de direcciones MAC, en contrapunto a la entrega *host to host* del protocolo IP.

2.10.1 Evaluación

Esta práctica es evaluable (**0.5 puntos** sobre los 3.0 puntos totales de las prácticas), por lo que deberás acceder al cuestionario correspondiente en Campus Virtual. El cuestionario se cerrará al acabar sesión. Cada respuesta errónea se penalizará con un 1/3 del valor asignado a cada pregunta.

2.10.2 Preparación

Para la realización de la práctica es necesario que la conexión de red esté configurada correctamente:

1. Se suponen conocidos aspectos ya vistos en prácticas anteriores, por ejemplo, el comando para conocer la propia IP y el manejo de Wireshark y Tshark.
2. Como herramienta para generar tráfico sintético utilizaremos `PackETH`, por lo que necesitarás tener esa herramienta instalada.

> **Nota**
>
> Recuerda que para instalar ésta herramienta puedes utilizar `sudo apt install packeth`.

3. Si deseas guardar las capturas, prepara tu directorio de trabajo:
   - Dentro de `/home/alumno` crea un directorio llamado `p10`. Puedes hacerlo con estos comandos:

     ```
     $ rm -rf p10
     $ mkdir p10
     $ cd p10
     $ pwd
     ```

Debe devolver:

```
/home/alumno/p10
```

2.10.3 Recepción de una trama Ethernet broadcast

En este primer ejercicio debes capturar una trama Ethernet que será enviada de manera periódica durante toda la sesión desde el ordenador del profesor.

1. Arranca Wireshark y empieza a realizar una captura en la interfaz de red cableada de tu equipo.
2. Añade un filtro de visualización que muestre únicamente tramas `LLC` que tengan la dirección de *broadcast* (`FF:FF:FF:FF:FF:FF`) como dirección de destino.
3. En unos segundos deberías recibir una trama categorizada como `LLC`.
4. Detén la captura y responde a las preguntas de esta parte del cuestionario.

2.10.4 Envío de una trama Ethernet broadcast

En este ejercicio debes usar el programa PackETH para generar una trama Ethernet sintética para enviar a la red.

> **Nota**
>
> Recuerda que si no tienes PackETH instalado puedes usar `apt` para instalarlo como se menciona más arriba.

1. Averigua la dirección IP y Ethernet de la tarjeta de red de tu equipo con el siguiente comando:

   ```
   $ ip addr
   ```

2. Averigua la dirección IP del enrutador por defecto. Para ello, analiza la salida del siguiente comando:

   ```
   $ ip route
   ```

   > **Nota**
   >
   > Para saber cuál es la IP del enrutador, ten en cuenta que su IP debe estar en la misma subred que la IP de tu interfaz.

3. Lanza el programa PackETH con permisos de administrador:

   ```
   $ sudo packeth
   ```

4. En la sección *link layer* de la interfaz gráfica selecciona lo siguiente:
   - **ver II** (Ethernet version II)
   - *MAC Header*:
     - **Destination**: añade la dirección Ethernet de *broadcast*.
     - **Source**: añade la dirección Ethernet de tu equipo que obtuviste anteriormente.
     - **Ethertype**: selecciona ARP del desplegable y comprueba que el valor es `0806`.
   - **Next layer**: selecciona «Arp packet».

> **Nota**
>
> Recuerda que puedes copiar la dirección MAC en el portapales y pegarla en tantos sitios como necesites.

5. En la sección *Arp payload* selecciona lo siguiente:
   - **Message type**: ARP request (0x0001)
   - **Sender MAC**: de nuevo, la dirección Ethernet de tu equipo (recuerda que puedes copiar y pegar).
   - **Sender IP**: la dirección IP de tu equipo.
   - **Target MAC**: `00:00:00:00:00:00` (es la que queremos averiguar).
   - **Target IP**: la dirección IP del enrutador por defecto que averiguaste anteriormente.
6. Abre Wireshark y comienza una captura en la interfaz de red adecuada.

> **Nota**
>
> Para este ejercicio puedes utilizar un filtro de captura o de visualización para el protocolo ARP con «arp».

7. Una vez iniciada la captura, vuelve a la ventana de PackETH; pincha en el botón `Interface` y selecciona la interfaz adecuada.
8. En la ventana de PackETH, pincha en el botón `Send`.
9. Vuelve a la ventana de Wireshark, observa si se han capturado paquetes ARP. Detén la captura y responde a las preguntas de este apartado del cuestionario.

2.10.5 Envío de una trama Ethernet unicast a un host de la red

De nuevo, utiliza PackETH para generar un nuevo paquete. Esta vez lo que debes conseguir es enviar un datagrama ICMP a la IP `8.8.8.8`. Para poder enviar un datagrama que, a nivel de red, va hacia una red **externa** a la nuestra, deberemos enviar la trama Ethernet a la dirección MAC del enrutador por defecto.

1. Ten a mano la dirección MAC del enrutador por defecto de tu red que has obtenido en el ejercicio anterior en el paquete ARP Reply que hayas capturado.
2. Abre de nuevo PackETH con permisos de administrador:

   ```
   $ sudo packeth
   ```

3. En la sección *link layer* de la interfaz gráfica selecciona lo siguiente:
   - **ver II** (Ethernet version II)
   - *MAC Header*:
     - **Destination**: añade la dirección Ethernet del enrutador por defecto.
     - **Source**: añade la dirección Ethernet de tu equipo.
     - **Ethertype**: pon `0800`, o selecciona IPv4 del desplegable.
   - **Next layer**: selecciona «IPv4».
4. En la sección *IPv4 data* selecciona lo siguiente:
   - **Version**: **0x4**
   - **Header Length**: **0x5**

- **TOS**: **00**.
- **Total length**: Marca la casilla **«Auto»**.
- **Identification**: **0x0001**.
- **Flags**: **2** (elige «Don't fragment» a «Set» y «More fragments» a «Not set»).
- **Fragent offset**: **0**
- **TTL**: **128**
- **Protocol**: **1** (o selecciona **ICMP**) del desplegable.
- **Header cks**: Marca la casilla **«Auto»**.
- **Source IP**: selecciona tu dirección IP.
- **Destination IP**: `8.8.8.8`
- **Next layer**: **ICMP**

5. En la sección *ICMP data*:
   - **Type**: selecciona **Echo request** en el desplegable.
   - **Code**: **00**
   - **Checksum**: selecciona **«Auto»**.
   - **Identifier**: **0x0123**
   - **Seq. number**: **0x0123**
6. Abre Wireshark y comienza una captura en la interfaz de red adecuada.

> **Nota**
>
> Para este ejercicio puedes utilizar un filtro de captura o de visualización para el protocolo ICMP con «icmp»

7. Una vez iniciada la captura, vuelve a la ventana de PackETH; pincha en el botón `Interface` y selecciona la interfaz adecuada.
8. En la ventana de PackETH, pincha en el botón `Send`.
9. Vuelve a la ventana de Wireshark, observa si se han capturado paquetes ICMP. Debes capturar tanto la petición como la respuesta. Después, detén la captura y responde a las preguntas de este apartado del cuestionario.

2.10.6 Finalización

Finalmente, borra el directorio generado `/home/alumno/p10` con:

```
$ rm -rf /home/alumno/p10
```