



## Práctica 1 2020/2021 (Solucion)

Redes y aplicaciones Internet (Universitat Oberta de Catalunya)



Escanea para abrir en Studocu



## **Redes y Aplicaciones Internet y Sistemas de Internet**

### **Reto 2**

### **Práctica 1: Observa la red y sabrás qué está pasando**

En esta práctica analizaremos algunos de los protocolos que intervienen constantemente en las comunicaciones que hacemos cada día por la red. Para este fin, nos será muy útil instalar un analizador de paquetes, que, visualmente, nos permita entender qué está pasando de un vistazo. Los analizadores son programas que capturan los paquetes que envía y recibe la tarjeta de red de nuestro ordenador. En el apartado **Anexo: Orientaciones previas Wireshark** tenéis instrucciones para instalar y ver cómo funciona el analizador de paquetes que utilizaremos en esta asignatura: *Wireshark*.

Una vez preparada la herramienta necesaria para observar qué pasa por la red, necesitamos poner un escenario de estudio común a todos nosotros: la navegación por Internet. Para resolver los siguientes ejercicios, tendréis que iniciar el programa Wireshark y hacer que empiece a capturar tramas. Una vez hecho esto, abrid el navegador e id a la página principal de la UOC. Tenéis que acceder con vuestro usuario y contraseña y entrar en el aula de la asignatura. En este punto, podéis parar la captura de paquetes del Wireshark e incluso guardarla en un fichero para seguir con la práctica más adelante. Sólo necesitaréis abrir el Wireshark y cargar el archivo con el nombre que le pusisteis al guardar.

### **Calificación**

Esta práctica consta de cuatro partes, que se han de realizar en orden secuencial (primero la parte 1, después la parte 2...). La nota irá en función de las partes entregadas:

<i>Parte 1. El nivel de enlace y de red</i>	<i>Nota máxima C-</i>
<i>Parte 2. El nivel de transporte</i>	<i>Nota máxima C+</i>
<i>Parte 3. El nivel de aplicación</i>	<i>Nota máxima B</i>
<i>Parte 4. Seguridad en la red</i>	<i>Nota máxima A</i>

Para obtener los niveles indicados de calificación es necesario realizar todos los ejercicios y apartados de las partes indicadas. En caso de haber algún ejercicio o apartado no realizado o incompleto, implicará no obtener la calificación correspondiente.

**Nota:** *Entregad vuestras respuestas en el registro de evaluación continuada del aula de la asignatura. Usad algún programa de compresión para hacer la entrega en un solo archivo comprimido que incluya un único fichero pdf con las respuestas a las preguntas de todas las partes y el fichero de capturas de Wireshark. Llamad al fichero: Practica1\_Apellido1\_Apellido2\_Nombre.[zip, tar, ...]*

## Primera parte (nota máxima: C-): El nivel de enlace y de red

La pila de protocolos Internet está estructurada en varios niveles. En esta primera parte, trabajaréis con los niveles más bajos y relacionados con las tarjetas de red: el nivel de enlace (*Link Layer*) y de red (*Network Layer*). Trabajaréis con direcciones MAC y direcciones IP, identificando unívocamente la tarjeta de red y el dispositivo, respectivamente.

Como material de soporte para realizar esta segunda parte de la práctica os proponemos los siguientes apartados del libro Computer Networking (7a edición), especialmente los apartados relacionados con las cabeceras de los protocolos:

- 4.3 Switched Local Area Networks pag.504-505
  - 6.4.2 Ethernet
- 4.3 The Internet Protocol (IP) pag.358-360
  - 4.3.1 IPv4 Datagram Format

1. Clica sobre una trama concreta de la captura de paquetes que habéis realizado con Wireshark. Comprueba que incluye una cabecera **Ethernet** con datos de la capa de enlace. Muestra mediante una captura de pantalla los contenidos de la cabecera Ethernet. Responde a las siguientes cuestiones:

The screenshot displays the Wireshark interface with the following details:

- Packet List:**

No.	Time	Source	Destination	Protocol	Length	Info
4768	22.595241	100.100.1.1	192.168.1.163	DNS	228	Standard query response 0xc917 A cognito-identity.e
4769	22.596368	100.100.1.1	192.168.1.163	DNS	324	Standard query response 0xee2d AAAA cognito-identit
4770	22.597209	2a0c:5a84:3301:6900...	cognito-identity.eu...	TCP	86	55209 → 443 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS
4771	22.597358	2a0c:5a84:3301:6900...	2a0c:5a80:0:2::1	DNS	120	Standard query 0x8890 A cognito-identity.eu-west-1.
4772	22.622594	192.168.1.163	100.100.1.1	DNS	100	Standard query 0x8890 A cognito-identity.eu-west-1.
4773	22.630852	localhost.local	239.255.255.250	SSDP	371	NOTIFY * HTTP/1.1
4774	22.635529	100.100.1.1	192.168.1.163	DNS	228	Standard query response 0x8890 A cognito-identity.e
4775	22.636211	2a0c:5a84:3301:6900...	2a0c:5a80:0:2::1	DNS	120	Standard query 0xf4c2 AAAA cognito-identity.eu-west
- Packet Details:**
  - Frame 4766: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface \Device\NPF\_{FF9C8FD1-BA58-4948-9AA3-5D4645008E2}
  - Ethernet II, Src: IntelCor\_22:d5:ad (50:eb:71:22:d5:ad), Dst: zte\_87:d5:02 (90:fd:73:87:d5:02)
    - Destination: zte\_87:d5:02 (90:fd:73:87:d5:02)
    - Source: IntelCor\_22:d5:ad (50:eb:71:22:d5:ad)
    - Type: IPv4 (0x0800)
  - Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: 100.100.1.1 (100.100.1.1)
  - User Datagram Protocol, Src Port: 51703, Dst Port: 53
  - Domain Name System (query)
- Packet Bytes:**

```

0000  90 fd 73 87 d5 02 50 eb 71 22 d5 ad 08 00 45 00  ..s..P. q"....E.
0010  00 56 00 ea 00 00 80 11 00 00 c0 a8 01 a3 64 64  .V.....dd
0020  01 01 c9 f7 00 35 00 42 28 04 c9 17 01 00 00 01  .....5.B (.....
0030  00 00 00 00 00 00 10 63 6f 67 6e 69 74 6f 2d 69  ....c ognito-i
0040  64 65 6e 74 69 74 79 09 65 75 2d 77 65 73 74 2d  dentity. eu-west-

```

a) ¿Cuáles son las direcciones origen y destino? ¿A quién crees que corresponden?

```
> Frame 4766: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645008E21}
▼ Ethernet II, Src: IntelCor_22:d5:ad (50:eb:71:22:d5:ad), Dst: zte_87:d5:02 (90:fd:73:87:d5:02)
  > Destination: zte_87:d5:02 (90:fd:73:87:d5:02)
  > Source: IntelCor_22:d5:ad (50:eb:71:22:d5:ad)
    Type: IPv4 (0x0800)
  > Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: 100.100.1.1 (100.100.1.1)
  > User Datagram Protocol, Src Port: 51703, Dst Port: 53
  > Domain Name System (query)
```

Origen: es la dirección física o dirección MAC del transmisor de la trama. Son 48 bits diferentes para cualquier terminal de la red Ethernet.

Destino: es la dirección MAC del destinatario especificada de la misma manera que la dirección de origen. En este caso, tenemos tres tipos de direcciones posibles: unicast, multicast y broadcast.

b) ¿Quién asigna estas direcciones?

La dirección MAC la asigna el fabricante en el Firmware de la tarjeta de red.

c) ¿Qué significa el campo tipo?

Indica de qué tipo es el contenido del campo de datos que lleva la trama. En este caso es IPv4.

d) ¿Qué función tiene el campo CRC?

Es un código de redundancia cíclica de 32 bits para la detección de errores. Abarca toda la trama a excepción del preámbulo. Las tramas que no tienen un CRC correcto se ignoran.

e) ¿Para qué sirve el preámbulo en una trama Ethernet?

Está formado por 64 bits, alternativamente 0 y 1. Los dos últimos son 11. Esto genera una señal cuadrada que permite a los terminales sincronizar de manera adecuada los relojes de sincronismo de bit. Los dos últimos bits señalan dónde empieza la trama (sincronismo de trama). Su forma es idéntica en todas las tramas. Nosotros obviaremos su presencia en el resto de la explicación, ya que sólo son una señal para marcar el inicio de la trama.

f) ¿Qué datos lleva esta trama Ethernet?

Lleva la suma de comprobación, información de control, incluye direcciones y protocolos.

2. En la trama anterior, fíjate que también incluye el detalle de una cabecera IP. Muestra mediante una captura de pantalla los contenidos de la cabecera IP. Responde a las siguientes cuestiones:

a) ¿Cuáles son las direcciones IP origen y destino? ¿A quién crees que corresponden?

```
> Frame 4766: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645008E}
> Ethernet II, Src: IntelCor_22:d5:ad (50:eb:71:22:d5:ad), Dst: zte_87:d5:02 (90:fd:73:87:d5:02)
> Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: 100.100.1.1 (100.100.1.1)
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 86
    Identification: 0x00ea (234)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.163 (192.168.1.163)
    Destination Address: 100.100.1.1 (100.100.1.1)
  > User Datagram Protocol, Src Port: 51703, Dst Port: 53
  > Domain Name System (query)
```

“Source” le corresponde a la IP de la maquina donde se realizó la petición y “Destination” es la IP a la maquina donde queremos enviar los recursos.

b) ¿Por qué el paquete lleva un identificador?

El identificador es un numero de serie del paquete, si un paquete se parte en pedazos mas pequeños (se fragmenta) por el camino, cada uno de los fragmentos llevara el mismo numero de identificación.

c) ¿Qué *flags* están activos en este paquete?

```
> Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: 100.100.1.1 (100.100.1.1)
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 86
    Identification: 0x00ea (234)
  > Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  Fragment Offset: 0
  Time to Live: 128
  Protocol: UDP (17)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 192.168.1.163 (192.168.1.163)
  Destination Address: 100.100.1.1 (100.100.1.1)
  > User Datagram Protocol, Src Port: 51703, Dst Port: 53
  > Domain Name System (query)
```

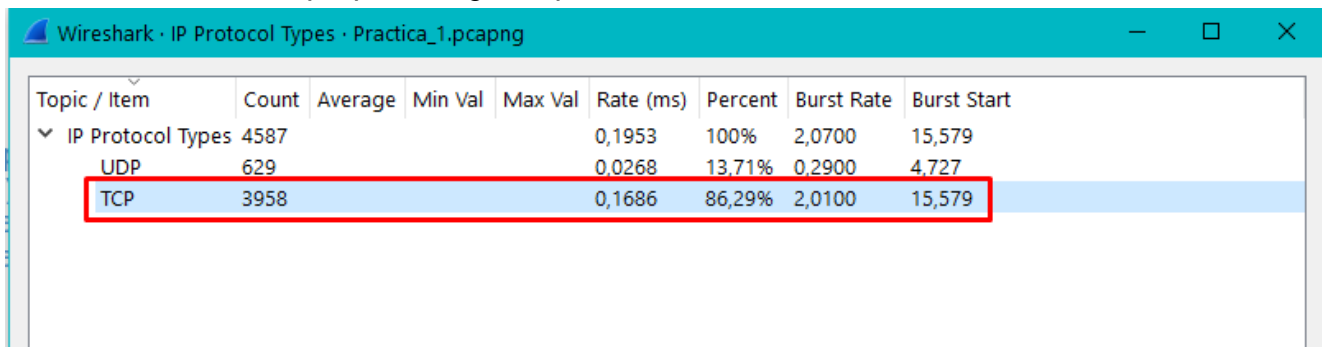
d) Explica qué significa el valor TTL del paquete analizado

Define el tiempo de que dispone el datagrama para llegar a nuestro destino, con el fin de evitar la existencia de datagrama que, por errores en el encaminamiento, estén dando vueltas indefinidamente en la red.

e) ¿Por qué es necesario un campo de *checksum* de nuevo en el protocolo IP?

En este campo se almacena un checksum de los campos de la cabecera. Es un mecanismo simple para detectar posibles errores en los campos de la cabecera del datagrama, los cuales podrían provocar situaciones

- f) Id al menú *Statistics > IPv4 statistics > IP protocol types*. ¿De qué protocolo se han enviado más paquetes? ¿Por qué?



The image shows the 'IP Protocol Types' statistics window in Wireshark. The table lists the count, average, and rate for various IP protocols. TCP is highlighted with a red box, indicating it is the most frequent protocol.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
IP Protocol Types	4587				0,1953	100%	2,0700	15,579
UDP	629				0,0268	13,71%	0,2900	4,727
TCP	3958				0,1686	86,29%	2,0100	15,579

Se han enviado mas paquetes de TCP ya que es mas seguro que el UDP

3. Finalmente, sin ningún filtro aplicado, iremos al menú de estadísticas *Statistics > Protocol Hierarchy*. Muestra en una captura de pantalla los resultados y coméntalos, relacionándolos con el proceso de encapsulamiento y des encapsulamiento de paquetes, pilar fundamental de las comunicaciones en las redes.

Protocolo	Porcentaje de paquetes	Paquetes	Porcentaje de bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	4859	100.0	3330594	1119k	0	0	0
Ethernet	100.0	4859	2.0	68026	22k	0	0	0
Internet Protocol Version 6	5.3	256	0.3	10240	3442	0	0	0
User Datagram Protocol	4.1	197	0.0	1576	529	0	0	0
Domain Name System	4.1	197	0.2	7913	2660	197	7913	2660
Transmission Control Protocol	1.0	49	0.0	1568	527	49	1568	527
Internet Control Message Protocol v6	0.2	10	0.0	456	153	10	456	153
Internet Protocol Version 4	94.4	4587	2.8	91740	30k	0	0	0
User Datagram Protocol	12.9	629	0.2	5032	1691	0	0	0
Simple Service Discovery Protocol	2.7	132	1.1	37164	12k	132	37164	12k
Multicast Domain Name System	1.0	49	0.3	10267	3451	49	10267	3451
Domain Name System	9.1	442	1.0	32359	10k	442	32359	10k
Data	0.1	6	0.0	294	98	6	294	98
Transmission Control Protocol	81.5	3958	91.9	3060303	1028k	2201	1425061	479k
VSS Monitoring Ethernet trailer	7.9	386	0.0	720	242	386	720	242
Transport Layer Security	26.5	1290	63.0	2097160	705k	1208	1970089	662k
Malformed Packet	0.0	1	0.0	0	0	1	0	0
Hypertext Transfer Protocol	3.4	166	27.8	927405	311k	77	230655	77k
Online Certificate Status Protocol	0.7	34	0.3	9678	3253	34	10123	3403
Media Type	0.1	6	7.7	258120	86k	6	219350	73k
Line-based text data	0.2	12	0.9	29186	9811	12	31491	10k
JPEG File Interchange Format	0.0	1	0.0	664	223	1	664	223
JavaScript Object Notation	0.7	34	11.6	384806	129k	21	348870	117k
eXtensible Markup Language	0.0	2	0.0	1608	540	2	1997	671
Data	0.2	9	0.0	9	3	9	9	3
HomePlug AV protocol	0.0	2	0.0	92	30	2	92	30
Address Resolution Protocol	0.3	14	0.0	410	137	14	410	137

- Protocolo: Nombre de este protocolo
- Porcentaje de paquetes: El porcentaje de paquetes de protocolo en relación con todos los paquetes en la captura.
- Paquetes: El número total de paquetes de este protocolo
- Porcentaje de Byte: El porcentaje de bytes de protocolo en relación con el total de bytes en la captura.
- Bytes: El número total de bytes de este protocolo.
- Bits/s: El ancho de banda de este protocolo en relación con el tiempo de captura.
- Paquetes finales: El número absoluto de paquetes de este protocolo donde fue el protocolo más alto en la pila (última disección).
- Bytes finales: El número absoluto de bytes de este protocolo donde era el protocolo más alto en la pila (última disección).
- Bits finales/s: El ancho de banda de este protocolo en relación con el tiempo de captura donde fue el protocolo más alto en la pila (última disección).

Los paquetes generalmente contienen múltiples protocolos.



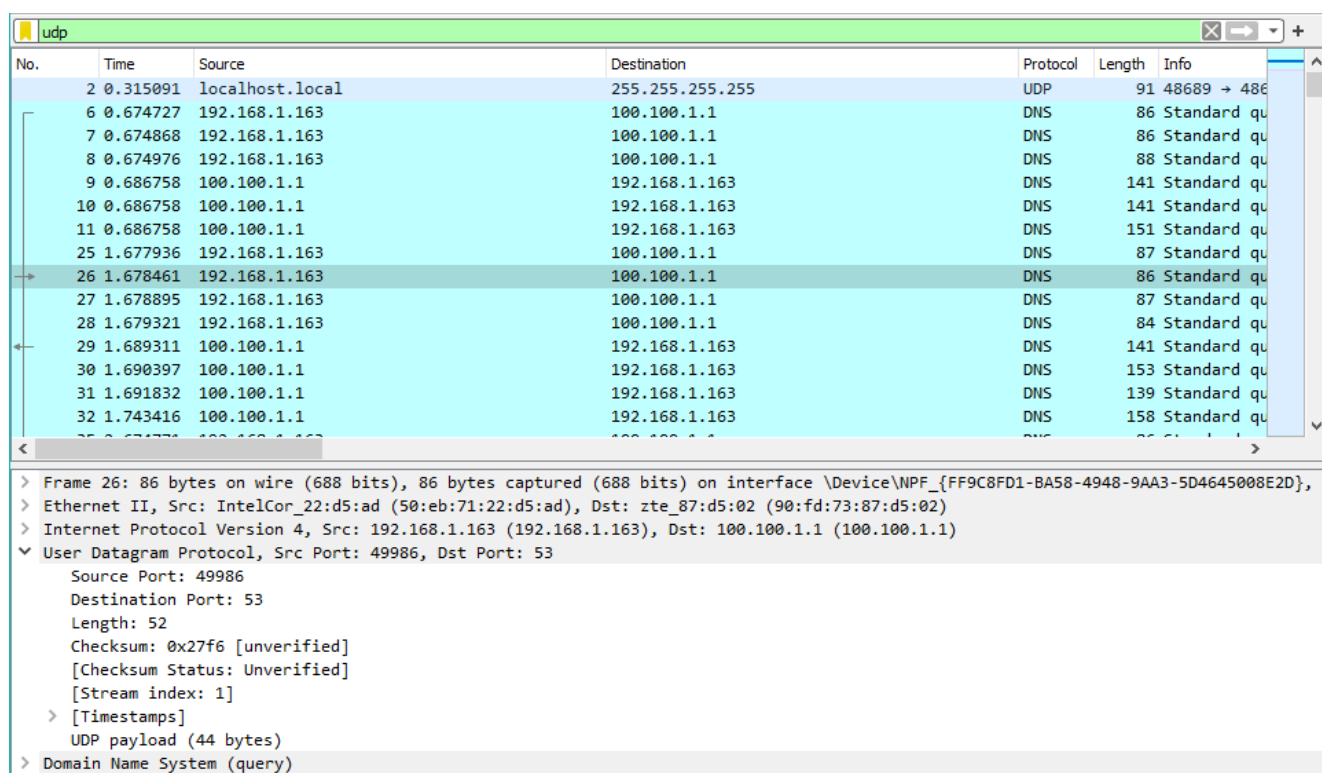
## Segunda parte (nota máxima: C+): El nivel de transporte

En esta segunda parte de la práctica profundizaremos en el siguiente nivel de la pila de Internet: el nivel de transporte (*Transport Layer*). Dos protocolos ofrecen servicios diferentes en este nivel: UDP (*User Datagram Protocol*) y TCP (*Transport Control Protocol*). Podéis trabajar sobre la misma captura de paquetes que habéis realizado en la primera parte de la práctica.

Como material de soporte para realizar esta segunda parte de la práctica os proponemos los siguientes apartados del libro Computer Networking (7a edición), especialmente los apartados relacionados con las cabeceras de los protocolos:

- **3.3 Connectionless Transport: UDP** pag.232
  - **3.3.1 UDP Segment Structure**
- **3.5 Connection-Oriented Transport: TCP** pag.264-269
  - **3.5.2 TCP Segment Structure**

1. En primer lugar, analizaremos el protocolo **UDP**. Podéis ponerlo en el filtro del Wireshark y así sólo veréis los paquetes de este protocolo. Muestra mediante una captura de pantalla los campos de la cabecera y responde las siguientes cuestiones:



- a) ¿Qué protocolo/s del nivel de aplicación ha/n generado estos paquetes?

Se ha generado a través del protocolo de DNS.

- b) ¿Por qué crees que se utiliza este protocolo en lugar de TCP?

El protocolo UDP es mucho más rápido que el TCP, ya que los datos se envían sin haber establecido previamente una conexión entre origen y destino, lo que provoca que esto sea sin confirmación por parte de la otra máquina. TCP si que es un protocolo de conexión, envía los paquetes y se queda esperando el ACK para asegurar de que se han recibido.



c) ¿Qué valor tiene el puerto origen y el puerto destino? ¿Qué significa?

El puerto de origen tiene 49986

d) ¿Por qué es necesario el campo *checksum*?

El campo checksum (16 bits) es opcional y protege tanto la cabecera como los datos UDP (se debe recordar que el checksum del datagrama IP sólo cubre la cabecera IP). Cuando el UDP recibe un datagrama y determina que hay errores, lo descarta y no lo entrega a ninguna aplicación.

e) ¿Cómo se calcula?

Es relativamente sencillo:

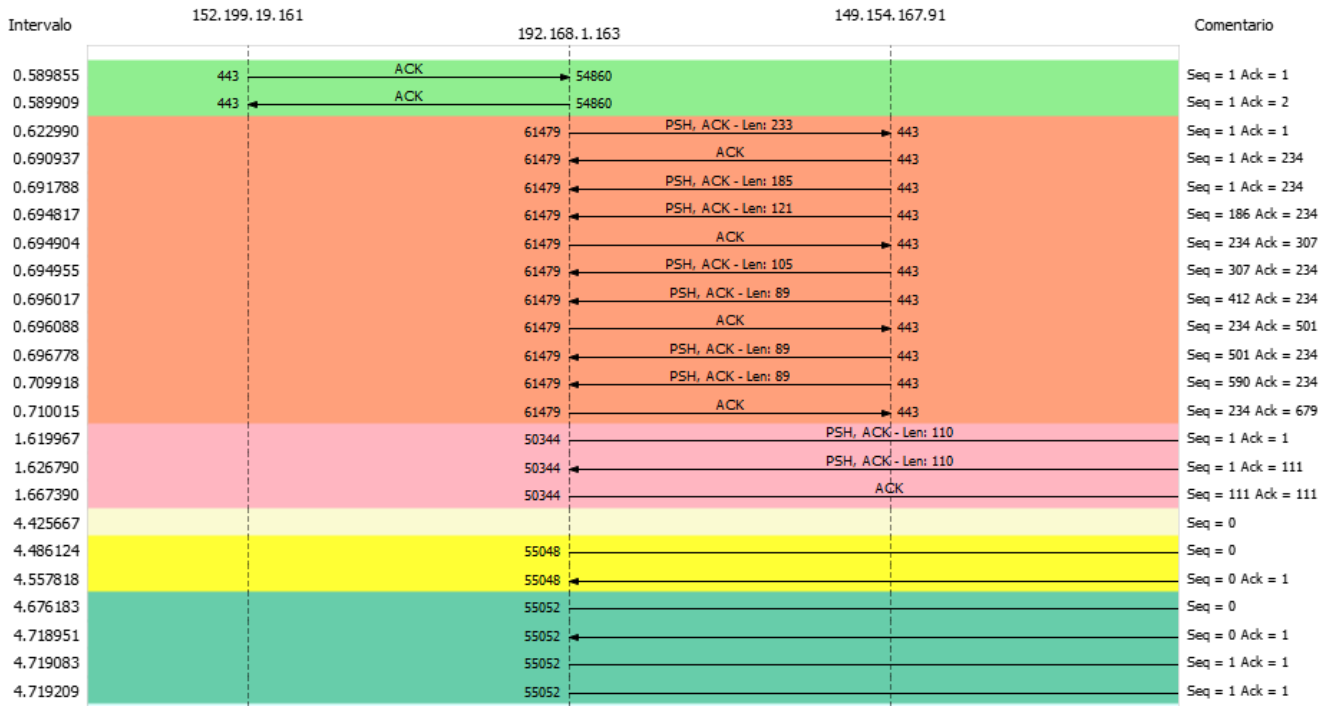
- Tomamos los dígitos hexadecimales según explico más abajo
- Los pasamos a decimal
- Los sumamos
- Calculamos el resto a 256
- Restamos el resto a 256
- Convertimos el número decimal que nos da a hexadecimal, y ese es el checksum.

f) ¿Qué pone en el campo longitud? Comprueba con los valores hexadecimales del paquete UDP que realmente es así.

```
> Frame 26: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645008E2D}, i
> Ethernet II, Src: IntelCor_22:d5:ad (50:eb:71:22:d5:ad), Dst: zte_87:d5:02 (90:fd:73:87:d5:02)
> Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: 100.100.1.1 (100.100.1.1)
< User Datagram Protocol, Src Port: 49986, Dst Port: 53
  Source Port: 49986
  Destination Port: 53
  Length: 52
  Checksum: 0x27f6 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 1]
  > [Timestamps]
  UDP payload (44 bytes)
  > Domain Name System (query)
<
0000  90 fd 73 87 d5 02 50 eb 71 22 d5 ad 08 00 45 00  ..s...P..q"....E.
0010  00 48 00 18 00 00 80 11 00 00 c0 a8 01 a3 64 64  .H.....dd
0020  01 01 c3 42 00 35 00 34 27 f6 32 98 01 00 00 01  ...B-5-4'2....
0030  00 00 00 00 00 00 03 31 36 33 01 31 03 31 36 38  ....163.1.168
0040  03 31 39 32 07 69 6e 2d 61 64 64 72 04 61 72 70  .192.in-addr.arp
0050  61 00 00 0c 00 01  a.....
```

Como se puede observar nos dan una longitud de 52, dicho valor lo pasamos Hexadecimal es un 34, tal y como muestra la captura.

2. A continuación, veremos cómo funciona el protocolo de transporte **TCP**. Iremos al menú de estadísticas *Statistics > Flow Graph*, y seleccionaremos sólo el tipo TCP, poniendo *Flow type > TCP flow*. Muestra una captura de pantalla donde se observen las fases de establecimiento y fin de la comunicación. Explica el proceso. ¿Para qué sirve el flag PSH?



PSH es un bit que se encuentra en el campo del código en el protocolo TCP . Cuando PSH está activado indica que los datos de ese segmento y los datos que hayan sido almacenados anteriormente en el buffer del receptor deben ser transferidos a la aplicación receptora lo antes posible. A veces llegan varios segmentos que transportan datos y no tienen activado el bit PSH; el receptor almacenará esos datos, pero no los entregará a la aplicación receptora hasta que reciba un segmento con el PSH activado.

Con el bit a 1 está activado, y a 0 desactivado.

- Iremos al listado principal de paquetes. Podéis filtrar por el protocolo TCP para ver sólo los paquetes referentes a este protocolo. Selecciona un paquete y muestra mediante una captura de pantalla los contenidos de la cabecera TCP. Responde a las siguientes cuestiones:

```
> Frame 24: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645008E2D},
> Ethernet II, Src: IntelCor_22:d5:ad (50:eb:71:22:d5:ad), Dst: Google_1f:0d:15 (3c:8d:20:1f:0d:15)
> Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: 36319056-abf8-7d19-1bde-48c0d8d2541e.local (192.168.1.131)
▼ Transmission Control Protocol, Src Port: 50344, Dst Port: 8009, Seq: 111, Ack: 111, Len: 0
  Source Port: 50344
  Destination Port: 8009
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence Number: 111 (relative sequence number)
  Sequence Number (raw): 2713734399
  [Next Sequence Number: 111 (relative sequence number)]
  Acknowledgment Number: 111 (relative ack number)
  Acknowledgment number (raw): 644370036
  0101 .... = Header Length: 20 bytes (5)
> Flags: 0x010 (ACK)
  Window: 508
  [Calculated window size: 508]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x8491 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
▼ [SEQ/ACK analysis]
  [This is an ACK to the segment in frame: 23]
  [The RTT to ACK the segment was: 0.040600000 seconds]
> [Timestamps]
```

```
0000 3c 8d 20 1f 0d 15 50 eb 71 22 d5 ad 08 00 45 00 <...P. q"....E.
0010 00 28 22 5c 40 00 80 06 00 00 c0 a8 01 a3 c0 a8 .(" @... ..
0020 01 83 c4 a8 1f 49 a1 c0 4c ff 26 68 4e 74 50 10 .....I.. L.&hNtP.
0030 01 fc 84 91 00 00 .....
```

- ¿Cuál es el número de secuencia? ¿Para qué sirve?

El numero de secuencia es 111 y sirve para identificar el byte del flujo de datos enviados por el emisor TCP al receptor TCP que representa el primer byte de datos del segmento.

- ¿Y el número de ACK?

El TCP reconoce datos mediante la técnica de piggybacking. Al activar un bit de la cabecera (el bit ACK), el TCP tiene en cuenta el número de secuencia ACK que indica al otro extremo TCP el próximo byte que está dispuesto a recibir. Dicho de otro modo, el número ACK menos uno indica el último byte reconocido.

- ¿Qué valor tiene el puerto origen y el puerto destino? ¿Qué significa?

Source: 50344

Destination: 8009

Que identifican las aplicaciones en los terminales de origen y de destino.

d) ¿Qué *flags* están activos en este paquete?

```
0101 .... = Header Length: 20 bytes (5)
▼ Flags: 0x010 (ACK)
 000. .... = Reserved: Not set
 ...0 .... = Nonce: Not set
 .... 0... = Congestion Window Reduced (CWR): Not set
 .... 0... = ECN-Echo: Not set
 .... 0... = Urgent: Not set
 .... 1... = Acknowledgment: Set
 .... 0... = Push: Not set
 .... 0... = Reset: Not set
 .... 0... = Syn: Not set
 .... 0... = Fin: Not set
 [TCP Flags: .....A....]
```

Esta activado el ACK.

e) Explica qué significan los valores relativos a la ventana.

Indica cuántos bits componen la ventana de transmisión del protocolo de control de flujo por ventana deslizante. A diferencia de los protocolos del nivel de enlace, en los cuales la ventana era constante y contaba tramas, en el TCP la ventana es variable y cuenta bytes.

Con cada segmento transmitido, un extremo TCP advierte el otro extremo de la cantidad de datos que está dispuesto a recibir en cada momento. De este modo, el extremo que recibe un segmento actualiza el tamaño de su ventana de transmisión.

f) ¿Qué datos lleva este paquete TCP?

Un Tcp lleva la cabecera , el establecimiento de conexión y el OK de recepción.

## Tercera parte (nota máxima: B): El nivel de aplicación

Continuando con la estructura de la pila de Internet, el último nivel es el que interactúa con las aplicaciones que ejecuta el usuario, directa o indirectamente: el nivel de Aplicación (*Application Layer*). En este nivel encontraremos una multitud de protocolos, pero, en esta parte nos fijaremos en dos de ellos: DNS (*Domain Name Server*) y HTTP (*HyperText Transfer Protocol*).

Como material de soporte para realizar esta primera parte de la práctica os proponemos los siguientes apartados del libro *Computer Networking* (7a edición), especialmente los apartados relacionados con las cabeceras de los protocolos:

- **2.4 DNS—The Internet's Directory Service** pag. 163-165
  - **2.4.3 DNS Records and Messages**
- **2.2 The Web and HTTP** pag.131-138
  - **2.2.3 HTTP Message Format**

1. En primer lugar, analizaremos una cabecera **DNS** de una petición y su respuesta, por lo que podéis poner DNS en el filtro del Wireshark y así sólo veréis los paquetes de este protocolo. Muestra mediante una captura de pantalla los campos de las cabeceras y responde las siguientes cuestiones:

The screenshot displays a Wireshark capture of a DNS query. The packet list at the top shows frame 56 as a DNS query from 100.100.1.1 to 192.168.1.163. The details pane below shows the structure of the DNS query:

- Frame 56: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface \Device\NPF\_{FF9C8FD1-BA58-4948-9AA3-5D4645008E2D}
- Ethernet II, Src: IntelCor\_22:d5:ad (50:eb:71:22:d5:ad), Dst: zte\_87:d5:02 (90:fd:73:87:d5:02)
- Internet Protocol Version 6, Src: 2a0c:5a84:3301:6900:4d1b:da6e:8e80:a428 (2a0c:5a84:3301:6900:4d1b:da6e:8e80:a428), Dst: 2a0c:5a80:0:2::1
- User Datagram Protocol, Src Port: 60071, Dst Port: 53
- Domain Name System (query)
  - Transaction ID: 0x4f4c
  - Flags: 0x0100 Standard query
    - 0... .. = Response: Message is a query
    - .000 0... .. = Opcode: Standard query (0)
    - ... ..0. ... = Truncated: Message is not truncated
    - ... ..1 ... = Recursion desired: Do query recursively
    - ... ..0... .. = Z: reserved (0)
    - ... ..0 ... = Non-authenticated data: Unacceptable
  - Questions: 1
  - Answer RRs: 0
  - Authority RRs: 0
  - Additional RRs: 0
  - Queries

a) ¿Qué puertos origen y destino se utilizan?

```

Internet Protocol version 6, Src: 2a0c:3a84:3301:6900:4010:0a0e:0e00:a420
User Datagram Protocol, Src Port: 60071, Dst Port: 53
  Source Port: 60071
  Destination Port: 53
  Length: 37
  Checksum: 0xff8a [unverified]
  [Checksum Status: Unverified]
  [Stream index: 6]
  > [Timestamps]
    UDP payload (29 bytes)
  > Domain Name System (query)
    
```

b) ¿Para qué sirve el identificador de transacción?

Este identificador se copia en la respuesta correspondiente del servidor de nombres y se puede usar para diferenciar respuestas cuando concurren múltiples consultas. 0x8eea

c) ¿Qué indican los flags?

```

Domain Name System (query)
  Transaction ID: 0x4f4c
  Flags: 0x0100 Standard query
    0... .. = Response: Message is a query
    .000 0... .. = Opcode: Standard query (0)
    ....0. .... = Truncated: Message is not truncated
    ....1 .... = Recursion desired: Do query recursively
    ....0.. .... = Z: reserved (0)
    ....0 .... = Non-authenticated data: Unacceptable
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  > Queries
    
```

Response: Flag que indica consulta (0) o respuesta (1). Esta captura es 0 pregunta.

Opcode: Campo de 4 bits que especifica el tipo de consulta. 0 es Standard query.

Truncated: Activo si el mensaje es mas largo de los que permite la línea de transmisión,

Recursion desired: El bit se copia en la respuesta e indica al servidor de nombres una resolución recursiva.

Z: Son 3 bits reservado para su futuro, que siempre tiene que ser 0.

d) En la petición, ¿qué dominio está preguntando?

```
Authority RRs: 0
Additional RRs: 0
▼ Queries
  ▼ www.uoc.edu: type AAAA, class IN
    Name: www.uoc.edu
    [Name Length: 11]
    [Label Count: 3]
    Type: AAAA (IPv6 Address) (28)
    Class: IN (0x0001)
```

En la petición estamos preguntando por el dominio www.uoc.edu

e) ¿De qué tipo y clase es este dominio? ¿Qué quiere decir?

```
Authority RRs: 0
Additional RRs: 0
▼ Queries
  ▼ www.uoc.edu: type AAAA, class IN
    Name: www.uoc.edu
    [Name Length: 11]
    [Label Count: 3]
    Type: AAAA (IPv6 Address) (28)
    Class: IN (0x0001)
```

El dominio que tenemos es de tipo AAAA

Registros AAAA o "IPv6 Address": Estos son los que asignan las direcciones IPv6 a un dominio o subdominio; pero para este último uso, se suelen utilizar los registros CNAME.

Primero está el nombre completo del dominio (incluyendo el punto al final), "IN" corresponde a la clase del registro, "AAAA" sería el tipo del registro y finalmente la IP a la que apunta.

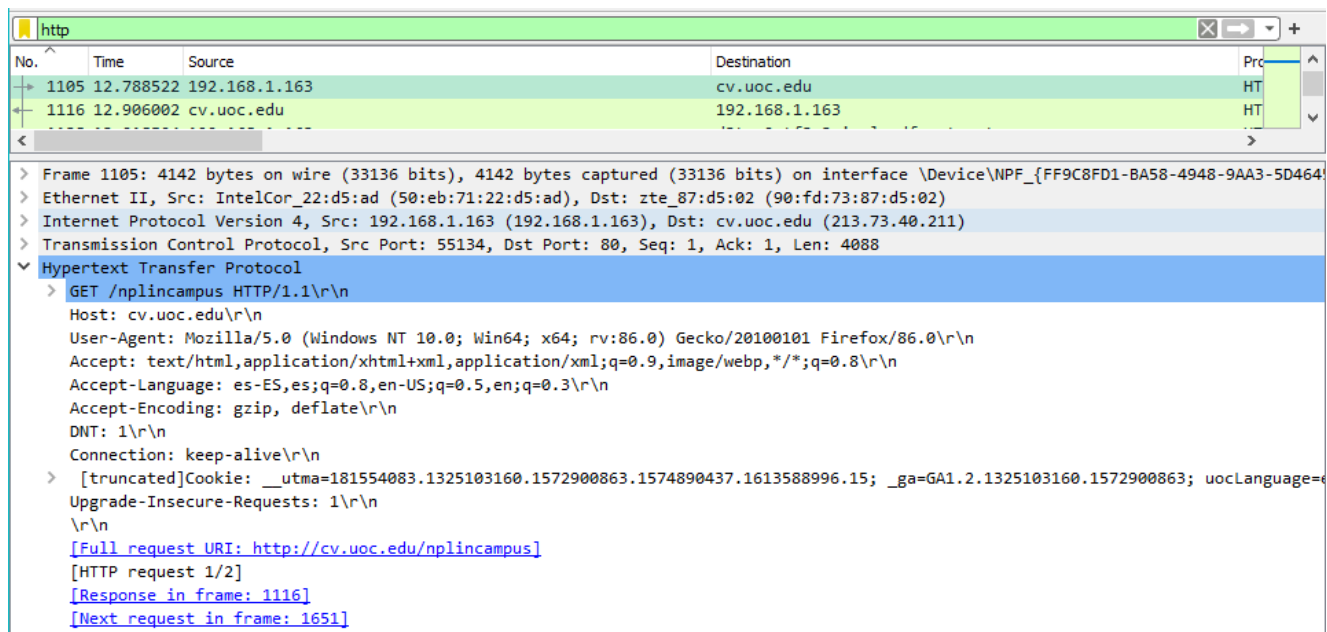
f) ¿Puede haber varios registros (*RR*, *resource records*) en la respuesta?

```
.... .... ...0 .... = Non-a
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
▼ Queries
```

Puede haber diferentes registros en la respuesta.



2. Ahora analizaremos una cabecera **HTTP** de una petición GET. Muestra mediante una captura de pantalla los campos de la cabecera y responde las siguientes cuestiones:



a) ¿Qué puertos origen y destino se utilizan?

> Transmission Control Protocol, Src Port: 55158, Dst Port: 80, Seq: 5067, Ack: 1375, Len: 4837

El puerto de origen es 55134 y el de destino el 80

b) ¿Qué versión del protocolo HTTP?

```

< Hypertext Transfer Protocol
  > GET /nplincampus HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /nplincampus HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /nplincampus
    Request Version: HTTP/1.1
  
```

Se utiliza la versión HTTP/1.1, es la versión más utilizada actualmente las conexiones persistentes están activadas por defecto y funcionan bien con los Proxys. (Tanto en la petición como la respuesta tenemos la misma versión).

c) ¿Se utilizan conexiones persistentes?

```

Hypertext Transfer Protocol
> GET /nplincampus HTTP/1.1\r\n
Host: cv.uoc.edu\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n
Accept-Encoding: gzip, deflate\r\n
DNT: 1\r\n
Connection: keep-alive\r\n
[truncated]Cookie: __utma=181554083.1325103160.1572900863.1574890437.1613588996.15; __ga=GA1.2.1325103160.1572900863; uocLang
Upgrade-Insecure-Requests: 1\r\n
    
```

Vemos que tenemos Connection: keep-alive a parte anteriormente hemos dicho que es HTTP/1.1 la conexión es persistente por defecto a menos que añada la cabecera "Conexión: cerrar" a la solicitud HTTP. Por lo tanto, vemos que se utiliza una conexión persistente. (Tanto en la petición como la respuesta tenemos la misma versión).

d) ¿Qué idioma se utiliza? ¿Por qué se facilita?

```

Hypertext Transfer Protocol
> GET /nplincampus HTTP/1.1\r\n
Host: cv.uoc.edu\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n
Accept-Encoding: gzip, deflate\r\n
DNT: 1\r\n
    
```

En este caso se utiliza los idiomas es-ES, en-US.

Este campo informa qué idiomas puede comprender el cliente y qué variante de configuración regional se prefiere.

Este es una pista que se utilizará cuando el servidor no tenga forma de determinar el idioma de otra forma, como una URL específica, que esté controlada por una decisión explícita del usuario.

e) ¿Qué tipo de contenidos podrá procesar?

```

1138 13.018504 192.168.1.163 d3toq2etf3p3xi.cloudfront.net HT
1145 13.089014 d3toq2etf3p3xi.cloudfront.net 192.168.1.163 HT
1147 13.103315 192.168.1.163 d3toq2etf3p3xi.cloudfront.net HT
<
>
> Frame 1145: 247 bytes on wire (1976 bits), 247 bytes captured (1976 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645008
> Ethernet II, Src: zte_87:d5:02 (90:fd:73:87:d5:02), Dst: IntelCor_22:d5:ad (50:eb:71:22:d5:ad)
> Internet Protocol Version 4, Src: d3toq2etf3p3xi.cloudfront.net (13.33.234.125), Dst: 192.168.1.163 (192.168.1.163)
> Transmission Control Protocol, Src Port: 80, Dst Port: 55135, Seq: 2901, Ack: 3088, Len: 193
> [3 Reassembled TCP Segments (3093 bytes): #1143(1450), #1144(1450), #1145(193)]
Hypertext Transfer Protocol
> HTTP/1.1 200 \r\n
Content-Type: text/html;charset=UTF-8\r\n
Transfer-Encoding: chunked\r\n
Connection: keep-alive\r\n
Date: Sun, 28 Feb 2021 10:04:38 GMT\r\n
Set-Cookie: AWSALB=V2HuPe8RsPoM2G41H1s9PhqoYfSaAyotyIH/0wWliw/eXU4i+V7nszn4yru6nwrPLMeqRv2+F2noTNeAPPyfAM0aIBpCHvCUwpJyoxSKHwOqZCX:
Set-Cookie: AWSALBCORS=V2HuPe8RsPoM2G41H1s9PhqoYfSaAyotyIH/0wWliw/eXU4i+V7nszn4yru6nwrPLMeqRv2+F2noTNeAPPyfAM0aIBpCHvCUwpJyoxSKHwO:
    
```

Podremos procesar contenido de tipo txt/html como bien nos indica el content-Type

f) ¿Qué formatos de codificación?

```

1138 13.018504 192.168.1.163 d3toq2etf3p3xi.cloudfront.net HT
1145 13.089014 d3toq2etf3p3xi.cloudfront.net 192.168.1.163 HT
1147 13.103315 192.168.1.163 d3toq2etf3p3xi.cloudfront.net HT
< >
> Frame 1145: 247 bytes on wire (1976 bits), 247 bytes captured (1976 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645008
> Ethernet II, Src: zte_87:d5:02 (90:fd:73:87:d5:02), Dst: IntelCor_22:d5:ad (50:eb:71:22:d5:ad)
> Internet Protocol Version 4, Src: d3toq2etf3p3xi.cloudfront.net (13.33.234.125), Dst: 192.168.1.163 (192.168.1.163)
> Transmission Control Protocol, Src Port: 80, Dst Port: 55135, Seq: 2901, Ack: 3088, Len: 193
> [3 Reassembled TCP Segments (3093 bytes): #1143(1450), #1144(1450), #1145(193)]
> Hypertext Transfer Protocol
> HTTP/1.1 200 \r\n
Content-Type: text/html; charset=UTF-8\r\n
Transfer-Encoding: chunked\r\n
Connection: keep-alive\r\n
Date: Sun, 28 Feb 2021 10:04:38 GMT\r\n
Set-Cookie: AWSALB=V2HuPe8RsPoM2G4lH1s9PhqoYfsaAyotyIH/0wWliW/eXU4i+V7nszn4yru6nwrPLMeqRv2+F2noTNeAPPyfAM0aIBpCHvCUwpJyoxSKHwOqZCX:
Set-Cookie: AWSALBCORS=V2HuPe8RsPoM2G4lH1s9PhqoYfsaAyotyIH/0wWliW/eXU4i+V7nszn4yru6nwrPLMeqRv2+F2noTNeAPPyfAM0aIBpCHvCUwpJyoxSKHwOq

```

El charset nos permite los formatos de codificación UTF-8

3. Nos centraremos en una cabecera HTTP de una respuesta exitosa a una petición GET. Muestra mediante una captura de pantalla los campos de la cabecera y responde las siguientes cuestiones:

a) ¿Qué contiene esta respuesta?

[illegible]

### Línea de estatus:

La primera línea de una respuesta tiene siempre un formato como HTTP/1. El primer campo indica el protocolo HTTP usado. El segundo campo es un valor numérico de tres dígitos que indica el tipo de respuesta dada. El último campo es un texto descriptivo del estatus.

### Cabeceras (Header):

La respuesta contiene todas las cabeceras que el servidor considere oportuno incluir.

**CRLF:**

Una línea en blanco separa las cabeceras del cuerpo de la respuesta.

### Cuerpo (Body):

Esta parte contiene el contenido "real" de la respuesta propiamente dicho. Así si por ejemplo se ha solicitado una página web, el contenido a mostrar se encuentra aquí.

b) ¿Qué tipo de contenido es?

La respuesta de la cabecera HTTP contiene un contingente HTML como ya hemos dicho en anterioridad, contiene el cuerpo de la respuesta contiene el contingente real de la web.

c) ¿Está fragmentado el contenido? ¿Cómo lo sabemos?

```
Identification: 0x669d (26269)
▼ Flags: 0x40, Don't fragment
  0... .... = Reserved bit: Not set
  .1.. .... = Don't fragment: Set
  ..0. .... = More fragments: Not set
Fragment Offset: 0
Time to Live: 128
```

d) ¿Qué campos relacionados con fechas hay? ¿Qué significan?

```
HTTP/1.1 200 OK
Date: Sun, 28 Feb 2021 10:04:40 GMT
Server: Apache
Last-Modified: Thu, 07 Feb 2019 13:36:18 GMT
ETag: "181e-5814ded583880"
Accept-Ranges: bytes
```

El campo date nos indica la fecha en la que nos ha respondido el servidor.



e) ¿Se utilizan *cookies*? ¿Cómo funcionan?

[illegible]

Una cookie es una pequeña pieza de datos que un servidor envía al navegador web del usuario. El navegador guarda estos datos y los envía de regreso junto con la nueva petición en el mismo servidor. Las cookies se utilizan para decirle al servidor que dos peticiones tienen su origen en el mismo navegador web lo que permite, por ejemplo, mantener la sesión de un usuario abierta.

f) Comenta qué significan los campos relacionados con *cachés*.

```
HTTP/1.0 200 OK
Cache-Control: No-Cache
Pragma: No-Cache
Server: BigIP
Connection: Keep-Alive
Content-Length: 7530
```

Los campos relacionados con caches se utilizan normalmente para especificar directivas para el mecanismo de almacenaje tanto en solicitudes como en respuestas.

En mi caso tenemos Cache Control: No-Cache



4. Siguiendo con el estudio del protocolo HTTP, iremos al menú de estadísticas del Wireshark *Statistics > HTTP > Packet counter*. Muestra una captura de pantalla donde se vean las respuestas no exitosas que se han recibido (tienen el valor *Count > 0*). Describe el significado de cada código de error y cuántos paquetes se han dado de este tipo. Después, elige uno de los códigos de error y localízalo en la pantalla principal de

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ Total HTTP Packets	298				0,0155	100%	0,1300	15,606
Other HTTP Packets	0				0,0000	0,00%	-	-
▼ HTTP Response Packets	83				0,0043	27,85%	0,0700	15,622
??? : broken	0				0,0000	0,00%	-	-
5xx: Server Error	0				0,0000	0,00%	-	-
▼ 4xx: Client Error	1				0,0001	1,20%	0,0100	13,849
404 Not Found	1				0,0001	100,00%	0,0100	13,849
▼ 3xx: Redirection	17				0,0009	20,48%	0,0400	13,380
304 Not Modified	9				0,0005	52,94%	0,0300	13,408
302 Found	7				0,0004	41,18%	0,0200	21,110
301 Moved Permanently	1				0,0001	5,88%	0,0100	8,979
▼ 2xx: Success	65				0,0034	78,31%	0,0500	15,428
200 OK	65				0,0034	100,00%	0,0500	15,428
1xx: Informational	0				0,0000	0,00%	-	-
▼ HTTP Request Packets	215				0,0112	72,15%	0,0900	15,052
SEARCH	8				0,0004	3,72%	0,0200	20,380
POST	18				0,0009	8,37%	0,0300	5,183
NOTIFY	124				0,0065	57,67%	0,0600	8,807
GET	65				0,0034	30,23%	0,0700	15,606

Los códigos de estado HTTP describen de forma abreviada la respuesta HTTP. El primer dígito del código de estado especifica uno de los 5 tipos de respuesta, el mínimo para que un cliente pueda trabajar con HTTP es que reconozca estas 5 clases.

Índice de contenido:

- 1XX Respuestas informativas que en nuestro caso tenemos 0
- 2XX Peticiones correctas que en nuestro caso tenemos 65
- 3XX Redirecciones que en nuestro caso tenemos 17
- 4XX Errores del cliente que en nuestro caso tenemos 1
- 5XX Errores de servidor que en nuestro caso tenemos 0

A parte de estos índices dentro de cada uno hay otros tipos diferentes como tendríamos en nuestro caso dentro del índice 3XX, tendríamos:

- 304: Este nos indica que no hay necesidad de retransmitir los recursos solicitados. En nuestro caso tenemos 9
- 302: Este nos indica que el recurso solicitado ha sido movido temporalmente a la URL dada por las cabeceras Location. En nuestro caso tenemos 7
- 301: Este nos indica que el host ha sido capaz de comunicarse con el servidor pero que el recurso solicitado ha sido movido a otra dirección permanentemente. En nuestro caso tenemos 1.

1319	13.799018	192.168.1.163	d3toqetf3p3xi.cloudfront.net	HTTP	3201	GET /webst/es/UOCSeif-Regular.48bf6d7fe5aac624fa.otf HTTP/1.1
1325	13.814459	d3toqetf3p3xi.cloudfront.net	192.168.1.163	HTTP/1.1	200	JavaScript Object Notation (application/json)
1339	13.818554	d3toqetf3p3xi.cloudfront.net	192.168.1.163	HTTP/1.1	200	JavaScript Object Notation (application/json)
1352	13.832663	d3toqetf3p3xi.cloudfront.net	192.168.1.163	HTTP	402	HTTP/1.1 304 Not Modified
1354	13.848772	d3toqetf3p3xi.cloudfront.net	192.168.1.163	HTTP/1.1	404	JavaScript Object Notation (application/json)
1396	13.897967	192.168.1.163	ocsp.ocalb.amazontrust.com	OCSP	403	Request
1409	14.014880	ocsp.ocalb.amazontrust.com	192.168.1.163	OCSP	1059	Response

Frame 1354: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{FF9C8FD1-8A58-4948-9AA3-5D4645008E2D}, id 0	
Ethernet II, Src: ste_07:d5:02 (08:fd:73:07:d5:02), Dst: IntelCor_22:d5:ad (58:eb:71:22:d5:ad)	
Internet Protocol Version 4, Src: d3toqetf3p3xi.cloudfront.net (13.33.234.161), Dst: 192.168.1.163 (192.168.1.163)	
0800 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 65 Identification: 0x997a (39290) Flags: 0x00 0... .... = Reserved bit: Not set 0... .... = Don't Fragment: Not set ..0. .... = More fragments: Not set Fragment Offset: 0 Time to Live: 240 Protocol: TCP (6) Header Checksum: 0x7142 [validation disabled] [Header checksum status: Unverified] Source Address: d3toqetf3p3xi.cloudfront.net (13.33.234.161) Destination Address: 192.168.1.163 (192.168.1.163)	
Transmission Control Protocol, Src Port: 80, Dst Port: 55145, Seq: 1057, Ack: 3072, Len: 5	
[2 Reassembled TCP Segments (1061 bytes): #1353(1056), #1354(5)]	
Hypertext Transfer Protocol	
JavaScript Object Notation: application/json	
VSS Monitoring Ethernet trailer, Source Port: 21	

## Buscamos el Error **404 Not Found**

Este es el detonante clásico del error 404 es que el contenido de la web ha sido eliminado o trasladado a otra URL. Sin embargo, hay otras razones por las que puede aparecer una página HTTP 404 en su navegador.

Otras de las razones más comunes podrían ser:

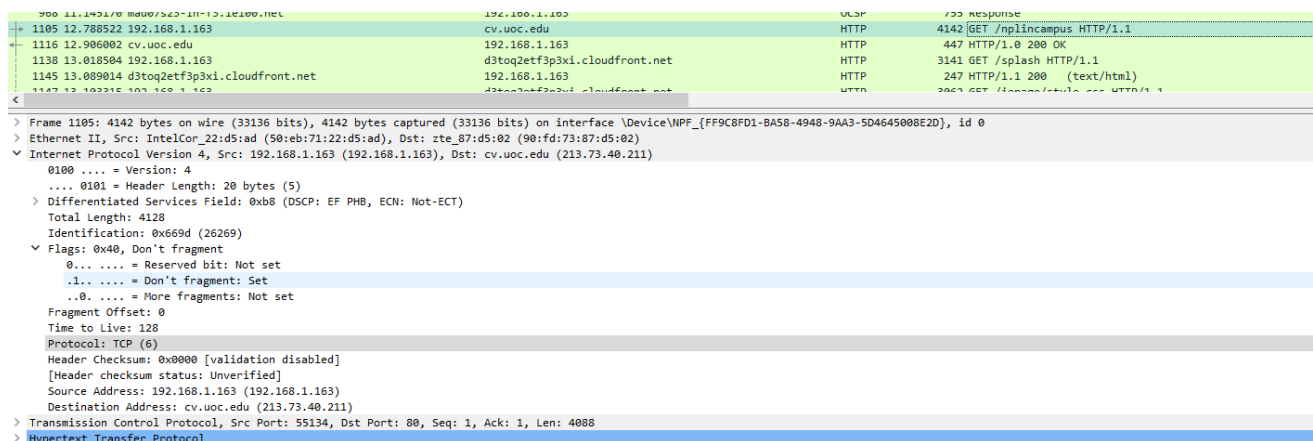
- El nombre de dominio no existe.
- La dirección URL o sus contenidos se han eliminado o cambiado.
- El servidor web responsable no está en funcionamiento o hay problemas de conexión.
- El nombre de dominio solicitado no puede ser convertido en una dirección IP para el Domain Name System.

El enlace no se colocó correctamente, la URL se enlazó de forma incorrecta o la dirección que el usuario introdujo en la barra del navegador no concuerda con el URL.

captura de paquetes. Muestra mediante una captura de pantalla los campos de la cabecera con una breve explicación.

## Cuarta parte (nota máxima: A): Seguridad en la red

Para realizar esta última parte del enunciado, nos fijaremos en la interacción que se da cuando ponemos nuestra contraseña en el campus. Para localizar los paquetes de esta parte, puedes usar la captura anterior o hacer otra nueva, centrándote explícitamente en el acceso al campus.



1. ¿Qué protocolo del nivel de aplicación se utiliza para ofrecer seguridad en el acceso al campus? Describe sus principales características, así como qué tipo de encriptación utiliza.

Se utiliza el protocolo HTTPS (HyperText Transfer Protocol Secure, Protocolo de transferencia de hipertexto), este es un protocolo de comunicación de Internet que protege la integridad y la confidencialidad de los datos de los usuarios entre sus ordenadores y el sitio web.

HTTPS utiliza un cifrado basado en SSL/TSL\* para crear un canal cifrado más apropiado para el tráfico de información sensible que el protocolo HTTP.

De este modo se consigue que la información sensible (user/password) no pueda ser utilizada por un atacante que hay conseguido interceptar la transferencia de datos de la conexión, ya que lo único que se obtendrá será un flujo de datos cifrados que resultara imposible descifrar. El puerto estándar para este protocolo es el 443.

Para entrar un poco más en materia explicare cómo funciona la encriptación que se utiliza. Principalmente para cifrar datos se necesita una clave.

Esa clave tendrá que saberla tanto el navegador como el servidor para poder comunicarse.

Con esto observamos a simple vista que tenemos un problema, es decir, ¿cómo compartimos una clave sin que nadie pueda saberla?

También se ha de dejar claro que la clase tienes que ser única, es decir, no se puede dejar de manera preconfigurada ni nada por el estilo.

Por ello se utiliza un sistema de clave pública/cave privada.

Cuando el cliente, contacta con el servidor web, este le envía primero su certificado.

Este certificado SSL prueba que el servidor es auténtico y no está simulando una identidad falsa.

El cliente comprueba la validez del certificado y envía al servidor un número aleatorio cifrado con la clave pública (public key) del servidor.

A partir de ese número aleatorio, el servidor genera la clave de sesión (session key) con la que debe encriptarse la comunicación. Dado que el número aleatorio procede del cliente, este puede estar seguro de que la clave de sesión se origina realmente en el servidor contactado.

El servidor remite la clave de sesión al cliente de forma cifrada. Este cifrado se realiza mediante el protocolo criptográfico Diffie-Hellmann.

2. Muestra mediante una captura de pantalla los campos de la cabecera de este protocolo y explícalos en detalle, de manera similar a como habéis hecho en apartados anteriores.

20 0.709918	149.154.167.91	192.168.1.163	SSL	143 Continuation Data
201 4.813398	192.168.1.163	firefox.settings.services.mozilla.com	TLSv1.3	567 Client Hello
216 4.823763	192.168.1.163	d228z91aul1ukj.cloudfront.net	TLSv1.3	567 Client Hello
226 4.858477	d228z91aul1ukj.cloudfront.net	192.168.1.163	TLSv1.3	1504 Server Hello, Change Cipher Spec, Application Data
229 4.858477	d228z91aul1ukj.cloudfront.net	192.168.1.163	TLSv1.3	1101 Application Data, Application Data, Application Data
230 4.858477	firefox.settings.services.mozilla.com	192.168.1.163	TLSv1.3	1504 Server Hello, Change Cipher Spec, Application Data
236 4.861723	firefox.settings.services.mozilla.com	192.168.1.163	TLSv1.3	1133 Application Data, Application Data, Application Data
238 4.864130	192.168.1.163	d228z91aul1ukj.cloudfront.net	TLSv1.3	118 Change Cipher Spec, Application Data
239 4.864352	192.168.1.163	d228z91aul1ukj.cloudfront.net	TLSv1.3	441 Application Data
240 4.864714	192.168.1.163	firefox.settings.services.mozilla.com	TLSv1.3	118 Change Cipher Spec, Application Data
247 4.897700	d228z91aul1ukj.cloudfront.net	192.168.1.163	TLSv1.3	659 Application Data
248 4.899925	192.168.1.163	d228z91aul1ukj.cloudfront.net	TLSv1.3	394 Application Data
249 4.900723	192.168.1.163	d228z91aul1ukj.cloudfront.net	TLSv1.3	78 Application Data

<p>&lt; Frame 201: 567 bytes on wire (4536 bits), 567 bytes captured (4536 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645008E2D}, id 0</p> <p>Ethernet II, Src: IntelCor_22:d5:ad (50:eb:71:22:d5:ad), Dst: zte_87:d5:02 (90:fd:73:87:d5:02)</p> <p>Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: firefox.settings.services.mozilla.com (13.224.105.125)</p> <p>0100 .... = Version: 4</p> <p>.... 0101 = Header Length: 20 bytes (5)</p> <p>&gt; Differentiated Services Field: 0xb8 (DSCP: EF PHB, ECN: Not-ECT)</p> <p>Total Length: 553</p> <p>Identification: 0x20fc (8444)</p> <p>Flags: 0x40, Don't fragment</p> <p>0... .... = Reserved bit: Not set</p> <p>1... .... = Don't fragment: Set</p> <p>..0. .... = More fragments: Not set</p> <p>Fragment Offset: 0</p> <p>Time to Live: 128</p> <p>Protocol: TCP (6)</p> <p>Header Checksum: 0x0000 [validation disabled]</p> <p>[Header checksum status: Unverified]</p> <p>Source Address: 192.168.1.163 (192.168.1.163)</p> <p>Destination Address: firefox.settings.services.mozilla.com (13.224.105.125)</p> <p>&gt; Transmission Control Protocol, Src Port: 55054, Dst Port: 443, Seq: 1, Ack: 1, Len: 513</p> <p>&gt; Transport Layer Security</p>	
--	--

20 0.709918	149.154.167.91	192.168.1.163	SSL	143 Continuation Data
201 4.813398	192.168.1.163	firefox.settings.services.mozilla.com	TLSv1.3	56 Client Hello
216 4.823763	192.168.1.163	d228z91aul1ukj.cloudfront.net	TLSv1.3	567 Client Hello

Comienza con llamada de "Client hello", normalmente, el primer mensaje del protocolo de enlace TLS es el mensaje de saludo del cliente que envía el cliente para iniciar una sesión con el servidor.

## Cliente Hello

```
> Transmission Control Protocol, Src Port: 55054, Dst Port: 443, Seq: 1, Ack: 1, Len: 513
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 508
  ▼ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 504
    Version: TLS 1.2 (0x0303)
    Random: 65c4311ba09d4340260b8c3c0ebe2ed5753bc5b2e2aad5538057ab875fe48b14
    Session ID Length: 32
    Session ID: bf83b4cf6f0c4f8fb5c04c8896548257cc07cc7e600adf32b209b43b9a7e091d
    Cipher Suites Length: 36
    > Cipher Suites (18 suites)
      Compression Methods Length: 1
    > Compression Methods (1 method)
      Extensions Length: 395
    > Extension: server_name (len=42)
```

1. Versión: el número de versión del protocolo TLS que el cliente desea usar para comunicarse con el servidor. Esta es la versión más alta admitida por el cliente.
2. Client Random: un número pseudoaleatorio de 32 bytes que se utiliza para calcular la Master secret (utilizado en la creación de la clave de cifrado).
3. Session Identifier: un número único utilizado por el cliente para identificar una sesión.
4. Cipher Suite: la lista de conjuntos de cifrado admitidos por el cliente ordenado según la preferencia del mismo. La suite de cifrado consta de un algoritmo de intercambio de claves, un algoritmo de cifrado masivo, un algoritmo MAC y una función pseudoaleatoria.
5. Compression Method: contiene una lista de algoritmos de compresión ordenados según la preferencia del cliente. Esto es opcional.

## Server Hello

216 4.823763	192.168.1.163	d228z91au1lukj.cloudfront.net	TLsv1.3	567 Client Hello
226 4.858477	d228z91au1lukj.cloudfront.net	192.168.1.163	TLsv1.3	1504 Server Hello, Change Cipher Spec, Application Data
229 4.858477	d228z91au1lukj.cloudfront.net	192.168.1.163	TLsv1.3	1101 Application Data, Application Data, Application Data
230 4.858477	firefox.settings.services.mozilla.com	192.168.1.163	TLsv1.3	1504 Server Hello, Change Cipher Spec, Application Data
236 4.861723	firefox.settings.services.mozilla.com	192.168.1.163	TLsv1.3	1133 Application Data, Application Data, Application Data
238 4.864130	192.168.1.163	d228z91au1lukj.cloudfront.net	TLsv1.3	118 Change Cipher Spec, Application Data
239 4.864352	192.168.1.163	d228z91au1lukj.cloudfront.net	TLsv1.3	441 Application Data
240 4.864714	192.168.1.163	firefox.settings.services.mozilla.com	TLsv1.3	118 Change Cipher Spec, Application Data
247 4.897700	d228z91au1lukj.cloudfront.net	192.168.1.163	TLsv1.3	659 Application Data
248 4.899925	192.168.1.163	d228z91au1lukj.cloudfront.net	TLsv1.3	394 Application Data
249 4.900723	192.168.1.163	d228z91au1lukj.cloudfront.net	TLsv1.3	78 Application Data

```

> Frame 226: 1504 bytes on wire (12032 bits), 1504 bytes captured (12032 bits) on interface \Device\NPF_{FF9C8FD1-BA58-4948-9AA3-5D4645808E2D}, id 0
> Ethernet II, Src: zte_87:d5:02 (90:fd:73:87:d5:02), Dst: IntelCor_22:d5:ad (58:eb:71:22:d5:ad)
> Internet Protocol Version 4, Src: d228z91au1lukj.cloudfront.net (54.192.105.123), Dst: 192.168.1.163 (192.168.1.163)
> Transmission Control Protocol, Src Port: 443, Dst Port: 55057, Seq: 1, Ack: 514, Len: 1450
▼ Transport Layer Security
  ▼ TLsv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 122
    ▼ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 118
      Version: TLS 1.2 (0x0303)
      Random: 064be6d1d773e240664c8289ddf1ce9185649bba7fe6eb3bf40a87bba89568
      Session ID Length: 32
      Session ID: 1355e265f3ccad56295a8115ed089eeeb49557f53fe5d377df9c54a92835d9
      Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
      Compression Method: null (0)
      Extensions Length: 46
      > Extension: supported_versions (len=2)
      > Extension: key_share (len=36)
    ▼ TLsv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.2 (0x0303)
      Length: 1
      Change Cipher Spec Message
    ▼ TLsv1.3 Record Layer: Application Data Protocol: http-over-tls
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 27
      Encrypted Application Data: ca01c721b807370519e875b123df0872b171ec8b2ef46f6457d355
      [Application Data Protocol: http-over-tls]

```

1. **Server Version:** La versión de protocolo TLS más alta admitida por el servidor que también es compatible con el cliente.
2. **Server Random:** Número pseudoaleatorio de 32 bytes utilizado para generar la master secret.
3. **Session Identifier:** Número único para identificar la sesión para la conexión correspondiente con el cliente.
4. **Cipher Suite:** La suite de cifrado más potente que admiten tanto el servidor como el cliente. Si no hay un conjunto de cifrado compatible, se crea una alerta de error de protocolo de enlace.
5. **Compression Method:** El algoritmo de compresión acordado por el servidor y el cliente. Esto es opcional.

## Server Certificate

▼ Transport Layer Security	
▼ TLsv1.2 Record Layer: Handshake Protocol: Certificate	
Content Type: Handshake (22)	
Version: TLS 1.2 (0x0303)	
Length: 2947	
▼ Handshake Protocol: Certificate	
Handshake Type: Certificate (11)	
Length: 2940	
Certificates Length: 2940	
▼ Certificates (2940 bytes)	
Certificate Length: 1072	
> Certificate: 308204a388283c2a08302018202108a3588b55c202a017df8ade5c08ff7a4388b06602a... (id-at-commonName=push.services.mozilla.com,id-at-organizationName=Mozilla Corporation,id-at-localityName=Mountain View,id-at-stateOrProvinceName=California)	
Certificate Length: 1262	
> Certificate: 308204a388283c2a08302018202108a3588b55c202a017df8ade5c08ff7a4388b06602a... (id-at-commonName=DigiCert TLS RSA SHA256 2020 CA1,id-at-organizationName=DigiCert Inc,id-at-countryName=US)	

El servidor envía al cliente una lista de certificados para autenticarse. El certificado del servidor contiene su clave pública. Esta autenticación de certificado la realiza un tercero (Autoridad de certificación) en el que confían los peers, el sistema operativo y el navegador que contiene la lista de Autoridades de certificación conocidas o mediante la importación manual de certificados en los que el usuario confía. (282)

## Server Key Exchange

- ▼ Transport Layer Security
  - ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
    - Content Type: Handshake (22)
    - Version: TLS 1.2 (0x0303)
    - Length: 333
  - ▼ Handshake Protocol: Server Key Exchange
    - Handshake Type: Server Key Exchange (12)
    - Length: 329
    - EC Diffie-Hellman Server Params

El mensaje es opcional y se envía cuando la clave pública presente en el certificado del servidor no es adecuada para el intercambio de claves o si el conjunto de cifrado impone una restricción que requiere una clave temporal. El cliente utiliza esta clave para cifrar el intercambio de claves de cliente más adelante en el proceso.

## Server Hello Done

Este mensaje indica que el servidor está listo y está esperando la respuesta del cliente.

- ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
  - Content Type: Handshake (22)
  - Version: TLS 1.2 (0x0303)
  - Length: 4
- ▼ Handshake Protocol: Server Hello Done
  - Handshake Type: Server Hello Done (14)
  - Length: 0

## Client Key Exchange

- ▼ Transport Layer Security
  - ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
    - Content Type: Handshake (22)
    - Version: TLS 1.2 (0x0303)
    - Length: 70
  - ▼ Handshake Protocol: Client Key Exchange
    - Handshake Type: Client Key Exchange (16)
    - Length: 66
    - ▼ EC Diffie-Hellman Client Params
      - Pubkey Length: 65
      - Pubkey: 04fe95f9857e92c68868a5bd83814ccc9b60c9234bbac67c0efbf8fdca2bf78b444a3d1b...
  - TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  - TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

Version: La versión del protocolo del cliente que el servidor verifica si coincide con el mensaje de saludo del cliente original.

PubKey: un número aleatorio generado por el cliente y cifrado con la clave pública del servidor. Esto, junto con el número aleatorio de cliente y servidor, se utiliza para crear el secreto maestro. Si el servidor puede descifrar el mensaje usando la clave privada y puede crear el secreto maestro localmente, entonces el cliente tiene la seguridad de que el servidor se ha autenticado.



### Change Cipher Spec

- > TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
- ▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  - Content Type: Change Cipher Spec (20)
  - Version: TLS 1.2 (0x0303)
  - Length: 1
  - Change Cipher Spec Message
- > TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

Este mensaje notifica al servidor que todos los mensajes futuros se cifrarán utilizando el algoritmo y las claves que se acaban de negociar

### Encrypted Handshake Message

- ▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
  - Content Type: Handshake (22)
  - Version: TLS 1.2 (0x0303)
  - Length: 40
  - Handshake Protocol: Encrypted Handshake Message

El mensaje Finalizado es complicado ya que es un hash de todos los mensajes intercambiados previamente junto con una etiqueta ("cliente finalizado"). Este mensaje indica que la negociación TLS se completó para el cliente.

## New Session Ticket

```

▼ TLSv1.2 Record Layer: Handshake Protocol: New Session Ticket
  Content Type: Handshake (22)
  Version: TLS 1.2 (0x0303)
  Length: 234
  > Handshake Protocol: New Session Ticket

```

El cliente almacena en caché este ticket junto con el Master Secret y otros parámetros asociados con la sesión actual.

Cuando el cliente desea reanudar la sesión, incluye el ticket en el “SessionTicket extensión” dentro del mensaje “Client Hello”.

El servidor luego descifra el boleto recibido, verifica la validez del boleto, recupera el estado de la sesión del contenido del ticket y usa este estado para reanudar la sesión.

Si el servidor verifica el ticket del cliente, luego puede renovar el ticket incluido un mensaje de protocolo de enlace “New Session Ticket” después de “Server Hello”.

## Change Cipher Spec

```

▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  Content Type: Change Cipher Spec (20)
  Version: TLS 1.2 (0x0303)
  Length: 1
  ▼ Change Cipher Spec Message
    ▼ [Expert Info (Note/Sequence): This session reuses previously negotiated keys (Session resumption)]
      [This session reuses previously negotiated keys (Session resumption)]
      [Severity level: Note]
      [Group: Sequence]

```

El servidor informa al cliente que los mensajes serán encriptados con los algoritmos y claves existentes. La capa de registro ahora cambia su estado para usar el cifrado de clave simétrica.

## Finished

```

▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
  Content Type: Handshake (22)
  Version: TLS 1.2 (0x0303)
  Length: 40
  Handshake Protocol: Encrypted Handshake Message

```

Como el mensaje Cliente finalizado, pero contiene una etiqueta diferente (“servidor finalizado”). Una vez que el cliente descifra y valida el mensaje con éxito, el servidor se autentica correctamente.

## Application Data Flow

Una vez que todo el protocolo de enlace de TLS se completa con éxito y los peers se validan, las aplicaciones de los peers pueden comenzar a comunicarse entre sí.

4294 21.889396 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4296 21.874412 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4297 21.874412 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4299 21.883890 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4300 21.884176 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4301 21.884176 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4302 21.884176 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4304 21.884680 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4305 21.884680 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]
4307 21.886150 cv.uoc.edu	192.168.1.163	TLSv1.2	1506 Application Data	[TCP segment of a reassembled PDU]

- Explica sobre qué protocolos de los niveles superiores de la pila Internet (Transporte, Red, Enlace) viaja este protocolo que nos ofrece seguridad al entrar en el campus. Describe los principales campos de cada uno de ellos.

El protocolo TLS es una evolución del protocolo SSL (Secure Sockets Layer), es un protocolo mediante el cual se establece una conexión segura por medio de un canal cifrado entre el cliente y servidor. Así el intercambio de información se realiza en un entorno seguro y libre de ataques. Se implementa sobre el protocolo de transporte TCP o UDP. El TLS Handshake Protocol es el que negocia los parámetros de seguridad mediante los cuales se van a comunicar ambos extremos, y por otro lado el protocolo de registro (TLS Record Protocol) que especifica la forma de encapsular los datos a emitir (incluyendo los parámetros de configuración acordados). La capa donde actúa el protocolo HandShake se encuentra ubicada en medio de la capa donde opera el protocolo de registro y la capa aplicación. La capa donde opera el protocolo de registro esta situada sobre la capa de transporte y acondiciona los mensajes que recibe el handshake y los envía.

- Finalmente, situaros en un paquete que tenga el protocolo **HTTP**. Si queréis, podéis optar por ir al menú *Analyze> Follow> HTTP stream*, para tener una visión más amigable de la interacción que elijáis. Muestra mediante una captura de pantalla los contenidos relacionados con el protocolo HTTP. Explica qué sucede cuando se ha aceptado vuestra contraseña del campus

El protocolo HTTP, es un protocolo de transferencia de hipertexto que se usa en la web el significado de las siglas HTTP se "HyperText Transfer Protocol", o "Protocolo de Transferencia de Hipertexto".

El http, es un lenguaje que hay entre las peticiones del cliente y las respuestas del servidor en Internet, para permitir una comunicación fluida y en un mismo "lenguaje".

Este protocolo establece las pautas a seguir, los métodos de petición y cuenta con cierta flexibilidad para incorporar nuevas peticiones y funcionalidades, especialmente a medida que se avanza en sus versiones.

## Petición HTTP:

No.	Time	Source	Destination	Protocol	Length	Info
1102	12.712312	192.168.1.163	cv.uoc.edu	TCP	66	55134 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1103	12.780230	cv.uoc.edu	192.168.1.163	TCP	64	80 → 55134 [SYN, ACK] Seq=0 Ack=1 Min=4350 Len=0 MSS=1452 SACK_PERM=1
1104	12.780394	192.168.1.163	cv.uoc.edu	TCP	54	55134 → 80 [ACK] Seq=1 Ack=1 Min=64240 Len=0
1105	12.788529	192.168.1.163	cv.uoc.edu	HTTP	4142	GET /nplincampus HTTP/1.1
1106	12.847649	cv.uoc.edu	192.168.1.163	TCP	60	80 → 55134 [ACK] Seq=1 Ack=2005 Win=7260 Len=0
1109	12.848056	cv.uoc.edu	192.168.1.163	TCP	1586	80 → 55134 [ACK] Seq=1 Ack=4089 Min=7260 Len=1452 [TCP segment of a reassembled PDU]
1110	12.848056	cv.uoc.edu	192.168.1.163	TCP	1586	80 → 55134 [ACK] Seq=1453 Ack=4089 Min=7260 Len=1452 [TCP segment of a reassembled PDU]
1111	12.848056	cv.uoc.edu	192.168.1.163	TCP	1586	80 → 55134 [ACK] Seq=4089 Ack=4089 Min=7260 Len=1452 [TCP segment of a reassembled PDU]
1112	12.848756	192.168.1.163	cv.uoc.edu	TCP	54	55134 → 80 [ACK] Seq=4089 Ack=4357 Win=65340 Len=0
1114	12.900002	cv.uoc.edu	192.168.1.163	TCP	1586	80 → 55134 [ACK] Seq=4357 Ack=4089 Min=8444 Len=1452 [TCP segment of a reassembled PDU]
1115	12.900002	cv.uoc.edu	192.168.1.163	TCP	1586	80 → 55134 [ACK] Seq=5089 Ack=4089 Min=8444 Len=1452 [TCP segment of a reassembled PDU]
1116	12.900002	cv.uoc.edu	192.168.1.163	HTTP	442	HTTP/1.1 200 OK
1117	12.900008	192.168.1.163	cv.uoc.edu	TCP	54	55134 → 80 [ACK] Seq=4089 Ack=7654 Win=65340 Len=0
1651	15.110015	192.168.1.163	cv.uoc.edu	HTTP	4062	GET /Agilidat/fin_tup_target=googleo.htm&up_favoritedownload=37C0137C0237C0387C&up_title=MissatgesRap
1605	15.100720	cv.uoc.edu	192.168.1.163	TCP	60	80 → 55134 [ACK] Seq=7654 Ack=6993 Win=11240 Len=0

```

Frame 1105: 4142 bytes on wire (33136 bits), 4142 bytes captured (33136 bits) on interface vdevicewp_0 [PPC8P01-8A58-4948-9AA3-5D46458888P2D], id 8
Ethernet II, Src: IntelCor_22:d5:ad (50:0b:71:2d:d5:ad), Dst: zle_87:d5:02 (00:0d:73:87:d5:02)
Internet Protocol Version 4, Src: 192.168.1.163 (192.168.1.163), Dst: cv.uoc.edu (213.73.48.211)
Transmission Control Protocol, Src Port: 55134, Dst Port: 80, Seq: 1, Ack: 1, Len: 4088
Hypertext Transfer Protocol
  GET /nplincampus HTTP/1.1
    [Export Info (Chat/Sequence): GET /nplincampus HTTP/1.1]
    [GET /nplincampus HTTP/1.1]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
    Request URI: /nplincampus
    Request Version: HTTP/1.1
    Host: cv.uoc.edu
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
    Accept-Language: es-ES,en;q=0.8,en-US;q=0.5,en;q=0.3
    Accept-Encoding: gzip, deflate
    Connection: keep-alive
    [truncated]Cookie: __utma=181554083.1325103160.1572000863.1574800437.1613588896.15; __gcl__=541.2.1325103160.1572000863; uoclanguage=es; __utaw=235238227.1325103160.1572000863.1574705146.1574800448.11; __utaw=181554083.[2=TipusUsuari=UOC2FE
    [HTTP request 1/2]
    [Response in frame 1116]
    [Next request in frame 1651]

```

La respuesta:

[illegible]