Práctica 1: Wireshark y otras herramientas de red

Los analizadores de protocolos o de red, también conocidos como "*sniffers*", son herramientas de gran ayuda para los administradores de las redes de computadores, ya que permiten el análisis detallado de muchos factores del comportamiento de las mismas. Estas aplicaciones permiten capturar una copia de los paquetes que circulan por la red para su análisis posterior. Los analizadores más avanzados incluyen una interfaz gráfica capaz de mostrar los campos de los protocolos de comunicación de los distintos niveles y obtener estadísticas de utilización. El análisis posterior de los datos capturados facilita considerablemente la detección de problemas, así como la depuración del software de red durante su fase de elaboración. Por ejemplo, un administrador de red que detecte que las prestaciones de la red son bajas puede utilizar uno de estos analizadores para detectar qué segmentos de la red, protocolos y máquinas están generando más tráfico, y de esa forma llevar a cabo las acciones necesarias, o bien verificar el correcto funcionamiento de los diferentes dispositivos de red (*hosts*, servidores, *routers*, cortafuegos, NAT, etc).

Desde el punto de vista docente, los analizadores de protocolos permiten ver de forma práctica los protocolos de comunicación ya presentados en las clases de teoría, así como las relaciones entre los protocolos de distinto nivel. Por todo ello, intentaremos familiarizaros con el uso de una de estas herramientas.

Tras acabar esta práctica serás capaz de capturar paquetes usando la herramienta *Wireshark,* y analizar los protocolos de la pila TCP/IP que utilizan en las distintas capas los paquetes capturados.

Dado que por la red viajan multitud de paquetes, será necesario seleccionar aquellos que nos resulten de interés. Por ello vamos a aprender a utilizar los filtros que nos proporciona *Wireshark* para capturar paquetes. De manera que aceptaremos unos paquetes y desecharemos otros. También comenzarás a interpretar el contenido de los paquetes capturados, para afianzar los conceptos relativos a los protocolos estudiados en clase. Todo ello te permitirá poner en práctica los conocimientos adquiridos en las sesiones de aula, adquiriendo una mayor comprensión de los procesos que ocurren en la red cuando se llevan a cabo diversas acciones a nivel de usuario.

Lectura Previa: Leer los apartados 1, 2 y 3 del boletín antes de realizar la sesión práctica.

1. El analizador de protocolos: *Wireshark*

A la hora de elegir un analizador de protocolos nos encontramos con una abundante oferta, tanto de productos comerciales como de software de libre distribución. Uno de los más populares, y el elegido para algunas prácticas de la asignatura, es *Wireshark.* Se trata de un producto gratuito y muy versátil, que puede descargarse desde http://www.wireshark.org. Está disponible para sistemas Windows, Unix y MacOSX. Permite no sólo capturar tráfico de la red, sino también filtrarlo y

analizarlo. Además, permite leer ficheros de datos recogidos con otros analizadores de protocolos como *tcpdump* o *NetXRay*, con lo que pueden aprovecharse otras capturas previamente realizadas.

Por seguridad, los analizadores de protocolos requieren permisos de administrador del sistema para poder realizar capturas del tráfico de la red.

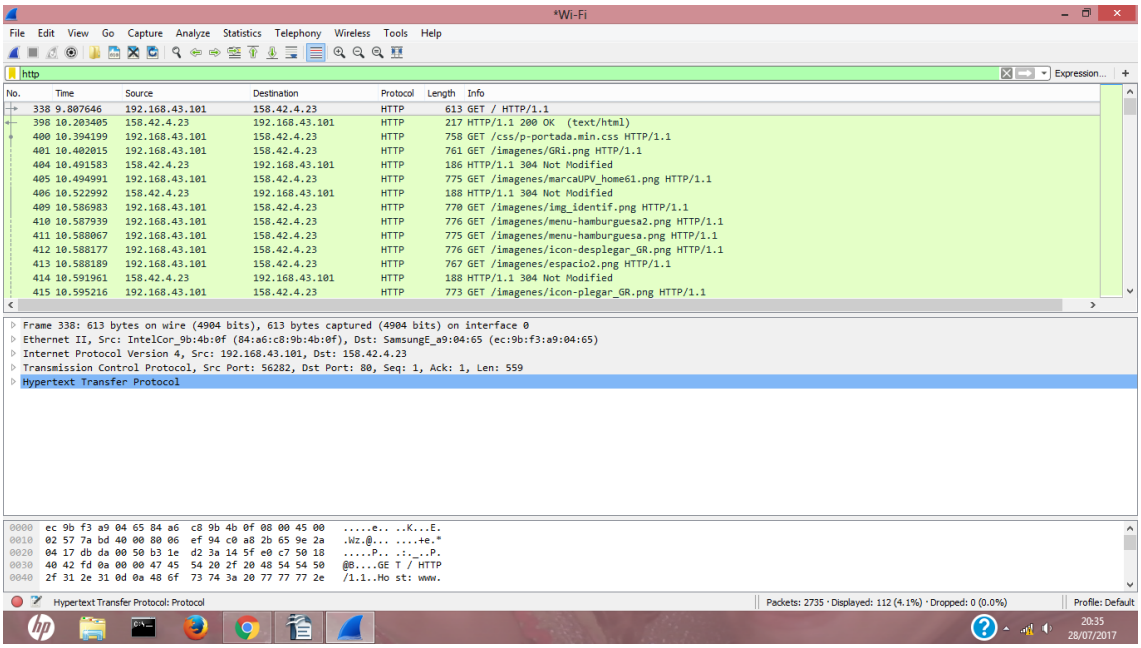


Figura 1. Descripción de Wireshark

Comenzaremos comentando el aspecto habitual del programa, que se muestra en la figura 1. La ventana de *Wireshark* se encuentra dividida horizontalmente en **tres ventanas o zonas principales**.

1) La parte superior (con fondo verde) contiene la lista de los paquetes capturados. Muestra una breve descripción de cada uno de los paquetes. Pulsando en cualquiera de los paquetes de esta lista podemos ver en las dos ventanas inferiores sus detalles.

2) La parte intermedia (de fondo azulado) muestra con mayor detalle el paquete seleccionado en la ventana superior. Indica los diferentes protocolos de la pila TCP/IP que se utilizan en cada nivel de este paquete, así como los valores de cada uno de los campos de esos protocolos.

3) Por último, la parte inferior (de fondo blanco) muestra el contenido del paquete seleccionado en la ventana superior, en notación hexadecimal y en ASCII, y marca en negro los datos seleccionados en la ventana intermedia.

Además de estas tres partes o ventanas, la ventana principal de *Wireshark* permite (en la barra verde de la parte superior) establecer un filtro en pantalla de los paquetes

capturados en función del tipo de paquetes y/o contenido de sus campos. Este filtrado *a posteriori* es complementario al filtrado de paquetes en el momento de la captura, tal y como se explicará más adelante.

2. Cómo capturar paquetes

Realizar una captura de paquetes es sencillo. Primero accederemos al menú *Capture* y allí seleccionaremos la opción *Options*. Esto nos lleva a una nueva ventana similar a la mostrada en la figura 2, llamada *Capture Interfaces*. En esta ventana podremos especificar los diversos parámetros relacionados con la captura.

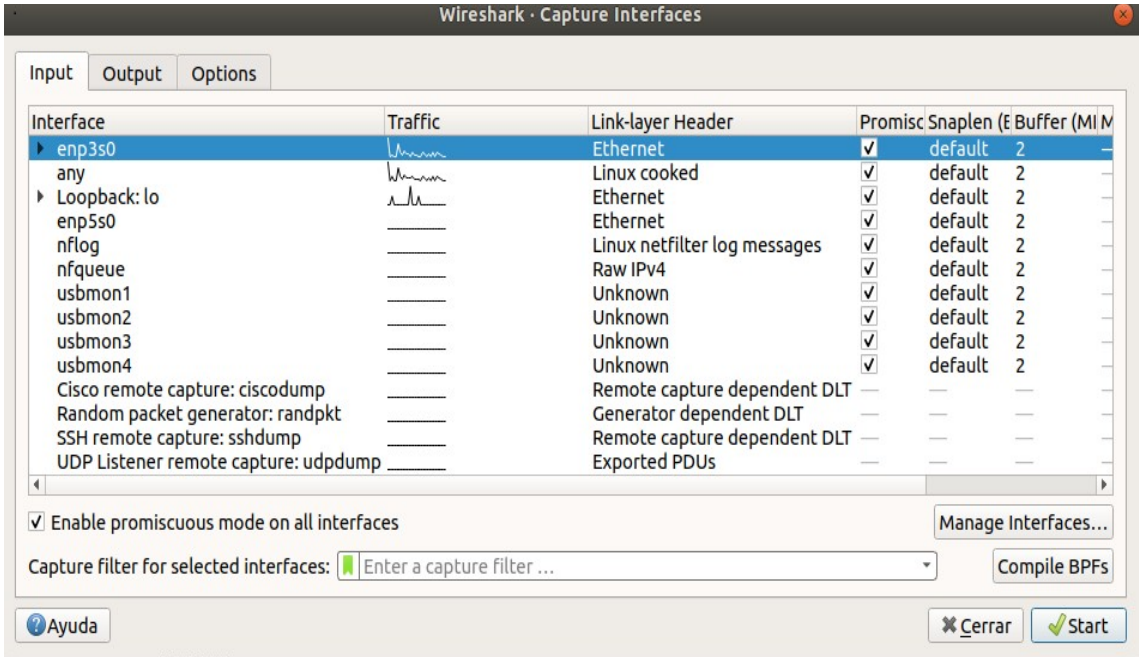


Figura 2. Ventana de Captura de paquetes: Pestaña Input

En primer lugar, seleccionamos la pestaña *Input*. El primer parámetro que podemos especificar es la interfaz de red. Esta opción sólo tiene sentido si disponemos de diferentes interfaces: interfaces inalámbricas, distintas tarjetas de red (posiblemente conectadas a diferentes redes), como es el caso de los computadores del laboratorio (o la máquina virtual equivalente). En este caso, la interfaz a seleccionar es la *enp0s3*.

Otra opción que podemos detallar es si deseamos realizar una captura en modo promiscuo. En este modo el programa capturará cualquier paquete que sea visible a la tarjeta de red, independientemente de si está destinado a ella o no. Por el contrario, podemos seleccionar la captura de únicamente aquellos paquetes que van destinados a, o que provienen de, nuestra tarjeta de red.

En esta pestaña también podemos introducir un **filtro de captura**, con el fin de especificar qué paquetes deseamos capturar, y procesar después más fácilmente la información obtenida. El uso de este tipo de filtros lo describiremos más adelante.

Otra posibilidad que nos ofrece esta ventana es la de volcar los paquetes capturados a un fichero. Para ello deberemos seleccionar la pestaña *Output* (Figura 3). Esto puede resultar interesante para guardar un registro del tráfico capturado. No obstante, dado que también podemos almacenar las capturas después de realizarlas, desde el menú *File/Save*, no vamos a hacer uso de esta posibilidad por el momento.

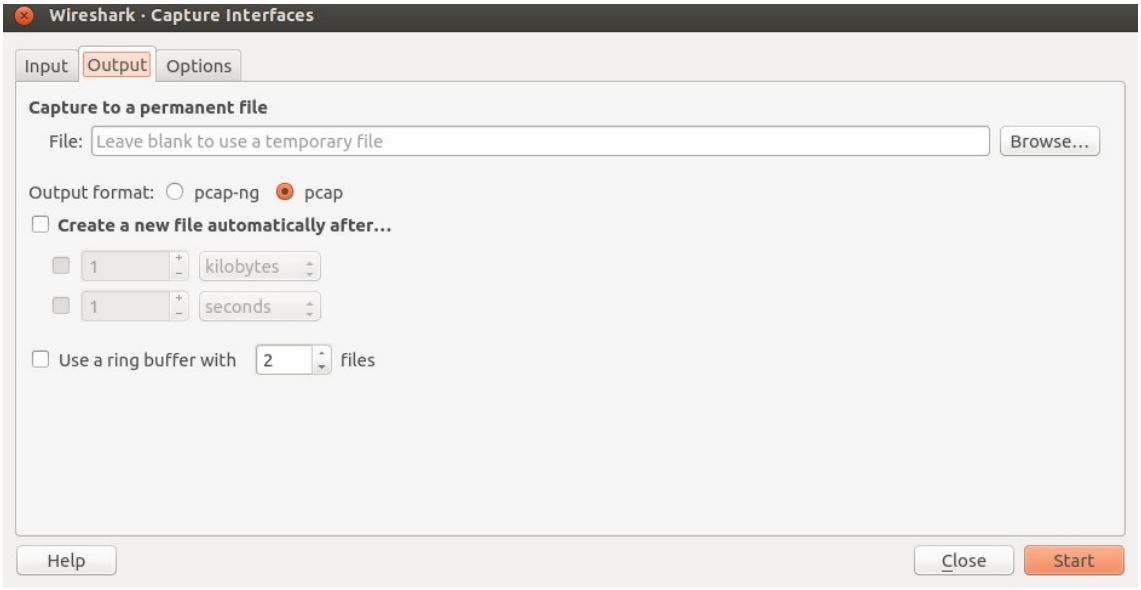


Figura 3. Ventana de Captura de paquetes:Pestaña Output

Otras opciones que esta ventana nos permite detallar están relacionadas con la visualización en pantalla de la captura. Seleccionando la pestaña *Options* (Figura 4), podemos optar por ver en tiempo real los paquetes que se van capturando y también podemos elegir que se realice un desplazamiento vertical automático de la pantalla (*scrolling*).

Finalmente, con el fin de terminar la captura, podemos indicar que ésta termine automáticamente cuando se haya capturado cierto número de paquetes, o se haya capturado una cantidad determinada de Kbytes, o bien cuando haya transcurrido cierta cantidad de tiempo. En caso de no seleccionar ninguna de las opciones, tendremos que finalizar la captura de forma manual. Para iniciar la captura presionaremos el botón *Start (*pestaña Input).

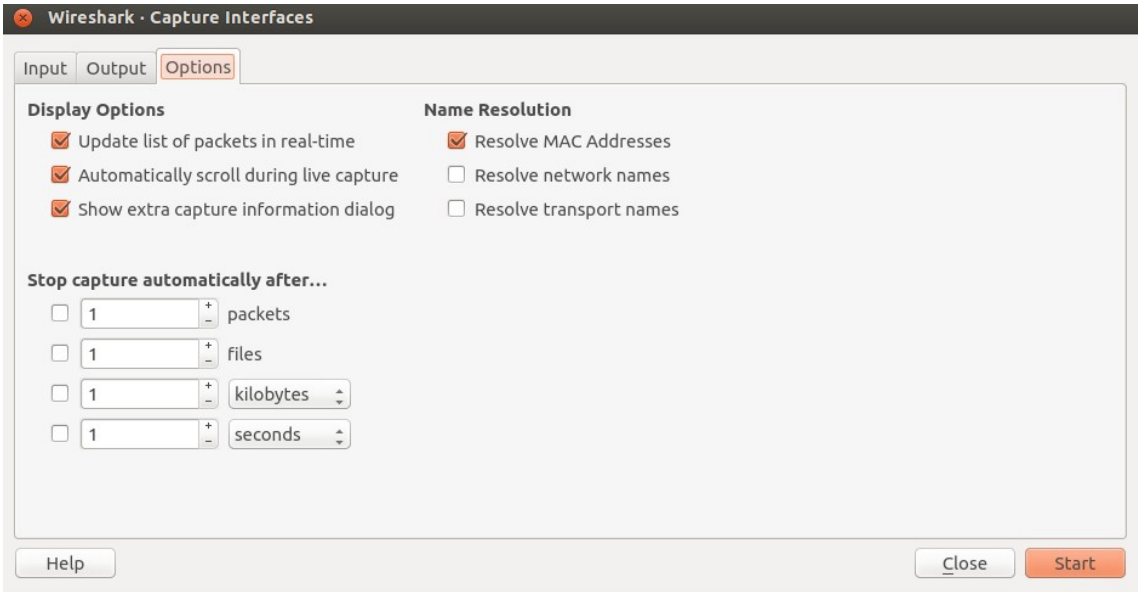


Figura 4. Ventana de Captura de paquetes:Pestaña Options

3. Filtros de captura y de pantalla

Al intentar analizar el tráfico de cualquier red, en particular el de la red de la UPV, resulta habitual encontrarse con gran cantidad de paquetes que emplean protocolos en los que no estamos interesados. Tal cantidad de tráfico dificulta el análisis de los paquetes capturados y aumenta innecesariamente el tamaño de los ficheros de captura, por lo que se hace indispensable filtrar toda esa información.

Para filtrar los paquetes y facilitar el análisis del tráfico capturado podemos usar dos alternativas, no excluyentes entre ellas. La primera es definir un **filtro de captura**, de modo que el propio *Wireshark*, cuando llega un nuevo paquete, decide si ese paquete se ajusta o no a los criterios establecidos por el filtro. En caso de que se ajuste, el paquete es aceptado y mostrado en pantalla, mientras que en caso contrario el paquete se descarta.

La otra alternativa para filtrar la información de los paquetes es establecer un **filtro de pantalla (o visualización)**. En este caso lo habitual es capturar todos los paquetes que circulan por la red, sin restricción alguna, y especificar un filtro para poder extraer entre todo ese tráfico capturado aquellos paquetes que nos interesan. En cualquier caso, es posible usar un filtro de captura y posteriormente, sobre los paquetes capturados, usar un filtro de pantalla para ver mejor los detalles que estemos buscando en cada momento.

Los filtros de captura se pueden especificar desde la ventana de captura (*Capture/ Input,* ver la parte inferior de la Figura 2). Podemos especificar un filtro escribiendo la expresión correspondiente en la caja de texto situada junto al botón *Filter*. La sintaxis de estas expresiones es la misma que la usada en la orden **tcpdump**, disponible

habitualmente en los sistemas Unix y Linux. En el apartado siguiente se mostrará un resumen de esta sintaxis.

Por otra parte, los filtros de pantalla se pueden especificar desde la parte superior de la ventana principal de *Wireshark* (ver en la Figura 1 el filtro http de visualización aplicado). La sintaxis para este tipo de filtros es ligeramente diferente. Es posible disponer de un asistente para especificar estos filtros.

Es conveniente tener cuidado con estos filtros de visualización, ya que pueden, en ocasiones, llevar a error. Por ejemplo, estos filtros se emplearán más adelante en TCP de forma automática para seguir un flujo TCP. Por tanto, para volver a visualizar todos los paquetes capturados es necesario limpiar este filtro de visualización, bien con la opción correspondiente (*clear filter*), o bien empleando un filtro vacío (línea en blanco).

3.1 Expresiones de filtros de captura

Para seleccionar qué paquetes serán capturados se emplea una expresión de filtro, de forma que el paquete será almacenado si cumple con los criterios del filtro (la expresión se evalúa a ***true***). Las expresiones de filtro se emplean siempre en minúsculas y consisten en una o varias primitivas conectadas mediante operadores lógicos (and, or, not). Cada una de estas primitivas consta de un identificador precedido, al menos, de alguno de los siguientes tres tipos de calificadores:

- De tipo: Indican a qué hace referencia el identificador

    1. **Computadores concretos (*host*)**. Por ejemplo, el filtro:

        ```
        host 158.42.180.62
        ```

        captura todas las tramas dirigidas o procedentes de esa dirección IP.

    2. **Redes concretas (*net*).** Por ejemplo, el filtro:

        ```
        net 158.42
        ```

        captura todas las tramas dirigidas o procedentes de dicha red IP.

    3. **Puertos determinados (*port*)**. Así, el filtro:

        ```
        port 7
        ```

        captura todas las tramas dirigidas o procedentes del puerto 7 de cualquier computador, tanto TCP como UDP.

    Si no se especifica el tipo se asume el tipo *host,* es decir, una única máquina. También se encuentra especificado el identificador *broadcast*, que permite hacer referencia a las direcciones de difusión.

- De dirección: Especifican si el identificador debe entenderse sobre el origen (src), el destino (dst) o ambos (src or dst). Los calificadores de dirección posibles son `src`, `dst`, `src or dst`, y `src and dst`. Se aplicarán sobre alguno de los calificadores de tipo. Ejemplos:

```
src 158.42.181.18
src port 25
src or dst net 158.42
```

- De protocolo: Indican un tipo de protocolo. En nuestro caso resultan de interés los protocolos `arp`, `ip`, `icmp`, `tcp`, o `udp`. Como norma general, no existen calificadores de nivel de aplicación, debiendo emplearse el número de puerto y protocolo de transporte para capturar el tráfico correspondiente a un protocolo de aplicación concreto.

Se pueden combinar varias primitivas mediante los conectores `and` y `or`. También es posible usar el operador `not`.

Ejemplo: `udp and dst 158.42.148.3`

Esto mismo es aplicable también a los identificadores.

Ejemplo: `dst 158.42.181.4 or 158.42.181.15`

También se puede hacer uso de paréntesis para indicar las precedencias deseadas.

Ejemplo: `dst 158.42.181.4 and (udp or icmp)`

> Para acceder a la documentación detallada acerca de los filtros y sus posibilidades se puede acudir al manual en línea de la orden **tcpdump**, tecleando en una consola de texto la orden `man tcpdump`.

4. Estudio de la pila de protocolos TCP/IP mediante el Wireshark

Como hemos visto anteriormente el Wireshark nos permite analizar con detalle los paquetes que circulan por la red. Nosotros vamos a utilizar esta herramienta para profundizar en los conceptos estudiados en las sesiones de aula acerca de la pila de protocolos TCP/IP. Wireshark nos va a permitir, por ejemplo, comprobar el encapsulado que realiza cada uno de los niveles de la pila con un mensaje que se genera en el nivel de aplicación.

Ejercicio 1.1:

Selecciona la interfaz adecuada (posiblemente enp0s3) y realiza una captura aplicando el filtro de captura "port 80". Con este filtro deberían capturarse todos los mensajes del protocolo HTTP que se generan cuando desde nuestro navegador solicitamos una página web a un servidor.

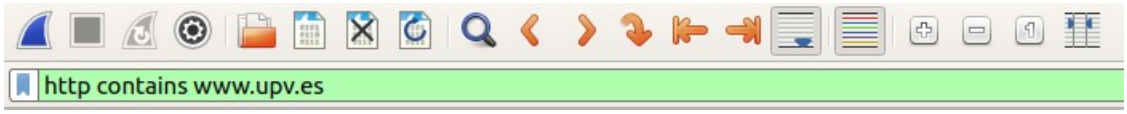
Para generar tráfico accede a una página web con el navegador. Por ejemplo "http://www.upv.es". Una vez hayas conseguido dicha página, para de capturar.

IMPORTANTE:

- Ten en cuenta que cuando el URL comienza por "https" en vez de por "http", el tráfico no se dirige al puerto 80 sino al 443, por lo que no podrías capturarlo con el filtro "port 80".
- Debes generar el tráfico desde el navegador de la máquina virtual. En caso contrario, no podrás capturarlo.

Deberías haber obtenido algo similar a lo que puedes ver en la Figura1. Para tener una captura más clara puedes utilizar un filtro de pantalla: "http" que te dejará ver solamente los paquetes del protocolo HTTP.

Además como queremos analizar solo los paquetes dirigidos al servidor de la UPV, podemos aún afinar más el filtro de visualización con "http contains www.upv.es".



Después de aplicar este filtro, selecciona en la primera ventana de las tres, el primer paquete de petición del protocolo HTTP (mensaje GET) dirigido al servidor web de la UPV que genera nuestro navegador.

Este mensaje se genera, por tanto, en el ***Nivel de Aplicación*** de nuestro ordenador (cliente HTTP) y va dirigido al ***Nivel de Aplicación*** del servidor web de la UPV (servidor HTTP). Pero para que este mensaje llegue a su destino:

- Tendrá que descender por todos los niveles inferiores de la pila de protocolos TCP/IP de nuestro ordenador hasta llegar al ***Nivel Físico.*** En ese descenso cada uno de los niveles intermedios habrá añadido una cabecera correspondiente al protocolo implicado en ese nivel (encapsulación).
- El ***Nivel Físico*** de nuestro equipo será el encargado de enviar el mensaje HTTP junto con la cabeceras añadidas en cada nivel por el medio físico disponible. En este caso el medio físico es un cableado de red (par trenzado).

- Una vez llegue al destino lo recibirá el Nivel Físico del servidor e irá ascendiendo por cada uno de los niveles TCP/IP hasta llegar al ***Nivel de Aplicación.*** Pero en ese ascenso cada nivel se quedará con la cabecera correspondiente a su nivel, que le va a permitir "dialogar" con el nivel equivalente en el otro extremo. De forma que lo que llegará al Nivel de Aplicación del servidor será unicamente el mensaje HTTP que le envía nuestra máquina.

Vamos ahora a analizar con más detalle cada uno de los niveles que aparecen en este primer paquete seleccionado. Para ello nos centramos en la **ventana intermedia**.

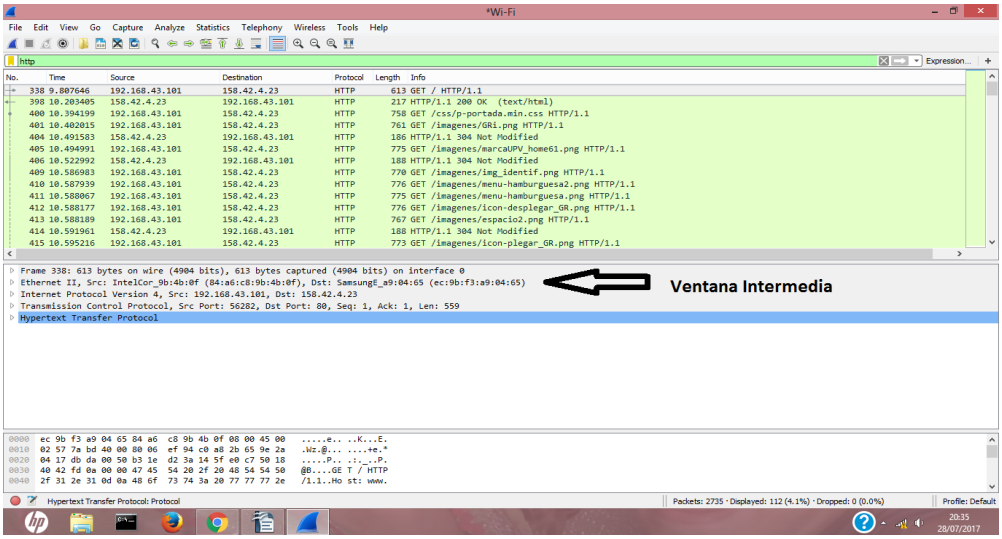


Figura 5. Ventana Intermedia de la Captura

En ella podemos observar 5 líneas. Cada una de ellas se corresponde con un nivel de la pila de protocolos TCP/IP pero de forma invertida a como lo representamos habitualmente: la primera línea se corresponde con el ***Nivel Físico*** y la última línea se corresponde con el ***Nivel de Aplicación.***

Cada uno de estos niveles tiene a la izquierda una pestaña en la que pinchando podemos desplegar y ver cada uno de los campos que pertenecen a ese nivel. Si nos interesa copiar el contenido de algunas de las líneas a un documento (por ejemplo, la información sobre direcciones y puertos), Wireshark nos permite seleccionarlas y copiarlas fácilmente utilizando después las opciones "Copy → Value" a las que accedemos con el botón derecho del ratón. La combinación de teclas Ctl + Shift + V permite también el mismo efecto.

Ejercicio 1.2:

Vamos a analizar el primer paquete que hemos seleccionado en el ejercicio 1.1. Nos centramos ahora en la **quinta línea** de la ventana intermedia, "Hypertext Transfer Protocol", que corresponde al ***Nivel de Aplicación.*** Si la desplegamos podemos ver completo el mensaje de petición HTTP que genera nuestra máquina y que va dirigido

al servidor web de la UPV. ¿Qué indica el campo “Host”? ¿Cómo se llama el campo que indica qué tipo de software está usando el cliente?

Ejercicio 1.3:

Desplegamos ahora la **cuarta línea**, “Transmission Control Protocol”, correspondiente al ***Nivel de Transporte.*** ¿Qué protocolo aparece en este nivel?[1] ¿Qué otro protocolo podría aparecer? ¿Cuál es la diferencia fundamental entre ambos protocolos? ¿Qué identificadores aparecen en este nivel como origen y destino? Toma nota de ellos.

El ***Nivel de Transporte*** es el encargado de recoger el mensaje generado por la aplicación y prepararlo para ser enviado de acuerdo a los requerimientos de la aplicación. En el destino este nivel es el que entrega el mensaje a la aplicación correspondiente. Por tanto, necesita diferenciar a cada aplicación con un identificador distinto. Este identificador es el **número de puerto.**

Observa los puertos origen y destino que aparecen en este nivel del programa Wireshark. ¿Son números arbitrarios que elige el sistema operativo? ¿En qué se diferencian el puerto del cliente y el del servidor?

Ejercicio 1.4:

Nos centramos ahora en la **tercera línea** de la ventana intermedia (“Internet Protocol”). Si la desplegamos podemos ver con detalle los campos pertenecientes a la cabecera que ha añadido el ***Nivel de Red.*** Recordemos que este nivel es el encargado de encaminar el paquete a través de Internet para llegar a su destino. Para ello necesita tener perfectamente identificadas tanto la máquina origen como la máquina destino. ¿Qué protocolo aparece en este nivel? ¿Qué tipo de direcciones identifican en este nivel el origen y el destino? ¿A quienes pertenecen las dos direcciones que aparecen? Apúntalas.

Otro aspecto interesante es poder ver el valor de cada uno de los campos que nos aparece en la segunda ventana, en hexadecimal y en ASCII. Para ello basta con seleccionar dicho campo en la ventana intermedia y observar los valores que quedan resaltados en la ventana inferior.

Indica los valores correspondientes en hexadecimal de las direcciones IP origen y destino.

[1] Transmission Control Protocol es el nombre que corresponde a las siglas TCP.

Ejercicio 1.5:

Vamos a analizar ahora la **segunda línea**, "Ethernet II", de la ventana intermedia que corresponde con el ***Nivel de Enlace.*** Dicho nivel se encarga de encaminar el paquete dentro de la red local a la que pertenece el equipo. Y, por tanto, las direcciones que aparecen como origen y destino pertenecerán siempre a la red local del equipo en el que se hace la captura.

¿Qué tipo de direcciones aparecen como origen y destino en este nivel? Toma nota de estas direcciones (observa que en este nivel el orden es distinto al habitual, la primera dirección es la dirección destino y la segunda, la dirección fuente) .

Y, por último, la **primera línea** de la ventana intermedia que tiene que ver con el ***Nivel Físico.*** Si desplegamos esta línea podemos ver detalles relacionados con este nivel cuya misión es enviar el mensaje HTTP, junto con todas la cabeceras, a través del medio físico. En este nivel podemos ver entre otras cosas el identificador de la interfaz por la que se transmite la información.

Ejercicio 2:

Pasamos a trabajar con otro protocolo de aplicación, el protocolo DNS. Este protocolo permite averiguar la dirección IP correspondiente a un nombre de máquina. Elimina el filtro de pantalla "http contains www.upv.es" (atención, una vez borrado sigue actuando hasta que presiones la tecla <Enter>). Inicia una nueva captura modificando previamente el filtro de captura a "port 53". Con este filtro deberían capturarse todos los mensajes del protocolo del nivel de aplicación DNS. A continuación, carga en el navegador una página web de un servidor al que no hayas accedido previamente en la sesión de prácticas y detén la captura.

Selecciona el primer paquete DNS que se corresponderá con la consulta que tu máquina hace al servidor DNS acerca del nombre de la página web que has consultado. Analiza detalladamente las cabeceras de cada nivel. ¿Aparecen diferencias en cuanto a los protocolos implicados en cada nivel con respecto a la captura anterior?[2]

Compara las direcciones origen y destino de esta captura y las de los ejercicios 1.4 y 1.5 (captura HTTP). ¿Hay alguna cabecera en la que las direcciones origen y destino coincidan con las que aparecen en la captura anterior? ¿Por qué? (Recuerda que sólo puedes ver direcciones de enlace de dispositivos que estén en tu misma red IP. Cuando el destino final está fuera de tu red, los paquetes salen a través del router, y la dirección de enlace del router es la que vemos).

[2]User Datagram Protocol es el nombre que corresponde a las siglas UDP.

5. Aplicación Netcat

Habitualmente nos conectamos a aplicaciones que tienen la funcionalidad de servidor. Sin embargo, en prácticas posteriores vamos a desarrollar aplicaciones de red que van a ser tanto clientes como servidores, y que trabajarán sobre TCP y sobre UDP. Para comprobar su correcto funcionamiento vamos a necesitar otro tipo de herramientas de red más versátiles.

Por ejemplo, para probar un cliente que acabamos de diseñar nos puede interesar abrir un puerto determinado de nuestro equipo, que permanezca a la escucha, actuando como un supuesto servidor. Y además poder especificar si ese puerto es TCP o UDP.

Todas estas funcionalidades nos las va a proporcionar la aplicación ***Netcat***.

Podemos hacer uso de *netcat* para establecer una conexión TCP con un servidor determinado. Para ello tendremos que identificar el servidor con el que queremos conectar indicando su número de puerto. Usaremos *netcat* desde un terminal de la siguiente forma:

$ nc nombre_servidor número_puerto

Ejercicio 3:

Prueba a conectarte mediante *netcat* al servidor de *echo* (puerto 7) del servidor *zoltar.redes.upv.es* del laboratorio de redes (si estás fuera de la UPV necesitarás establecer sesión VPN con la UPV). En este servicio no necesitarás introducir nombre de usuario ni contraseña. El servicio de eco intenta imitar el comportamiento del eco real, que nos devuelve todo lo que le gritamos. En este caso le enviaremos una línea de mensaje y el servidor nos responderá con una línea idéntica. Comprueba que funciona correctamente el diálogo con el servidor enviando cualquier mensaje que se te ocurra. Puedes cortar la comunicación con la combinación de teclas: <Ctrl><C>.

Puedes probar también a conectarte al servidor *daytime* (puerto 13). Ahora el servidor te enviará un mensaje en cuanto detecte que te has conectado. Para que la conexión finalice pulsa la tecla <Enter>.

Fíjate que estos dos servidores, que implementan servicios muy sencillos, muestran los dos comportamientos posibles de un servidor TCP. Una posibilidad es esperar hasta que el cliente envíe un mensaje al servidor. La otra es que sea el servidor el primero que envíe un mensaje, en cuanto detecta que un cliente se ha conectado.

Desde el momento en que estamos conectados con un servidor determinado, el diálogo que mantengamos con el servidor deberá seguir el protocolo del nivel de aplicación correspondiente. Por ejemplo, si hacemos un *netcat* al servidor web de la UPV ($ *nc www.upv.es 80)* el protocolo de aplicación que deberá regir la comunicación entre cliente y servidor a partir de ese momento será HTTP.

Como hemos dicho antes *netcat* nos va a permitir abrir un puerto y dejarlo a la escucha, tal y como haría un servidor. Para ello utilizaremos la siguiente forma:

$ ***nc -l número_puerto***

La opción -l (*listen*) le indica al programa que cree un *socket* y lo ponga a la escucha en el ordenador local y en el número de puerto que se indica.

Además, *netcat* también nos va a permitir ponernos en contacto con este puerto, haciendo la funcionalidad de cliente. Para ello tendremos que indicar el **nombre o la dirección IP** y el **número de puerto** donde hemos creado nuestro servidor :

$ nc nombre_servidor número_puerto

Ejercicio 4:

Utilizando dos consolas de texto diferentes establece un chat entre ambas mediante el comando *netcat*. En una de las consolas dejarás un puerto a la escucha, teniendo la precaución de no utilizar un puerto ya utilizado en la máquina virtual. Utiliza, por ejemplo, el 9999. En el otro terminal utilizarás *netcat* como cliente para conectarte a ese puerto. En este caso para identificar el equipo servidor puedes utilizar *localhost,* ya que está en la propia máquina. Una vez realizada la conexión comprueba que todo lo que se escribe en una ventana es recibido en la otra. Puedes cortar la comunicación con la combinación de teclas: <Ctrl><C>.

¡¡Recuerda que **siempre tendrá que estar el puerto a la escucha antes de que el cliente intente conectarse a él!!**

Por defecto *netcat* utiliza el protocolo TCP como protocolo de transporte, salvo que se le indique lo contrario. Para indicarle que queremos usar el protocolo UDP lo haremos con la opción *-u.* Los comando anteriores quedarán:

$ *nc -u -l número_puerto*

$ nc -u nombre_servidor número_puerto

Ejercicio 5:

Repite la experiencia anterior de comunicación entre dos consolas de texto pero en este caso utilizando puertos UDP.

¿Podríamos mantener ambos chats activos (TCP y UDP) utilizando en ambos casos el mismo número de puerto? Compruébalo y justifica tu respuesta.

Y, por último, comentar que en próximas prácticas será interesante poder probar y depurar aplicaciones de red en la propia máquina, tal y como hemos hecho ahora. Sin necesidad de disponer de dos máquinas. ¡¡Así es que recuerda el uso de *netcat*!!