Enrique Juan Gamboa Alejandro González Sánchez

Práctica 6. Protocolos de transporte UDP y TCP

Contenido

Guion basado en las prácticas de la asignatura de “Redes” de los grados de FDI UCM

6.1 Topología

Configurar la topología mostrada en la figura 6.1 Comprobar que todas las máquinas se conectan entre sí.

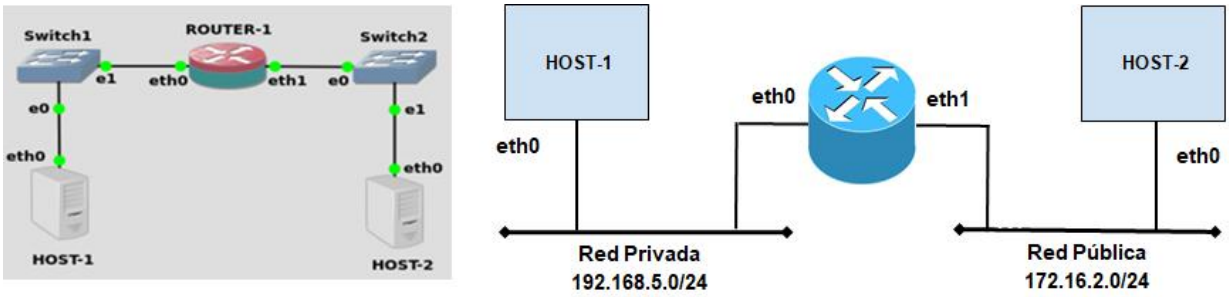


Figura 6.1. Topología de red física y de red lógica

Añade en la siguiente tabla las direcciones IP que has asignado a cada máquina y como has configurado sus tablas de encaminamiento para que los mensajes lleguen de una red a la otra.

Máquina	Dirección IPv4	Sobre la tabla de encaminamiento
HOST-1	`192.168.5.1/24`	Añadido una ruta a la IP del router para llegar a la red 172.16.2.0/24
HOST-2	`172.16.2.1/24`	Añadido una ruta a la IP del router para llegar a la red 192.168.5.0/24
Router	`Red1: 192.168.5.1/24` `Red2: 172.16.2.1/24`	Por defecto funciona para comunicar las máquinas entre sí, pero si no funcionara se haría `sysctl -w net.ipv4.conf.forwarding = 1` para activar el port forwarding del router.

6.2 Puertos bien conocidos

Cuando un proceso cliente solicita una comunicación con un servidor, el cliente debe conocer de antemano el número de puerto del servidor. Habitualmente, cada tipo de servidor utiliza un nº de puerto fijo, denominado **puerto bien conocido** (well-known port), que suele ser el mismo en todos los sistemas.

En cada una de las máquinas, el listado de puertos bien conocidos **se encuentra en el archivo del sistema `/etc/services`.**

Ejercicio 1: Visualiza el contenido del archivo `/etc/services` de una de las máquinas virtuales. Te saldrá información sobre muchos servicios. **Buscar los puertos bien conocidos asociados a los siguientes servicios** y describir brevemente su propósito:

```
# cat /etc/services
```

Servicio	Puerto	Protocolo capa transporte	Para qué son estos servicios
FTP	20 - 21	Tcp	Según esta página, el servicio FTP del puerto 20 es el encargado de la transferencia de datos, y el del puerto 21 es el control.
SSH	22	Tcp	Utilizado para establecer una conexión segura con otra máquina para mandar comando de Shell.
SMTP	25	tcp	Para e-mail (Simple Mail Transfer Protocol)
POP3	110	Tcp	Usado por los clientes de e-mail para acceder a una bandeja de entrada en un servidor.
HTTP (www)	80	Tcp	Internet
HTTPS	443	Tcp + udp	Con tcp, el protocolo de http a través de TLS/SSL; con udp acceso a http/3

6.3 La herramienta `netcat`

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

Cómo crear un Servidor con `netcat`

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

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

Las opciones de esta orden son:

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

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

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

La opción **`-u`** indica a que se tiene que abrir un puerto UDP

- Para salir del programa servidor: **`Ctrl+C`**
  - Cuando se sale del programa servidor, **NO se cierra el programa cliente**, será necesario finalizar también del cliente mediante `Ctrl+C`

Cómo crear un Cliente con `netcat`

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

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

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

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

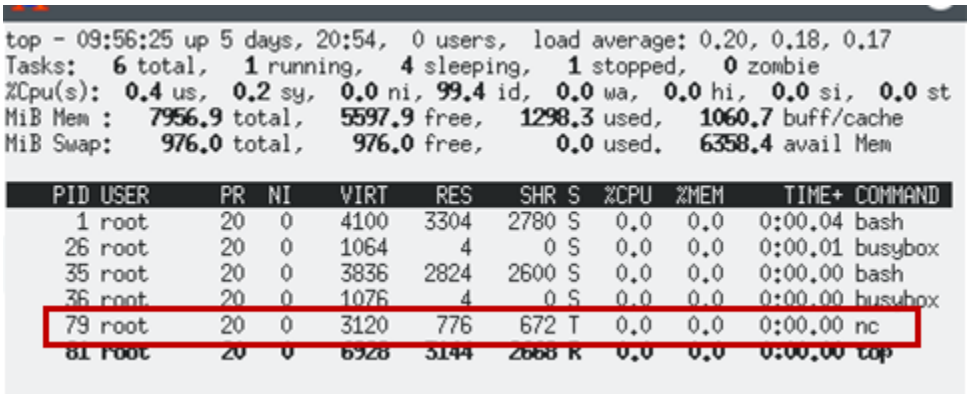
- Para salir del programa cliente: **`Ctrl+C`**
  - Cuando se sale del programa cliente **el programa servidor también finaliza su ejecución**

Una vez creados un par cliente- servidor, **cualquier cadena de texto que se introduzca en el lado del cliente, se envía al servidor y viceversa.**

IMPORTANTE:
Cuando se crea un cliente o un servidor con el comando **`netcat`** el **terminal** de la máquina donde se ha creado **está bloqueado**, porque el proceso que acabamos de abrir (crear un cliente o un servidor) se está ejecutando en primer plano. Esto **nos impide ejecutar otros comandos en la terminal**.

Para poder ejecutar otros comandos sin tener que cerrar el proceso seguimos los siguientes pasos:

1. **Suspendemos el proceso `netcat`** pulsando **`CTRL+Z`** (**esto nos devuelve el control del terminal**).
2. **Ejecutamos los comandos que queramos**
3. **Devolvemos** el proceso **`netcat`** **a primer plano**. Para ello:
   a. Usamos el comando **`top`** para mostrar todos los procesos que se están ejecutando
   b. **Buscamos el identificador (PID) del proceso `netcat (nc).`** Por Ejm en la siguiente figura el PID del proceso nc sería 79



c. **Salimos del comando `top`** pulsando **`CTRL+C`**

d. Ejecutamos el comando: **`fg % <PID del proceso nc>.`** En este ejemplo sería:

```
# fg % 79
```

6.4 Comando `ss` (socket statistics)

El comando **`ss`** en Linux se usa para **mostrar información detallada sobre los sockets y puertos abiertos.** Es una alternativa más rápida y moderna al comando netstat.

NOTA: un socket abierto y un puerto abierto están relacionados.

- Un **puerto abierto** indica que un servicio está escuchando, pero no necesariamente hay una conexión activa solo está listo para recibirlas.
- Un **socket abierto** es una **conexión activa que usa un puerto**. Un socket usa un puerto para la comunicación, pero también involucra direcciones IP y protocolos.

Las opciones del comando **`ss`** son:

-t (TCP): muestra solo conexiones TCP
-u (UDP): muestra solo conexiones UDP
-a (all): muestra tanto conexiones activas como en escucha.
-l (listening): muestra solo puertos en estado de escucha.
-n muestra el número de puerto en lugar del nombre del servicio
-p (processes): muestra el proceso que está usando cada conexión.

Si en una máquina queremos ver todas las conexiones TCP y UDP, tanto activas como en escucha, y que muestren el número de puerto **¿Qué opciones tienen que acompañar al comando `ss` para obtener lo que queremos?**

```
ss -a -n
```

NOTA:
Las máquinas del simulador GNS3, por defecto, no tienen ningún puerto TCP o UDP abierto, por tanto, la salida del comando `ss` se mostrará vacía.

Cuando realicéis los ejercicios, al crear conexiones cliente-servidor, se abrirán puertos y se crearán sockets, por lo tanto, al ejecutar el comando anterior, en **el terminal de la máquina se mostrará algo de este tipo:**

```
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:Port
tcp   ESTAB   0      0      192.168.1.100:443      192.168.1.50:53210
tcp   LISTEN  0      128    0.0.0.0:22             0.0.0.0:*
tcp   TIME-WAIT  0    0     192.168.1.100:80       192.168.1.51:60532
udp   UNCONN  0      0      0.0.0.0:68             0.0.0.0:*
```

La siguiente tabla describe cada uno de los campos

Campo	Descripción
Netid	Tipo de conexión (`tcp` , `udp` , etc.).
State	Estado de la conexión (`LISTEN` , `ESTAB` , `TIME-WAIT` , `UNCONN` , etc.).
Recv-Q	Cantidad de datos en la cola de recepción.
Send-Q	Cantidad de datos en la cola de envío.
Local Address:Port	Dirección IP y puerto de la máquina local.
Peer Address:Port	Dirección IP y puerto del host remoto (si aplica).

Interpretando la información del ejemplo anterior

Línea 1 de ejemplo:

```
tcp   ESTAB   0      0      192.168.1.100:443      192.168.1.50:53210
```

Indica que hay una conexión **TCP establecida (activa)** en el puerto 443 (HTTPS) de la máquina 192.168.1.100, con un cliente en 192.168.1.50 usando el puerto 53210.

Línea 2 de ejemplo:

```
tcp    LISTEN  0       128     0.0.0.0:22           0.0.0.0:*
```

Indica que:

- Un servicio está en **escucha en el puerto 22**
- La dirección IP y el puerto local en el que el servicio está escuchando es: **`0.0.0.0:22`**
  - 0.0.0.0 significa que escucha en todas las interfaces de red disponibles
  - 22 es el puerto, que en este caso corresponde al servicio SSH

- La dirección IP y el puerto del host remoto: **`0.0.0.0:*`**
  - Indica que cualquier dirección IP con cualquier puerto puede conectarse al puerto 22, sin restricciones.

Posibles estados de una conexión:

Las conexiones se pueden encontrar en diversos estados, los estados más comunes son los siguientes:

- **`UNCONN`** (*unconnected* o desconectado): significa que el servidor no está conectado a ningún cliente
- **`LISTEN`:** significa que el puerto está abierto (en escucha), a la espera de recibir conexiones de clientes.
- **`ESTABLISHED`:** significa que el puerto TCP de la máquina local (identificado en la columna `Local Address:Port`) ha establecido una conexión con una entidad TCP remota (identificada en la columna `Peer Address:Port`). Este estado permite el intercambio de datos entre ambos extremos.
- **`TIME_WAIT`:** significa que la aplicación TCP local ha cerrado la conexión, y ésta se mantiene en estado `TIME_WAIT` durante un cierto periodo de tiempo (60 s) antes de liberarse.
- **FIN-WAIT-1:** Se ha enviado una solicitud para cerrar la conexión.
- **FIN-WAIT-2:** La solicitud de cierre fue reconocida, pero el otro extremo aún no ha cerrado.

6.5 El protocolo TCP

En este apartado vamos a ver lo estudiado en el tema 4 sobre el protocolo TCP: cómo funciona el protocolo TCP e identifica los campos de la cabecera de los segmentos TCP.

6.5.1 Establecimiento de conexión, transmisión de datos y cierre de conexión TCP

Ejercicio 2: Crear un cliente-servidor tipo TPC

Supongamos que el cliente se ejecuta en Host-1 y el servidor en Host-2 y que el puerto servidor es el número 60000.

- **Arranca Wireshark** para analizar el tráfico de red (puedes hacerlo en cualquiera de los enlaces)
- **Arranca** con `netcat` **un servidor TCP** en el HOST-2 con número puerto 60000
- Comprueba el estado de la conexión con el comando ss

Protocol	State	Local Address	Peer Address
Tcp	LISTEN	0.0.0.0:60000	0.0.0.0:*

¿Qué significa esa información?

Que el servidor está a la escucha de una conexión con un cliente.

- **Arranca** con `netcat` el correspondiente **cliente TCP** en el HOST-1, para que se conecte al puerto 60000 servidor

- Comprueba el estado de la conexión en el cliente con el comando ss

Protocol	State	Local Address	Peer Address
Tcp	ESTAB	192.168.5.1:34160	172.16.2.1:60000

¿Qué significa esa información?

Que el cliente ha establecido conexión con el servidor

- Comprueba de nuevo el estado de la conexión en el servidor.

Protocol	State	Local Address	Peer Address
Tcp	ESTAB	172.16.2.1:60000	192.168.5.1:34160

- ¿Qué significa esa información?

Que el servidor ha establecido conexión con el cliente, especificando una dirección IP para conectarse con él.

- A continuación, cualquier cadena de texto que se introduzca en el lado del cliente, se envía al servidor y viceversa. **Intercambia algunos mensajes de texto entre Host-1 y Host-2 y viceversa**

Ejercicio 3: Continuando con el ejercicio anterior.

- Como estaba abierto el Wireshark, analiza los segmentos TCP que han intercambiado el cliente y el servidor. ¿Cuáles de ellos se corresponden a la fase de establecimiento de conexión?

Cabecera Red (IP)		Cabecera Transporte (TCP)					Datos
IP Origen	IP Destino	Puerto origen	Puerto Destino	Flags	Numero Seq.	Número ACK	
Host1	Host2	34156	60000	SYN	0	0	(establecimiento)
Host2	Host1	60000	34156	SYN, ACK	0	1	(establecimiento)

Host1	Host2	34156	60000	ACK	1	1	(establecimiento)
Host1	Host2	34156	60000	PSH, ACK	1	1	Hola
Host2	Host1	60000	34156	ACK	1	6	-
Host1	Host2	34156	60000	FIN, ACK	6	1	-
Host2	Host1	60000	34156	FIN, ACK	1	7	-
Host1	Host2	34156	60000	ACK	7	2	-

NOTA: para ver el contenido de los datos enviados mirar en la ventana del wireshark que muestra el contenido del mensaje en hexadecimal

- **Cierra la conexión desde la máquina cliente** (HOST-1) con Ctrl+C
- Comprueba el estado de la conexión en el servidor y el cliente con el comando ss

Cliente

Protocol	State	Local Address	Peer Address
Tcp	FIN-WAIT	192.168.5.1:34160	172.16.2.1:60000

Servidor

Protocol	State	Local Address	Peer Address
Tcp	CLOSE-WAIT	172.16.2.1:60000	192.168.5.1:34160

- ¿Qué significa esa información?

Esta información muestra que tanto el servidor como el cliente está esperando a un mensaje de confirmación de cierre, y que el servidor aún no lo ha mandado. Esto se debe a que el netcat de nuestro servidor estaba en segundo plano: en el momento que hacemos la misma operación con ambos netcat presentes en consola, las conexiones desaparecen en el panel que devuelve el comando ss en ambas máquinas.

- Comprueba que el servidor ha cerrado la conexión., si no es así ciérrala.
- Analiza, mediante el Wireshark, los mensajes intercambiados entre cliente y servidor en el proceso de cierro

Cabecera Red (IP)		Cabecera Transporte (TCP)					Datos
IP Origen	IP Destino	Puerto origen	Puerto Destino	Flags	Numero Seq.	Número ACK	

Host1	Host2	34156	60000	FIN, ACK	6	1	-
Host2	Host1	60000	34156	FIN, ACK	1	7	-
Host1	Host2	34156	60000	ACK	7	2	-

6.5.2 Intento de conexión a un puerto TCP cerrado

En este apartado vamos a observar que ocurre cuando un cliente TCP intenta establecer una conexión con un servidor TCP que tiene el puerto cerrado.

Ejercicio 4

- **Arranca Wireshark** para analizar el tráfico de red (puedes hacerlo en cualquiera de los enlaces)
- Arranca un cliente TCP en el HOST-1 que intente comunicarse a un puerto servidor cerrado
- Analiza, mediante el Wireshark, los mensajes intercambiados entre cliente y servidor

Cabecera Red (IP)		**Cabecera Transporte (TCP)**					**Datos**
IP Origen	**IP Destino**	**Puerto origen**	**Puerto Destino**	**Flags**	**Numero Seq.**	**Número ACK**	
Host1	Host2	34234	60000	SYN	0	0	-
Host2	Host1	60000	34234	RST, ACK	1	1	-

6.5.3 Opciones TCP

Este apartado es opcional

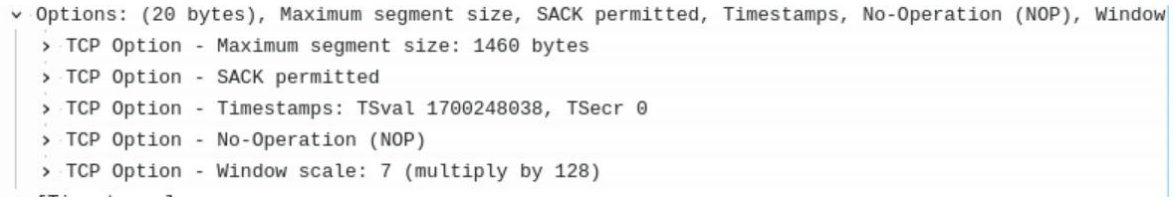
Durante el establecimiento de conexión TCP permite negociar algunas opciones:

- Opción tamaño máximo del segmento (MSS, *Maximum Segment Size*)
  - Permite negociar el tamaño máximo del bloque de datos que contienen los segmentos que se envían al destinatario
  - Normalmente se calcula a partir de la MTU de la red.
    - Ejemplo:
      - En Ethernet la MTU es de 1500 bytes
      - A este valor se le resta el tamaño típico de la cabecera IP (20 bytes) y el tamaño típico de la cabecera TCP (20 bytes)
      - El valor resultante es un MSS de 1460 bytes
- Opción factor de escala de ventana (*Window scale*)
  - Permite trabajar con ventanas mayores de 2^{16} bytes

    - Se puede activar/desactivar mediante el siguiente parámetro del kernel:
      `net.ipv4.tcp_window_scaling`
- Opción sello de tiempo (*Timestamps*)
    - Permite introducir la hora de envío de un segmento
    - Se puede activar/desactivar mediante el siguiente parámetro del kernel:
      `net.ipv4.tcp_timestamps`
- Opción de confirmación selectiva (SACK, *Selective ACK*)
    - Permite confirmar segmentos fuera de orden
    - Se puede activar/desactivar mediante el siguiente parámetro del kernel:
      `net.ipv4.tcp_sack`
- Opción de No Operación (NOP)
    - Se utiliza únicamente para relleno y alineamiento de las opciones dentro de la cabecera TCP, pero no tiene ninguna función.
        - Las opciones deben ocupar un nº entero de palabras de 32 bits. En caso de que ocupen menos, se rellena con la opción NOP hasta llegar a los 32 bits.

Ejercicio 5: OPCIONAL

- Usando el comando `sysctl` comprueba el valor de los tres siguientes parámetros del kernel (por defecto los tres están activados)
    - `net.ipv4.tcp_window_scaling`
    - `net.ipv4.tcp_timestamps`
    - `net.ipv4.tcp_sack`
- Arranca Wireshark para analizar el tráfico de red (puedes hacerlo en cualquiera de los enlaces)
- Crea una pareja de aplicaciones cliente-servidor TCP con `netcat` en las máquinas HOST-1 y HOST-2 usando el puerto servidor número 60000.
- Analiza, mediante Wireshark, el primer paquete de establecimiento de conexión TCP (SYN) y comprueba todas las opciones que se negocian. Deben ser las siguientes:



- Cierra la conexión `netcat` (mediante CTRL+C)
- Desactiva las opciones de "*Window scale*", "*Timestamps*" y "*SACK*" en las máquinas HOST-1 y HOST-2, poniendo a cero los siguientes parámetros del Kernel mediante el comando `sysctl`:
    - `sysctl -w net.ipv4.tcp_window_scaling=0`
    - `sysctl -w net.ipv4.tcp_timestamps=0`
    - `sysctl -w net.ipv4.tcp_sack=0`
- Crea de nuevo una pareja de aplicaciones cliente-servidor TCP con `netcat` en las máquinas HOST-1 y HOST-2 usando el puerto servidor número 60000.
- Analiza de nuevo, mediante Wireshark, el primer paquete de establecimiento de conexión TCP (SYN) y comprueba todas las opciones que se negocian. Deben ser las siguientes:

- Cierra la conexión `netcat` (mediante CTRL+C)

6.6 El protocolo UDP

En este apartado vamos a ver lo estudiado en el tema 4 sobre el protocolo UDP: cómo funciona el protocolo P e identifica los campos de la cabecera de los segmentos TCP.

6.6.1 Transmisión de datos UDP

Ejercicio 6: Crear un cliente-servidor tipo UDP

Supongamos que el cliente se ejecuta en Host-2 y el servidor en Host-1 y que el puerto servidor es el número 20000.

- **Arranca Wireshark** para analizar el tráfico de red (puedes hacerlo en cualquiera de los enlaces)
- **Arranca** con `netcat` **un servidor UDP** en el HOST-1 con número puerto 20000
- Comprueba el estado de la conexión con el comando ss

Protocol	State	Local Address	Peer Address
Udp	UNCONN	0.0.0.0:20000	0.0.0.0:*

¿Qué significa esa información?

Que está abierto el servidor pero, como el protocolo es UDP, el estado se mantiene en UNCONN ya que no escucha.

- **Arranca** con `netcat` el correspondiente **cliente UDP** en el HOST-2, para que se conecte al puerto 20000 servidor
- Comprueba el estado de la conexión en el cliente con el comando ss

Protocol	State	Local Address	Peer Address
Udp	ESTAB	172.16.2.1:41888	192.168.5.1:20000

¿Qué significa esa información?

Que se ha establecido la conexión entre el cliente (HOST2) y el servidor (HOST1)

- Comprueba de nuevo el estado de la conexión en el servidor. Añade solo la información nueva

Protocol	State	Local Address	Peer Address
Udp	UNCONN	0.0.0.0:20000	0.0.0.0:*

- ¿Qué significa esa información?

Que el servidor no sabe que se ha conectado el cliente debido a que no hay saludo entre cliente-servidor (SYN), y aún no se ha mandado ningún mensaje

- A continuación, cualquier cadena de texto que se introduzca en el lado del cliente, se envía al servidor y viceversa. **Intercambia algunos mensajes de texto entre Host-1 y Host-2 y viceversa**

- **Cierra la conexión desde la máquina cliente** (HOST-2) con Ctrl+C
- Comprueba el estado de la conexión en el servidor y el cliente con el comando ss

Cliente

Protocol	State	Local Address	Peer Address
-	-	-	-

Servidor

Protocol	State	Local Address	Peer Address
Udp	ESTAB	192.168.5.1:20000	172.16.2.1:41888

- ¿Qué significa esa información?

El cliente no tiene información ya que hemos cerrado la conexión desde él, pero el servidor al haber recibido mensaje del cliente se queda con el último estado que tenía (ESTAB). Esto se debe a que el cliente no manda mensaje de que ha sido cerrado.

- Comprueba que el servidor ha cerrado la conexión., si no es así ciérrala.

- Como estaba abierto el Wireshark, **analiza los segmentos UDP que han intercambiado** el cliente y el servidor. ¿Qué diferencia ves con el caso TCP?

Que no hay ACK, FIN, SYN, etc.: sólo aparece la cadena de texto que mandamos entre ellos

6.6.2 Intento de comunicación con un puerto UDP cerrado

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

Ejercicio 7

- **Arranca Wireshark** para analizar el tráfico de red (puedes hacerlo en cualquiera de los enlaces)

- Arranca un cliente UDP en el HOST-2 que intente comunicarse a un puerto servidor cerrado

- Analiza, mediante el Wireshark, que tipo de mensaje se ha enviado y quién lo envía

Hasta que no hemos intentado mandar una cadena de texto al servidor, no ha aparecido nada en Wireshark. En el momento que lo enviamos, aparecieron dos elementos: el mensaje UDP del cliente al servidor, y un mensaje en respuesta de protocolo ICMP que indicaba que el destino era inalcanzable debido a que el puerto es inalcanzable.