Práctica 2:
Protocolo HTTP

1 Trabajo previo.

Para el correcto desarrollo de la práctica es conveniente acudir al laboratorio habiendo estudiado el protocolo HTTP, en sus versiones 1.0 y 1.1 De esa forma podrás responder correctamente las cuestiones que se plantean y comprender mejor las capturas de tráfico realizadas. Por ello, previamente debes:

- Estudiar el capítulo 2.2 del libro de Kurose.

Si deseas información de consulta opcional, puedes consultar también los siguientes recursos:

  - HTTP: http://www.rfc-editor.org/rfc/rfc2616.txt
  - Guía usuario Wireshark: http://www.wireshark.org/docs/
  - Monitor de red de Firefox https://developer.mozilla.org/es/docs/Tools/Monitor_de_Red

2 Objetivos de la práctica.

Al acabar la práctica debes ser capaz de realizar las tareas que se indican a continuación.

Relacionadas con el programa nc:

- Utilizar el programa nc como cliente TCP básico, por ejemplo, para enviar una petición HTTP 1.0 y HTTP 1.1.

Relacionadas con el programa Wireshark:

- Utilizar Wireshark para capturar tráfico de una aplicación determinada, utilizando un filtro de captura, en particular, para tráfico HTTP.
- Utilizar la orden "Follow TCP Stream" para visualizar el diálogo cliente/servidor de una aplicación.

Relacionadas con las herramientas del navegador:

- Utilizar las herramientas de red para analizar las cabeceras de peticiones y respuestas
- Utilizar las herramientas de red para realizar análisis de rendimiento.

En cuanto a HTTP, deberías poder:

- Diferenciar entre las cabeceras obligatorias y opcionales en las versiones 1.0 y 1.1 en las peticiones del cliente.
- Describir cómo se separan las cabeceras y el cuerpo del mensaje (si lo hay) en las peticiones del cliente y las respuestas del servidor.
- Explicar cuándo se utilizan peticiones condicionales, en qué consisten, cómo las identifica el servidor y qué respuestas puede enviar.
- Describir el significado de las cabeceras: *Date*, *Last-Modified* y *Content-Length*.

3 Primera parte. Composición básica de una petición HTTP.

3.1 Uso de netcat para realizar una petición HTTP.

La herramienta netcat, ya descrita y utilizada en la primera práctica del curso (ver Apéndice A), nos permite enviar y recibir información utilizando sockets TCP y UDP mediante sencillas órdenes desde el *shell*. Por lo tanto, necesitarás utilizarla desde un terminal. El apéndice A muestra información detallada de los parámetros que acepta.

Dado que el protocolo HTTP es un protocolo orientado a caracteres su sintaxis es perfectamente legible, por lo que perfectamente podemos utilizar la orden *netcat* para componer peticiones y obtener respuestas.

Ejercicio 1: Uso de netcat.

Envía mediante *netcat* una petición HTTP de tipo GET al servidor web cuya dirección IP es 158.42.180.62 (serveis.redes.upv.es) en su puerto 80, para obtener el documento cuya URI es "http://serveis.redes.upv.es/ejercicio1.html". Prueba a solicitarlo con HTTP/1.0. Por ejemplo, teclea:

```
nc -C 158.42.180.62 80
```

Con esta orden estás conectándote, mediante el protocolo TCP (por defecto), al puerto 80 (puerto HTTP) del servidor 158.42.180.62 (*serveis-rdc.redes.upv.es*). La opción `-C` indica a *netcat* que debe enviar la pareja de caracteres *<CR><LF>* cuando se pulse la tecla *<Enter>* (en OSX la "c" debe ser minúscula).

Una vez establecida la conexión debes solicitar el recurso:

```
GET /ejercicio1.html HTTP/1.0 ↵
↵
```

Nota: Hay que pulsar la tecla de retorno de carro **dos veces** después de la línea de petición. Como resultado de esta acción se recibirá un texto parecido al siguiente:

```
HTTP/1.1 200 OK
Date: Thu, 06 Oct 2016 17:13:16 GMT
Server: Apache
Accept-Ranges: bytes
X-UA-Compatible: IE=EmulateIE7, IE=9
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Language: es
                ...
```

El contenido recibido se ha mostrado por la salida estándar, que era en este caso la pantalla. Si se desea capturar este texto se puede redirigir la salida a un archivo mediante el operador ">".

Cuestiones:

Cuestión 1.1. ¿Qué hace la opción "–C" en la orden "nc"? ¿Por qué es necesaria?

Cuestión 1.2. ¿Cuál es el código de la respuesta? ¿Qué significa ese código?

Nota muy importante: Si el código de la respuesta es 400 consulta con tu profesor.

Ejercicio 2: Uso de netcat (continuación).

Vuelve a establecer una nueva conexión con *netcat* y solicita de nuevo la página, esta vez utilizando la versión 1.1 del protocolo HTTP.

Cuestiones:

Cuestión 2.1. ¿Qué cabeceras son obligatorias en la petición del cliente?

Cuestión 2.2. ¿Cómo se indica el fin de las cabeceras en la petición del cliente y en la respuesta del servidor?

Cuestión 2.3. ¿Cuándo ha sido modificado por última vez el fichero *html* en el servidor?

Cuestión 2.4. ¿Qué diferencia hay entre las cabeceras *Date* y *Last-modified*?

Cuestión 2.5. ¿Cuál es el tamaño en bytes de la página que envía el servidor indicado en la cabecera "*Content-Length*"?

Ejercicio 3: Acceso a servidor virtual.

Vuelve a conectarte al mismo servidor mediante la orden:

```
nc -C 158.42.180.62 80
```

Realiza la misma petición que en el ejercicio anterior, pero utilizando la cabecera:

```
Host: zoltar.redes.upv.es
```

Cuestiones:

Cuestión 3.1. ¿La respuesta obtenida es la misma que en el ejercicio anterior?

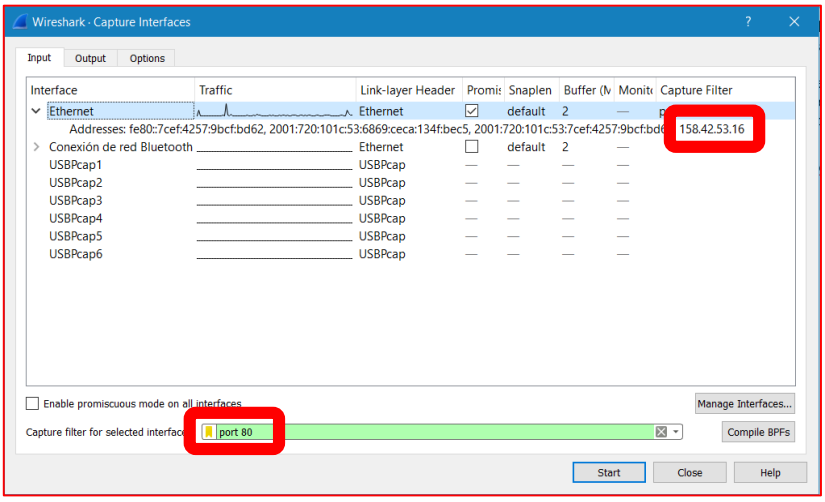
Cuestión 3.2. ¿Puedes explicar qué es lo que está pasando?

3.2 Usos de los analizadores de protocolos.

En esta parte de la sesión pasaremos a trabajar con el analizador de protocolos *Wireshark*, que nos permitirá analizar el tráfico HTTP intercambiado entre nuestro navegador y los servidores web a los que nos conectemos.

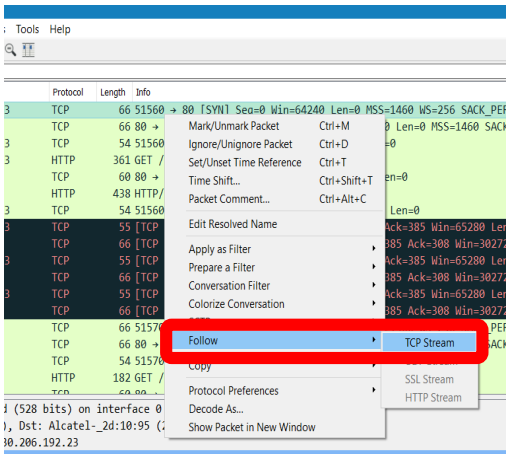
Recuerda que antes de generar el tráfico (solicitar las páginas web indicadas) debes iniciar las capturas con el programa *Wireshark*. Para ello ten en cuenta un par de pasos básicos:

- Es conveniente que compruebes que el interfaz sobre el que vas a capturar es el adecuado. Para ello accede al menú "*Capture"* y selecciona la opción "*Options*". Te aparecerá una nueva ventana similar a la mostrada en la figura siguiente, llamada "*Capture Options*". Comprueba que la interfaz tiene una dirección IP asociada que empiece por 158.42, y si no fuera así, selecciona la interfaz que tenga ese tipo de dirección asociada. Se trata de la dirección IP pública que nos permite acceder a Internet.
- Como vamos a capturar tráfico HTTP en el apartado "*Capture Filter*" introduce "port 80".

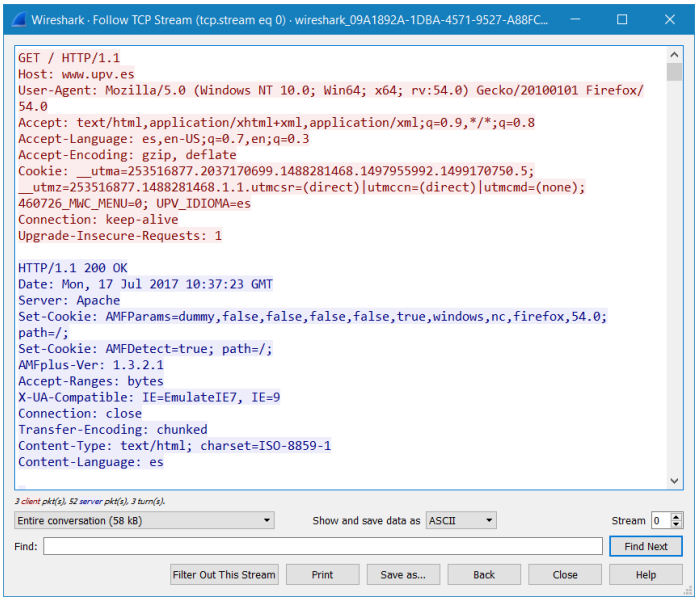


Utilidad de la orden "*Follow TCP Stream*"

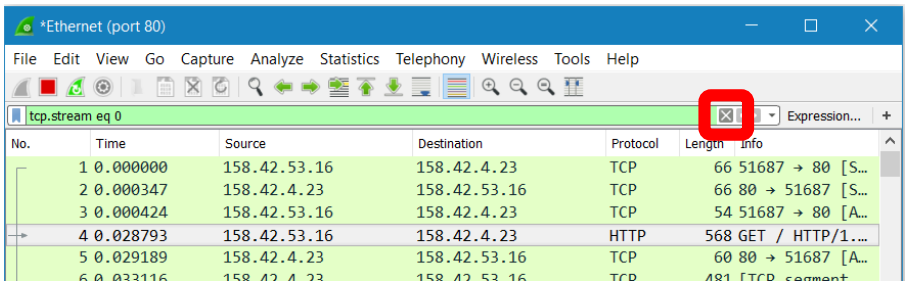
Al trabajar con protocolos de aplicación basados en TCP puede resultar muy útil poder ver los datos intercambiados en una conexión TCP de la misma forma que los ve el nivel de aplicación. Especialmente cuando en la captura aparecen intercalados los paquetes asociados a diferentes conexiones TCP. Para ello el programa *Wireshark* proporciona la opción "*Follow TCP Stream*". Al seleccionar un paquete de una conexión determinada, en la que estemos interesados, y seleccionar la opción "*Follow TCP Stream*", del menú "*Analyze*", *Wireshark* aplica automáticamente un filtro de visualización y abre una ventana adicional mostrando en orden todos los datos intercambiados a nivel de aplicación por el cliente y el servidor. También puedes activarlo mediante el menú que se abre al pulsar el botón derecho del ratón, después de haber seleccionado el paquete correspondiente:



Además, para facilitar su interpretación utiliza diferente color para los datos de cada uno de los extremos de la comunicación:



Si en la captura original aparecían varias conexiones TCP distintas, ahora sólo podrás ver una. Si deseas volver a verlas todas debes eliminar el filtro de visualización que se ha añadido automáticamente al seleccionar la opción "*Follow TCP Stream*". Pulsa "*el botón X*" para eliminarlo. La figura siguiente te servirá de ayuda.



Ejercicio 4: Captura de tráfico sobre el puerto 80.

Configura el programa Wireshark para capturar el tráfico del puerto 80 mediante un filtro de captura (sintaxis "port 80"). Inicia la captura y, utilizando el navegador `Firefox`, accede a este recurso:

`http://www.upv.es/`

Analiza, utilizando la opción "*Follow TCP Stream*", el primer intercambio petición/respuesta HTTP del tráfico capturado. Debes haber obtenido una respuesta del servidor "`HTTP/1.1 200 OK`", si no ha sido así revisa el URL que habías

introducido en el navegador (hay que incluir obligatoriamente el campo de protocolo de la URL). **Si tienes que volver a capturar la página deberás vaciar previamente la caché del navegador.**

Cuestiones:

Cuestión 4.1. Tu navegador, ¿utiliza HTTP 1.0 o 1.1? ¿Qué versión de HTTP está utilizando el servidor?

Cuestión 4.2. ¿Qué idiomas está indicando tu navegador al servidor que prefiere?

Cuestión 4.3. ¿Cuál es la dirección IP de tu ordenador? ¿Y la del servidor? (las direcciones IP puedes mirarlas en la ventana general de captura, aparecen etiquetadas como "*Source*" y "*Destination*").

Cuestión 4.4. Analiza las cabeceras "*Connection*" del cliente y el servidor, ¿están utilizando conexiones persistentes o no persistentes?

Cuestión 4.5. ¿Figura la cabecera "*Transfer-Encoding"* en la respuesta? ¿Qué valor tiene y que significa ese método de transferencia?

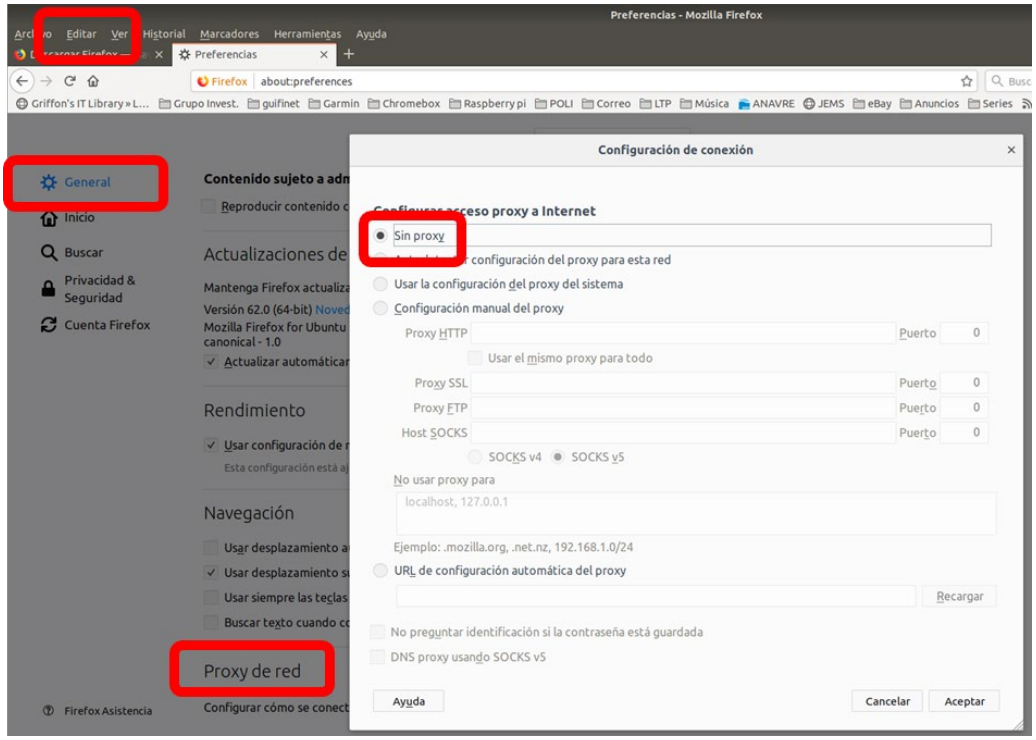
Cuestión 4.6. ¿Por qué no figura la cabecera "*Content-Length*"?

4 Segunda parte. Interacciones avanzadas HTTP.

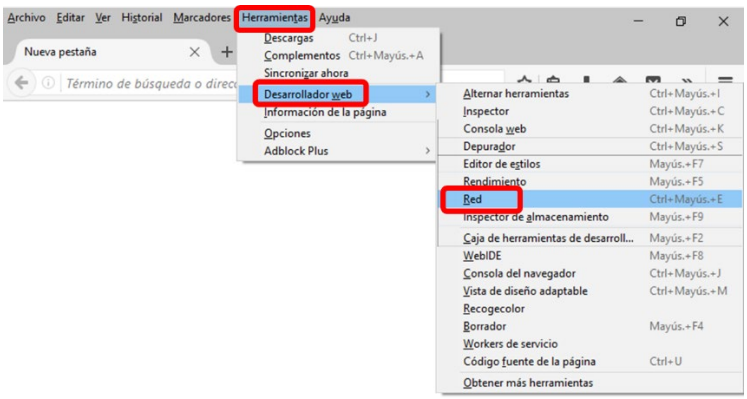
En los siguientes ejercicios vamos a utilizar el propio navegador `Firefox` como herramienta de análisis.

Para un correcto desarrollo de esta parte de la práctica es **MUY IMPORTANTE** que todas las URI's sean tecleadas en la barra de navegación tal y como aparecen, incluyendo la parte de **"`http://`"**.

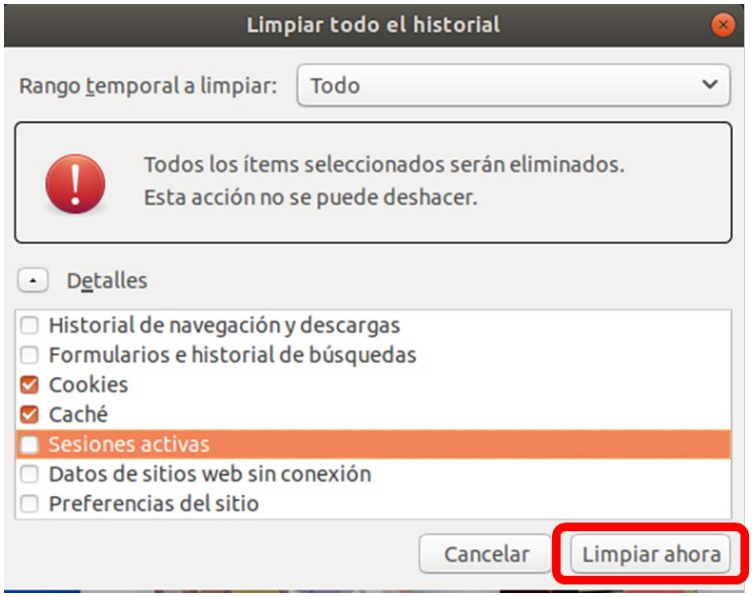
También es **MUY IMPORTANTE** tener desactivado cualquier Servidor Proxy si queremos que los siguientes ejercicios salgan correctamente, ya que el proxy podría introducir modificaciones en algunas cabeceras. Para desactivarlo accederemos al menú "`Editar`" > "`Preferencias`" ("`Herramientas`" > "`Opciones`" en `Windows`), en el apartado "`General`" desplazaremos hacia abajo la ventana hasta encontrar la sección "`Configuración de red`", pulsaremos el botón "`Configuración`" y seleccionaremos "`Sin Proxy`"



A continuación nos moveremos, dentro del menú, por los submenús de "`Herramientas`" > "`Desarrollador web`" > "`Red`" (o directamente pulsando "`Ctr+Mayús+E`"). En la parte inferior de la ventana se abrirá un marco.



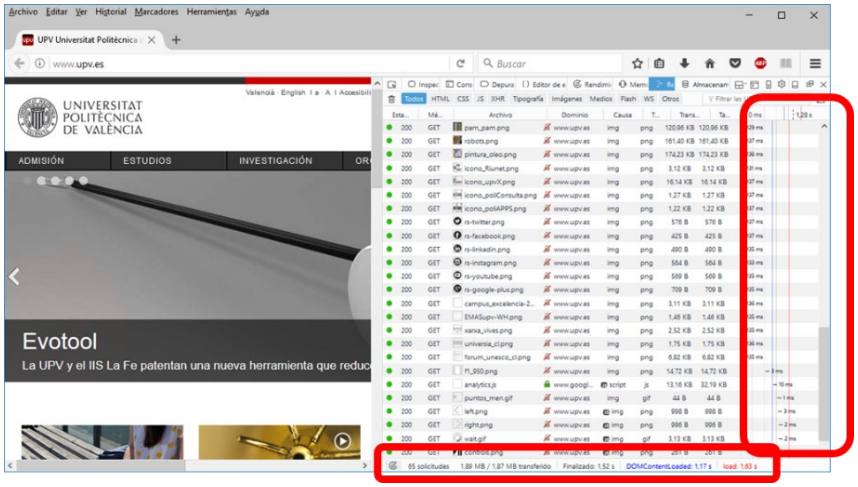
Borraremos los datos de la caché (pulsando"`Ctr+Mayús+Supr`")



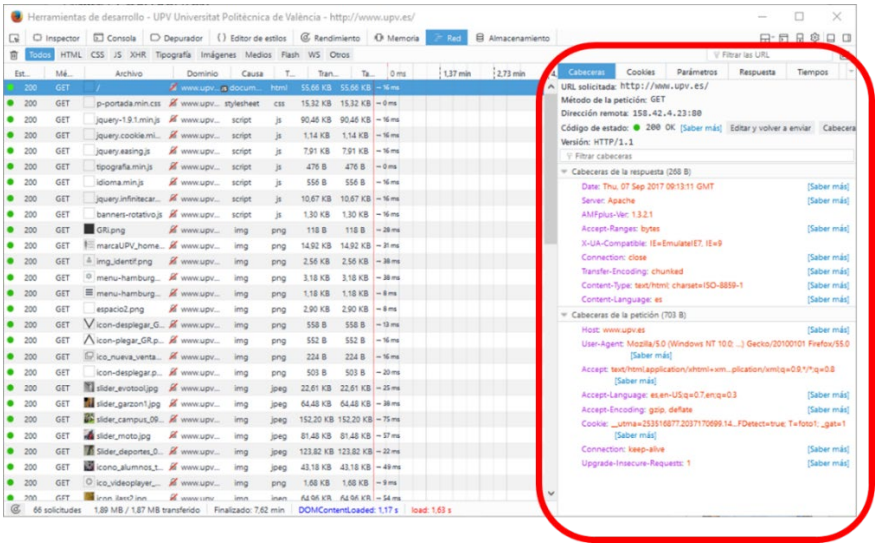
y accederemos a la siguiente URI:

`http://www.upv.es/`

Deberías obtener algo similar a lo que se muestra en la siguiente imagen. La parte remarcada a la derecha se denomina “la cascada” y visualiza la forma en la que se van obteniendo los distintos recursos que componen el documento. Abajo, en la barra de estado figuran los siguientes valores: número de peticiones, tamaño total / tamaño transferido (**nótese** que algunos recursos pueden ser comprimidos previamente a la transferencia, pueden haber aciertos en la cache, etc.), tiempo de finalización, y los tiempos en los que se producen los eventos *`DOMContentLoaded`* y *`load`* (en el primero el documento HTML se ha obtenido y analizado y se empieza a construir la interfaz, en el segundo, además, se han obtenido también el resto de recursos).



Para analizar las cabeceras HTTP de un par petición/respuesta basta con seleccionarlo y luego visualizar la pestaña de "`Cabeceras`".



Veamos ahora cómo afecta la caché del navegador a las peticiones que realiza el cliente. Si el navegador ya tiene el recurso en su caché, y sospecha que puede haber sido modificado desde que lo obtuvo, el cliente realizará una petición condicional, utilizando la cabecera "*`If-modified-since`*". Si el recurso no se ha modificado en el servidor, la respuesta que envía tampoco será la habitual (`200 OK`). Veamos las diferencias, tanto en la petición del cliente, como en la respuesta del servidor.

Ejercicio 5: La cache.

Borra la cache y accede a la URI:

http://www.upv.es/

Comprueba que todas las respuestas tienen "`200`" como código de estado. Anota los tiempos y tamaños que aparecen en la barra de estado de la herramienta. Sin borrar la cache, recarga (pulsando F5) de nuevo la página y vuelve a anotar los tiempos y tamaños.

Cuestiones:

Cuestión 5.1. ¿Se observa alguna diferencia respecto de los tiempos y tamaños?

Cuestión 5.2. ¿Cuántas respuestas con códigos distintos de 200 has obtenido? ¿Qué significan esos códigos?

Cuestión 5.3. En las cabeceras de petición, ¿figura alguna cabecera condicional además de la de "*`If-modified-since`*"? ¿Qué significa?

Ejercicio 6: Ejercicio opcional. Redirecciones.

Borra la cache y accede a la URI:

http://www.elpais.es/

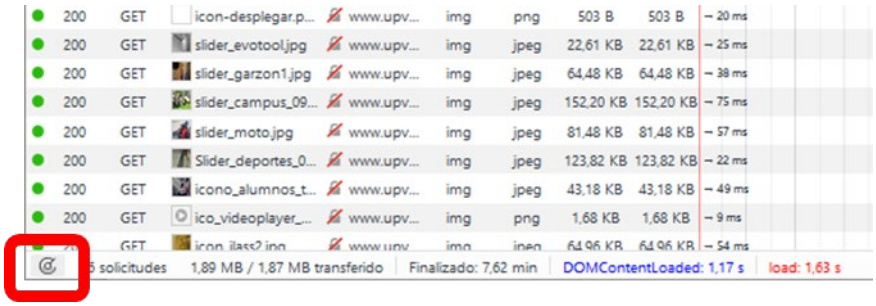
Cuestiones:

Cuestión 6.1. Al intentar descargar el recurso raíz el código de la respuesta es "301". ¿Qué significa este código?

Cuestión 6.2. ¿Qué otras respuestas de tipo "301" se obtienen y qué funciones realizan?

Ejercicio 7: Ejercicio opcional. Análisis del uso de la cache.

Ahora haremos un análisis comparativo de la utilización de la cache sobre la captura anterior. Para ello, pulsa en el icono a la izquierda de la barra de estado.



Podemos definir la **Tasa de Aciertos** como el cociente entre el número de recursos que se encuentran válidos en la cache respecto del número de recursos solicitados.

Cuestiones:

Nota: Al contestar las cuestiones téngase en cuenta que se está utilizando el protocolo HTTP/2, por lo que el tamaño real de los objetos va a coincidir con el tamaño transferido.

Cuestión 7.1. Calcula la tasa de aciertos obtenida.

Cuestión 7.2. Calcula ahora la tasa de aciertos en bytes.

Cuestión 7.3. ¿Hay diferencia entre ambas? ¿qué mide cada una de ellas?

Apéndice A

La herramienta netcat: usos y parámetros

```
usage: nc [-46DdhklnrStUuvzC] [-i interval] [-P proxy_username] [-p
source_port]
          [-s source_ip_address] [-T ToS] [-w timeout] [-X proxy_protocol]
          [-x proxy_address[:port]] [hostname] [port[s]]

Command Summary:
-4               Use IPv4
-6               Use IPv6
-D               Enable the debug socket option
-d               Detach from stdin
-h               This help text
-i secs          Delay interval for lines sent, ports scanned
-k               Keep inbound sockets open for multiple connects
-l               Listen mode, for inbound connects
-n               Suppress name/port resolutions
-P proxyuser     Username for proxy authentication
-p port          Specify local port for remote connects
-q secs          quit after EOF on stdin and delay of secs (-1 to not quit)
-r               Randomize remote ports
-S               Enable the TCP MD5 signature option
-s addr          Local source address
-T ToS           Set IP Type of Service
-C               Send CRLF as line-ending
-t               Answer TELNET negotiation
-U               Use UNIX domain socket
-u               UDP mode
-v               Verbose
-w secs          Timeout for connects and final net reads
-X proto         Proxy protocol: "4", "5" (SOCKS) or "connect"
-x addr[:port]   Specify proxy address and port
-z               Zero-I/O mode [used for scanning]

Port numbers can be individual or ranges: lo-hi [inclusive]
```