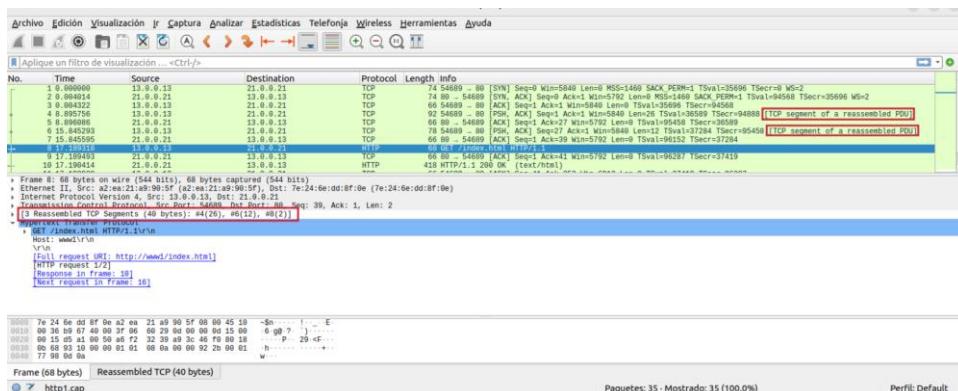


Práctica 5: HTTP

Nota previa (Recordatorio)



Varios reassembles tcp segments formaras más tarde un paquete entero, si pinchas en el paquete entero y despliegas las pestañas veras de que numero de fragmentos está formado.

1. Comunicación cliente-servidor HTTP

1. Indica qué dirección IP es la de la máquina cliente HTTP y cuál la del servidor.

El primer paquete dirección 13.0.0.13 - 21.0.0.21 es de client-servidor. Ademas podemos ver el paquete con (SYN).

El segundo paquete va con dirección 21-13. Contiene (SYN, ACK) y pertenece al servidor.

2. Indica qué versión HTTP se utiliza en esta comunicación.

66 80 -> 54689 [ACK] Seq=1 ACK=1
68 GET /index.html HTTP/1.1
66 80 -> 54689 [ACK] Seq=1 ACK=1

En el paquete 8 podemos ver como especifica la versión 1.1.

3. Indica el número de conexiones que se ven en el fichero de captura, y si los recursos del mismo servidor se transfieren todos por la misma conexión TCP o se usa una conexión TCP diferente para cada uno.

Hay 4 conexiones, dos por parte del cliente y otras dos por parte del servidor.

Cliente: Paquetes 8 y 16.

Servidor: Paquetes 10 y 30.

- Normalmente una página web está compuesta por varios recursos alojados en diferentes servidores.
 - Tras obtenerse el recurso principal, de él se extrae la relación de recursos adicionales que forman parte de la misma página:
 - Para cada nuevo servidor, se abrirá una conexión TCP nueva para pedirle sus recursos.
 - Si hay varios recursos en un mismo servidor, suelen solicitarse todos ellos a través de la misma conexión TCP.

Se encuentran en la misma conexión tcp ya que posteriormente pide una imagen, pero en el mismo servidor, por lo que no hace falta cambiar de conexión tcp.

4. ¿Cuántas peticiones GET observas desde el cliente?

Dos. La primera en el paquete 8 y la segunda en 16. Primero pide la dirección de la página y luego una imagen.

5. ¿Cuántas URLs crees que ha escrito el usuario en el navegador para obtener dicha captura? ¿Cuál/es? ¿Por qué?

Una única URI, que es la página a la que quería acceder.

6. Fíjate en el contenido de la página index.html que se ha descargado el cliente. ¿Qué crees que ocurrirá cuando el navegador se haya descargado index.html? Abre la captura http2.cap y responde a las siguientes preguntas:

(Contenido de la página index.html)

```
▼ Hypertext Transfer Protocol
  ▼ GET /index.html HTTP/1.1\r\n
    ▼ [Expert Info (Chat/Sequence): GET /index.html HTTP/1.1\r\n]
      [GET /index.html HTTP/1.1\r\n]
      [Severity level: Chat]
      [Group: Sequence]
      Request Method: GET
      Request URI: /index.html
      Request Version: HTTP/1.1
      Host: www1\r\n
\r\n
      [Full request URI: http://www1/index.html]
      [HTTP request 1/2]
      [Response in frame: 10]
      [Next request in frame: 16]
```

Se le proporcionará la dirección al cliente y podrá acceder.

7. Indica qué versión HTTP se utiliza en esta comunicación.

```
GET /foto1.jpg HTTP/1.0
```

Se está usando la versión 1.0.

8. Indica el número de conexiones que se ven en el fichero de captura, y si los recursos del mismo servidor se transfieren todos por la misma conexión TCP o se usa una conexión TCP diferente para cada uno

Cliente: 4, 27, 43 y 56.

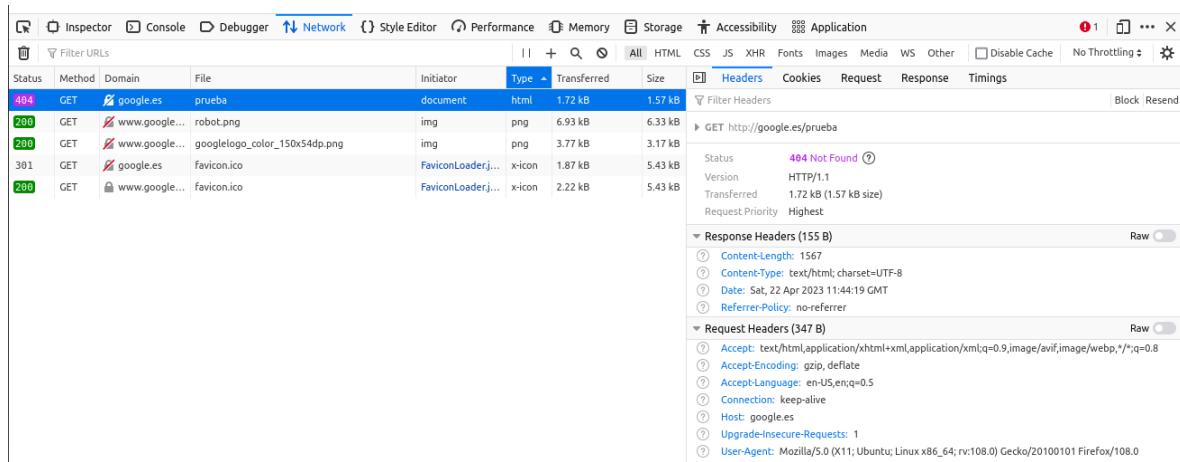
Servidor: 6, 20, 38, 54 y 66.

Es una conexión tcp del mismo servidor ya que el servidor solo está pidiendo fotos y no es necesario la redirección a otro servidor.

2. Diferentes tipos de respuestas de un servidor

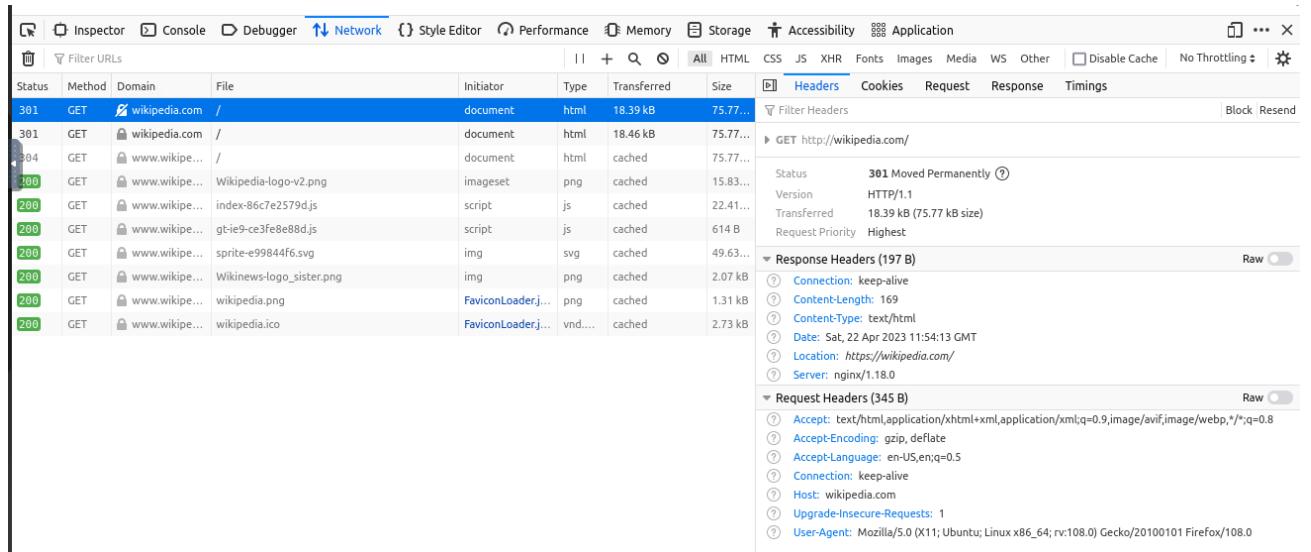


1. Escribe en la barra de dirección la URL <http://www.google.es/prueba>. Selecciona en el panel inferior la pestaña 'Todos' y pulsa sobre la primera petición GET que aparece (la del fichero 'prueba') y en el panel que se abre a la derecha selecciona la pestaña Cabeceras, como se muestra en la figura 3.



2. Escribe ahora en la barra de dirección la URL <http://www.wikipedia.com>. En la lista de peticiones, desplázate al principio del todo para localizar el primer GET. Seleccionando ese primer GET mira las líneas de cabecera que ves en la respuesta y explica la segunda petición GET.

Primera solicitud get:



Segunda solicitud get:

The screenshot shows the Firefox Developer Tools Network tab. It displays a list of 10 network requests made to wikipedia.com. The requests include the main document, various images (e.g., logo-v2.png, sprite-99844f6.svg), and scripts (e.g., index-86c7e257d.js, gt-ie9-ce3fe8e08d.js). The Headers panel shows the response headers for the main document, including:

- Status: 301 Moved Permanently
- Version: HTTP/2
- Transferred: 18.46 kB (75.77 kB size)
- Request Priority: Highest
- Response Headers (269 B):
 - content-length: 169
 - content-type: text/html
 - date: Sat, 22 Apr 2023 11:54:13 GMT
 - location: https://www.wikipedia.org/
 - server: nginx/1.18.0
 - strict-transport-security: max-age=106384710; includeSubDomains; preload
 - X-Firefox-Spdy: h2
- Request Headers (443 B):
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.5
 - Accept-Encoding: gzip, deflate, br
 - Accept-Language: en-US,en;q=0.5
 - Connection: keep-alive
 - Host: wikipedia.com
 - Sec-Fetch-Dest: document
 - Sec-Fetch-Mode: navigate
 - Sec-Fetch-Site: none
 - Sec-Fetch-User: ?1
 - TE: trailers
 - Upgrade-Insecure-Requests: 1

At the bottom, it shows 10 requests, 321.91 kB transferred, a finish time of 911 ms, DOMContentLoaded at 3.75 s, and load at 3.76 s.

3. Formularios en HTTP

1. Indica el número de conexiones entre cliente y servidor que aparecen en la captura.

Cliente: 6 y 17

Servidor: 8 y 21

2. Busca en la captura el paquete donde el servidor le envía al cliente un formulario. Indica los nombres de los campos del formulario que llenará el usuario.

En el paquete 6 el cliente solicita un formulario y en el 8 el servidor da el ok y lo entrega. En el formulario tendrá que llenar el nombre, la edad y un número de teléfono.

3. Indica cómo se decide la manera en la que debe enviar el cliente los datos del formulario (GET/POST). ¿Qué método se está usando en este caso?

● GET:

- Sigue una petición para obtener un recurso. El recurso solicitado se indica en la línea inicial.
- Ejemplo: solicitar una página web.

● POST:

- Envía datos a un recurso del servidor. El recurso se indica en la línea inicial y los datos van en el cuerpo de la petición.
- Ejemplo: enviar los datos que rellena el usuario a través de un formulario web.

● POST:

- Envía datos al servidor, normalmente los introducidos por el usuario en un formulario.
- **Los datos van en el cuerpo.**
- El *path* de la línea inicial (URL) se refiere normalmente al programa que tratará los datos que se envían.

● GET:

- GET también permite enviar los datos de un formulario. En este caso, **los datos van en el path de la línea inicial (URL), y no hay cuerpo**.
- El tamaño de los datos subidos con GET está limitado por el tamaño máximo de una URL (255 caracteres), por lo que NO se utiliza GET para subir datos de formularios grandes.

466 GET /cgi-bin/prog2.pl?nombre=Ana&edad=25&telefono=123456789 HTTP/1.0

Tanto para pedir el formulario como para entregarlo usa el método get

4. Busca en la captura el paquete donde el cliente envía los datos del formulario al servidor y comprueba que se está realizando con el método que has indicado en el apartado anterior.

Usa método get, ya que los datos los introduce en la línea del URL como se puede ver en las capturas anteriores.

5. ¿Cómo se llama el programa del servidor que va a recibir esos datos?

/cgi-bin/prog2.pl (Captura de arriba)

6. ¿Dónde viajan los datos que el cliente le envía al servidor? ¿Cuáles son esos datos?

Los datos viajan en el paquete 17 y al ser un mensaje get los datos van en el path de la línea inicial (URL), y no hay cuerpo. Los datos son el nombre, la edad y el número de teléfono.

Abre la captura http4.cap y responde a las siguientes preguntas:

9. Busca en la captura el paquete donde el servidor le envía al cliente un formulario. Indica los nombres de los campos del formulario que rellenará el usuario.

En el fragmento 6, en respuesta a la solicitud del formulario en el fragmento 4.

10. ¿Qué método se usa para enviar los datos al servidor en este caso? ¿Cómo lo sabes?

Frame	Source IP	Destination IP	Protocol	Length	Action	Details
14	5.383122	14.0.0.14	22.0.0.22	HTTP	470	POST /cgi-bin/prog2.pl HTTP/1.0 (application/x-www-form-urlencoded)
15	5.383387	22.0.0.22	14.0.0.14	TCP	66	80 → 34697 [ACK] Seq=1 Ack=405 Win=6864 Len=0 TSval=130735 TSecr=1615
16	5.414220	22.0.0.22	14.0.0.14	HTTP	696	HTTP/1.1 200 OK (text/html)
17	5.414220	22.0.0.22	14.0.0.14	TCP	66	80 → 34697 [ETX] ACK1 Seq=631 Ack=405 Win=6864 Len=0 TSval=130735 TSecr=1615

Frame 14: 470 bytes on wire (3760 bits), 470 bytes captured (3760 bits)
Ethernet II, Src: 72:6e:78:f6:53:d3 (72:6e:78:f6:53:d3), Dst: c2:de:96:e4:27:78 (c2:de:96:e4:27:78)
Internet Protocol Version 4, Src: 14.0.0.14, Dst: 22.0.0.22
Transmission Control Protocol, Src Port: 34697, Dst Port: 80, Seq: 1, Ack: 1, Len: 404
Hypertext Transfer Protocol
POST /cgi-bin/prog2.pl HTTP/1.0\r\nHost: www\r\n

Se usa el método post.

11. Busca en la captura el paquete donde el cliente le envía los datos del formulario al servidor y comprueba que se está realizando con el método que has indicado en el apartado anterior.

Frame	Source IP	Destination IP	Protocol	Length	Action	Details
14	5.383122	14.0.0.14	22.0.0.22	HTTP	470	POST /cgi-bin/prog2.pl HTTP/1.0 (application/x-www-form-urlencoded)
15	5.383387	22.0.0.22	14.0.0.14	TCP	66	80 → 34697 [ACK] Seq=1 Ack=405 Win=6864 Len=0 TSval=130735 TSecr=1615
16	5.414220	22.0.0.22	14.0.0.14	HTTP	696	HTTP/1.1 200 OK (text/html)
17	5.414220	22.0.0.22	14.0.0.14	TCP	66	80 → 34697 [ETX] ACK1 Seq=631 Ack=405 Win=6864 Len=0 TSval=130735 TSecr=1615

Frame 14: 470 bytes on wire (3760 bits), 470 bytes captured (3760 bits)
Ethernet II, Src: 72:6e:78:f6:53:d3 (72:6e:78:f6:53:d3), Dst: c2:de:96:e4:27:78 (c2:de:96:e4:27:78)
Internet Protocol Version 4, Src: 14.0.0.14, Dst: 22.0.0.22
Transmission Control Protocol, Src Port: 34697, Dst Port: 80, Seq: 1, Ack: 1, Len: 404
Hypertext Transfer Protocol
POST /cgi-bin/prog2.pl HTTP/1.0\r\nHost: www\r\n

12. ¿Cómo se llama el programa del servidor que va a recibir esos datos?

/cgi-bin/prog2.pl

13. Indica en qué parte del mensaje van los datos del formulario que el cliente le envía al servidor.

● **POST:**

- Envía datos al servidor, normalmente los introducidos por el usuario en un formulario.
- **Los datos van en el cuerpo.**
- El *path* de la línea inicial (URL) se refiere normalmente al programa que tratará los datos que se envían.

Al ser un mensaje post los datos del formulario van en el cuerpo.

14. Indica qué cabecera es la que representa el tipo de contenido que el cliente envía al servidor y cuál es su valor.

```
HyperText Transfer Protocol
  ▶ POST /cgi-bin/prog2.pl HTTP/1.0\r\n
    Host: www2\r\n
    Accept: text/html, text/plain, text/css, text/sgml, */*;q=0.01\r\n
    Accept-Encoding: gzip, compress, bzip2\r\n
    Accept-Language: en\r\n
    Pragma: no-cache\r\n
    Cache-Control: no-cache\r\n
    User-Agent: Lynx/2.8.7dev.10 libwww-FM/2.14 SSL-MM/1.4.1\r\n
    Referer: http://www2/form3.html\r\n
    Content-type: application/x-www-form-urlencoded\r\n
  ▶ Content-length: 29\r\n
  \r\n
  [Full request URI: http://www2/cgi-bin/prog2.pl]
  [HTTP request 1/1]
  [Response in frame: 16]
  File Data: 29 bytes
  ▶ HTML Form URL Encoded: application/x-www-form-urlencoded
```

15. Explica si en este caso es necesario la cabecera Content-Length en el mensaje HTTP que el cliente envía al servidor con los datos del formulario. ¿Por qué?

Si es necesario ya que esta cabecera especifica la longitud exacta de esos datos. Sin esta cabecera, el servidor no sabrá cuántos bytes esperar y no podrá procesar el formulario correctamente.

16. Observa si el servidor envía alguna respuesta cuando recibe los datos del formulario del cliente. En caso afirmativo localiza el número de paquete y observa en las cabeceras HTTP: tipo de contenido, longitud y cuerpo del mensaje.

La respuesta se realiza en el fragmento 16.

```
HyperText Transfer Protocol
  ▶ HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
    Date: Sun, 19 May 2013 16:10:47 GMT\r\n
    Server: Apache/2.2.9 (Debian)\r\n
    Set-Cookie: Authenticated=YES; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=www2; Path=/;\r\n
    Set-Cookie: UserID=1111; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=www2; Path=/;\r\n
    Set-Cookie: Age=23; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=www2; Path=/departamento/;\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    Content-Length: 147\r\n
      [Content length: 147]
      Connection: close\r\n
      Content-Type: text/html\r\n
      \r\n
      [HTTP response 1/1]
      [Time since request: 0.031098000 seconds]
      [Request in frame: 14]
      [Request URI: http://www2/cgi-bin/prog2.pl]
      Content-encoded entity body (gzip): 147 bytes -> 162 bytes
      File Data: 162 bytes
  ▶ Line-based text data: text/html (1 lines)
    <html><head><title>Resultado </title></head><body><h2>Hola Jaime, con NIF 1111, de 23 años de edad </h2>Te he enviado tus datos en forma de cookies</body></ht..
```

4. Cookies

4.1. Almacén de cookies en el navegador Firefox

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.ayto-fuenlabrada.es	/	document	html	33.03 kB	32.63 kB
200	GET	www.ayto-fuenlabrada.es	resolucion.asp?resolucion=1360*768	portal.js:24 (xhr)	html	229 B	0 B

Headers		Cookies	Request	Response	Timings	Security
GET https://www.ayto-fuenlabrada.es/						
Status	200 OK					
Version	HTTP/1.1					
Transferred	33.03 kB (32.63 kB size)					
Request Priority	Highest					
Response Headers (403 B)						
Accept-Language	es-es					
Cache-control	private					
Connection	Keep-Alive					
Content-Length	32626					
Content-Type	text/html; Charset=ISO-8859-1					
Date	Sun, 23 Apr 2023 09:38:19 GMT					
Expires	Sun, 23 Apr 2023 09:38:19 GMT					
Keep-Alive	timeout=5, max=100					
Pragma	no-cache					
Server	Microsoft-IIS/6.0					
Set-Cookie	ASPSESSIONIDQCCQCRDC=PPGIAKCCONPKKALLNMDHJPGA; path=/; X-Powered-By: ASP.NET					
Request Headers (439 B)						
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8					
Accept-Encoding	gzip, deflate, br					
Accept-Language	en-US,en;q=0.5					
Connection	keep-alive					
Host	www.ayto-fuenlabrada.es					
Sec-Fetch-Dest	document					

2. Pulsa en el deslizador de 'Sin procesar' en las cabeceras de la respuesta, o selecciona la pestaña "Cookies" para poder ver de forma más clara el contenido de las cookies. Observa que no hay campo 'Domain', lo que indica que la cookie sólo es válida para el servidor que la ha enviado, y que tampoco hay fecha de expiración, lo que indica que es una cookie de sesión, que se eliminará cuando se cierre el navegador.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
ASPSESSIONIDQCCQCRDC	PPGIAKCCONPKKALLNMDHJPGA		/	Tue, 22 Apr 2025 09:3...	60	false	false	None	Sun, 23 Apr 2023 09:39...

3. Selecciona ahora la herramienta de desarrollador "Almacenamiento" y en el panel de la izquierda despliega "Cookies" para ver las cookies obtenidas al descargar esta página, véase la figura 5. Observa como ahí aparecen rellenos los campos de 'Domain' y 'Expires'.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
__utma	65738302.2084527945.1682242787.1682242787.1682242787.1682242787	.ayto-fuenlabrada.es	/	Tue, 22 Apr 2025 09:3...	60	false	false	None	Sun, 23 Apr 2023 09:39...
__utmb	65738302.1.10.1682242787	.ayto-fuenlabrada.es	/	Sun, 23 Apr 2023 10:09:3...	30	false	false	None	Sun, 23 Apr 2023 09:39...
__utmc	65738302	.ayto-fuenlabrada.es	/	Session	14	false	false	None	Sun, 23 Apr 2023 09:39...
__utmt	1	.ayto-fuenlabrada.es	/	Sun, 23 Apr 2023 09:49:...	7	false	false	None	Sun, 23 Apr 2023 09:39...
__utmx	65738302.1682242787.1.1.utmcsr=(direct) utmccn=(direct) utmccv=(direct)	.ayto-fuenlabrada.es	/	Sun, 22 Oct 2023 21:39:...	75	false	false	None	Sun, 23 Apr 2023 09:39...
ASPSESSIONIDQCCQCRDC	PPGIAKCCONPKKALLNMDHJPGA	www.ayto-fuenla...	/	Session	44	false	false	None	Sun, 23 Apr 2023 09:39...

4. Vuelve a la herramienta de desarrollador "Red" y pulsa sobre la segunda petición GET que aparece, y observa las cookies que envía el cliente al servidor. Comprueba como se trata de la

cookie obtenida del servidor en el primer GET. Activa 'Sin procesar' o la pestaña Cookies para poder ver mejor los valores que se envían.

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace	Security
200	GET	www.ayto-fu...	/	document	html	33.03 kB	32.63...		ASPSESSIONIDQCCQCRCDC	"PPGIAKCCONPKALLNMDHJPGA"				
200	GET	www.ayto-fu...	resolucion.asp?resolucion=1360*768	portal.js:24 (xhr)	html	229 B	0 B							

(Mismo número de cookies que el apartado anterior)

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_utm	65738302.2084527945.1682242787.1682242787.1682242787...	ayto-fuenlabrada.es	/	Tue, 22 Apr 2025 09:39...	60	false	false	None	Sun, 23 Apr 2023 09:39...
_utmb	65738302.1.10.1682242787	ayto-fuenlabrada.es	/	Sun, 23 Apr 2023 10:09...	30	false	false	None	Sun, 23 Apr 2023 09:39...
_utmc	65738302	ayto-fuenlabrada.es	/	Session	14	false	false	None	Sun, 23 Apr 2023 09:39...
_utmt	1	ayto-fuenlabrada.es	/	Sun, 23 Apr 2023 09:49...	7	false	false	None	Sun, 23 Apr 2023 09:39...
_utmz	65738302.1682242787.1.1.utmcsr=(direct) utmccn=(direct)...	ayto-fuenlabrada.es	/	Sun, 22 Oct 2023 21:39...	75	false	false	None	Sun, 23 Apr 2023 09:39...
ASPSESSIONIDQCC...	PPGIAKCCONPKALLNMDHJPGA	www.ayto-fuenla...	/	Session	44	false	false	None	Sun, 23 Apr 2023 09:39...

4.2. Envío de Cookies en mensajes HTTP

1. Indica qué cookies envía el servidor al cliente:

El servidor le envía 5 cookies, que son:

Index	Length	Source IP	Destination IP	Protocol	Port	Timestamp	HTTP Status	Content-Type
6	176268	22.0.0.22	13.0.0.13	HTTP		998	HTTP/1.1 200 OK (text/html)	
7	176541	22.0.0.22	13.0.0.13	TCP		66	80 → 43559 [FIN, ACK] Seq=933 Ack=445 Win=6864 Len=0 T	
8	176762	13.0.0.13	22.0.0.22	TCP		66	43559 → 80 [ACK] Seq=445 Ack=933 Win=7704 Len=0 T	
9	213669	13.0.0.13	22.0.0.22	TCP		66	43559 → 80 [ACK] Seq=445 Ack=934 Win=7704 Len=0 T	

Frame 6: 998 bytes on wire (7984 bits), 998 bytes captured (7984 bits)
 ▶ Ethernet II, Src: 56:34:4c:ca:91:4a (56:34:4c:ca:91:4a), Dst: fe:96:85:bc:af:2f (fe:96:85:bc:af:2f)
 ▶ Internet Protocol Version 4, Src: 22.0.0.22, Dst: 13.0.0.13
 ▶ Transmission Control Protocol, Src Port: 80, Dst Port: 43559, Seq: 1, Ack: 445, Len: 932
 ▶ Hypertext Transfer Protocol
 ▶ HTTP/1.1 200 OK\r\n
 Date: Tue, 05 Jun 2012 05:25:35 GMT\r\n
 Server: Apache/2.2.9 (Debian)\r\n
 Set-Cookie: Nombre=Pepe Lozano; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=elcortebritanico.com; Path=/facturas/;\r\n
 Set-Cookie: Nif=22034; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=elcortebritanico.com; Path=/facturas;\r\n
 Set-Cookie: Edad=22; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=elcortebritanico.com; Path=/facturas;\r\n
 Set-Cookie: Carrito=obsequio-bienvenida; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=elcortebritanico.com; Path=/tienda;\r\n
 Set-Cookie: Sesión=1003; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=elcortebritanico.com; Path=/;\r\n
 Vary: Accept-Encoding\r\n
 Content-Encoding: gzip\r\n
 ▶ Content-Length: 154\r\n
 Connection: close\r\n
 Content-Type: text/html\r\n
 \r\n

2. Indica qué cookies enviará el cliente al servidor cuando acceda a la página con la URL:

<http://elcortebritanico/tienda/index.html>

Set-Cookie: Carrito=obsequio-bienvenida; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT; Domain=elcortebritanico.com; Path=/tienda;\r\n

Enviarán esta cookie ya que el path de la url es /tienda, al igual que la cookie.

3. ¿Y si el cliente accediera en el año 2025 a dicha URL?

También le valdría ya que la fecha de expiración es en 2030.

4. ¿Y si el cliente accediera en el año 2035 a dicha URL?

La cookie habría expirado y ya no valdría.

Abre la captura http6.cap y responde a las siguientes preguntas:

5. Indica qué cookies el cliente está enviando al servidor.

```

+ 4 0.012180 13.0.0.13 22.0.0.22 HTTP 356 GET /dir1/index.html HTTP/1.0
5 A A12668 22 A A 22 TCP 66 AA → 40963 ACK1 Seq=1 Ack=291 Win
Frame 4: 356 bytes on wire (2848 bits), 356 bytes captured (2848 bits)
Ethernet II, Src: 36:25:18:c0:47:99 (36:25:18:c0:47:99), Dst: 16:53:a7:0d:94:b0 (16:53:a7:0d:94:b0)
Internet Protocol Version 4, Src: 13.0.0.13, Dst: 22.0.0.22
Transmission Control Protocol, Src Port: 40963, Dst Port: 80, Seq: 1, Ack: 1, Len: 290
Hypertext Transfer Protocol
  GET /dir1/index.html HTTP/1.0\r\n
  Host: www2\r\n
  Accept: text/html, text/plain, text/css, text/sgml, */*;q=0.01\r\n
  Accept-Encoding: gzip, compress, bzip2\r\n
  Accept-Language: en\r\n
  User-Agent: Lynx/2.8.7dev.10 libwww-FM/2.14 SSL-MM/1.4.1\r\n
  Cookie2: $Version="1"\r\n
  Cookie: Nif=123456789A; Nombre=Andres\r\n
  Cookie pair: Nif=123456789A
  Cookie pair: Nombre=Andres

```

6. ¿Por qué crees que le envía dichas cookies?

Porque son cookies almacenadas anteriormente y que aún no han expirado. Las envía porque cumplen las condiciones que pide la página.

7. Escribe un ejemplo de las posibles cabeceras que le habrá enviado dicho servidor al cliente previamente.

Pues viendo la cap y viendo que las cookies son el nombre y el nif. Probablemente en el pasado el servidor le pidió una encuesta con estos datos.

La cabecera podría ser:

Set-cookie: Nombre=Andrés; Expires=Tuesday, 28-May-2029 17:55:00 GMT;
Domain=encuestastrabajo.com; Path=/socorrista/.

Set-cookie: Nif=123456789A; Expires=Tuesday, 28-May-2029 17:55:00 GMT;
Domain=encuestastrabajo.com; Path=/socorrista/.

8. A partir de la información de la captura ¿si el cliente accede a otra página con la URL: <http://www2/dir1/dir2/index.html> enviaría esas mismas cookies, más cookies o menos?

Enviaría estas cookies también porque comienzan con el mismo path, pero al haberlas aceptado ahora no será necesario al cambiar de servidor.

Pero como en el servidor estaremos buscando algo más específico que en la página principal, saldrán nuevas cookies para aceptar.

Ya que si nos metemos en aliexpress por ejemplo, no nos saltan las cookies de ropa, electrodomésticos etc. Cuando inicias te salen unas cookies más generales, y cuando vas especificando que buscas, te saldrán las cookies de eso en específico

9. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: <http://www2/index.html> enviaría esas mismas cookies, más cookies o menos?

Enviar las cookies que tengan el mismo path que la url anterior. Las cookies con path diferentes no se envían, y luego una vez accedida a la nueva url quizás sean necesarias otras diferentes.

10. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: <http://www/index.html> enviaría esas mismas cookies, más cookies o menos?

Se enviarán diferentes cookies, más en específico se enviarán las cookies del apartado anterior, ya que sí que tienen el mismo path.

5. Comunicación a través de un Proxy HTTP

1. Indica qué dirección IP es el cliente, el proxy y el servidor final.

Cliente: 13.0.0.13

Servidor: 22.0.0.22

Proxy: 23.0.0.23

2. ¿Qué diferencia la petición HTTP que realiza el cliente de la petición que realiza el proxy?

El cliente pide directamente la URL de la páginas, mientras que el proxy lo que pide es el path

3. Identifica el nombre de la máquina donde se encuentra el servidor HTTP.

ww2

4. ¿Se puede saber de la petición que realiza el proxy que dicho proxy tiene almacenada en su caché esa página?

```
▼ Hypertext Transfer Protocol
  ▼ GET /index.html HTTP/1.1\r\n
    ▶ [Expert Info (Chat/Sequence): GET /index.html HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /index.html
      Request Version: HTTP/1.1
      Host: www2\r\n
      User-Agent: Wget/1.11.4\r\n
      Accept: */*\r\n
      If-None-Match: "4006-78-4946ad6523a80"\r\n
      If-Modified-Since: Sat, 06 Nov 2010 23:34:50 GMT\r\n
      Via: 1.0 www3:8080\r\n
      Connection: Keep-Alive\r\n
      \r\n
      [Full request URI: http://www2/index.html]
      [HTTP request 1/1]
      [Response in frame: 13]
```

Estos son los datos que nos proporciona el proxy, no podemos saber su caché

Abre la captura http8.cap y responde a las siguientes preguntas:

5. Indica el número de conexiones entre cliente y servidor que aparecen en la captura.

Cliente-Servidor: Paquetes 6 y 14

Servidor-Cliente: Paquetes 8 y 34

6. Explica qué es lo que se está descargando el cliente del servidor HTTP y cuantos objetos se descarga.

Se está descargando una imagen. Se descarga 8 objetos.

7. Observa en las cabeceras HTTP el tipo de contenido de cada uno de los objetos.

Los fragmentos del paquete de la foto son tcp, por lo que no tienen http.

8. ¿Podrías saber si los paquetes capturados se corresponden a la comunicación entre un cliente y un proxy HTTP, entre un cliente y servidor final HTTP o entre un proxy y el servidor final HTTP? ¿Por qué?

Los paquetes capturados son entre cliente y proxy, ya que los paquetes del cliente solicitan una http entera y los paquetes enviados por el servidor llevan la cabecera via.

9. Sabiendo que la comunicación se ha realizado a través de un proxy HTTP, mira las cabeceras HTTP que envía dicho proxy para ver si en ellas existe alguna que muestre cuál es su nombre.

Abre la captura http9.cap y responde a las siguientes preguntas:

10. ¿Podrías saber si los paquetes capturados se corresponden a la comunicación entre un cliente y un proxy HTTP, entre un cliente y servidor final HTTP o entre un proxy y el servidor final?

Es una comunicación en la que el cliente pide información sobre una página, pero el proxy ya tiene esa información del servidor por lo que no hace falta volver a preguntarle. Es una comunicación entre cliente y proxy.

6. Cachés en HTTP

6.1. Caché en un proxy HTTP

1. Indica cuáles son las direcciones IP del cliente, proxy y servidor web. ¿Cómo lo sabes?

Cliente: 12.0.0.100

Servidor: 12.0.0.1 / 11.0.0.1

Proxy: 14.0.0.100

Se sabe sacando el camino que sigue la información.

2. Explica qué es lo que ocurre en estas capturas.

Primero el cliente solicita al proxy información sobre el servidor, esto ocurre dos veces. Mas tarde el proxy solicita esta info al servidor y este le responde que no ha cambiado nada. (Supongo que no ha cambiado nada respecto a al info que tenía almacenada el proxy en su caché)

3. Localiza los campos relevantes con respecto al tratamiento de caché que incluye en las líneas de cabecera el servidor. ¿Qué significan?

Cache-Control: public, max-age=60\r\n

Significa que se puede guardar en una caché pública y que tiempo máximo de almacena antes de que se convierta en información antigua es de 60 seg.

4. Explica qué ocurre en la segunda consulta que realiza el cliente.

El cliente realiza una consulta y el servidor le responde con la información que tenía guardada en la cache.

5. Viendo los paquetes 14 y 16 de la captura http10.cap indica cómo se puede saber que el contenido proviene de una caché.

Si miramos la respuesta del fragmento 16, vemos que en el age tiene 12/r/ que es mayor que 0, por lo que el contenido estaba guardado en la caché

6. ¿Crees que el cliente tiene caché?

No, porque si no no sería necesaria la respuesta del proxy, ya como el proxy tiene guardada la información en la cache, el cliente también podría haberlo hecho en caso de tener caché.