

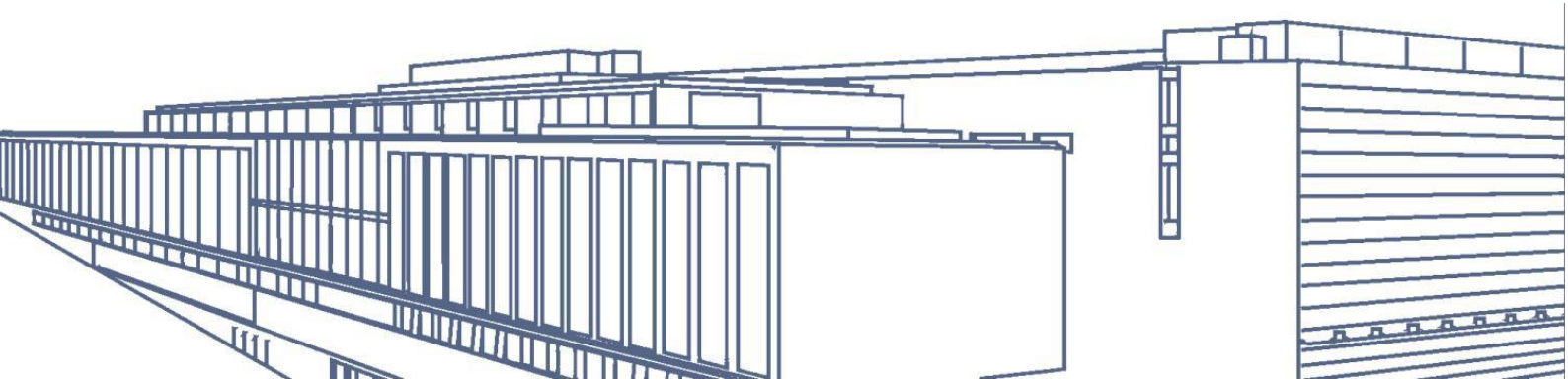


ESCOLA TÈCNICA SUPERIOR  
D'ENGINYERIA  
Universitat Rovira i Virgili



## REDES DE DATOS

### Informe Práctica 4: Análisis de protocolos



Damián Maleno González  
[franciscodamia.maleno@estudiants.urv.cat](mailto:franciscodamia.maleno@estudiants.urv.cat)  
Alberto Blanco Álvarez  
[alberto.blancoa@estudiants.urv.cat](mailto:alberto.blancoa@estudiants.urv.cat)  
Marcos Esteve Hernández  
[marcos.esteve@estudiants.urv.cat](mailto:marcos.esteve@estudiants.urv.cat)  
Curso 2019-2020

## Índice

1. Introducción .....	3
2. Distribución de las tareas .....	3
3. Realización de las tareas con TCPDump. ....	4
Tarea 1 Captura con TCPDump .....	4
Tarea 2 Mostrar los paquetes capturados con TCPDump .....	5
Tarea 3 Filtros con TCPDump .....	6
4. Realización de las tareas con Wireshark.....	7
Tarea 4 Captura con Wireshark .....	7
Tarea 5 Filtrado de paquetes con Wireshark.....	8
Tarea 6 Análisis del tráfico total .....	10
Tarea 7 Análisis de un ping .....	12
Tarea 8 Análisis de una conexión TCP .....	14
Tarea 9 Análisis de una petición HTTP .....	16
Tarea 10 Herramientas de estadística .....	17

## 1. Introducción

En esta práctica estudiaremos el comportamiento de los protocolos que hemos visto en teoría (TCP, IP ...), mediante un escuchador de red ("sniffer").

Para realizar esta práctica utilizaremos el sistema operativo Linux (en nuestro caso la distribución Debian propia de la imagen de la universidad), tanto para la parte de tcpdump como la parte de Wireshark.

Para obtener tcpdump en Linux hemos de hacer los siguientes pasos:

- Ponerse en modo superusuario "**sudo -s**", contraseña milax.
- Instalar el programa: "apt-get install tcpdump"

Para usar esta aplicación, tendremos que estar en modo superusuario.

## 2. Distribución de las tareas

TAREA	Analizador de Tramas	Estado
Tarea 1	TCPDump	Realizada
Tarea 2	TCPDump	Realizada
Tarea 3	TCPDump	Realizada
Tarea 4	WireShark	Realizada
Tarea 5	WireShark	Realizada
Tarea 6	WireShark	Realizada
Tarea 7	WireShark	Realizada
Tarea 8	WireShark	Realizada
Tarea 9	WireShark	Realizada
Tarea 10	WireShark	Realizada

### 3. Realización de las tareas con TCPDump.

#### Tarea 1 Captura con TCPDump

Tcpdump es una herramienta para la línea de comandos que permite al usuario captar y mostrar en tiempo real los paquetes transmitidos y recibidos por la red a la cual el ordenador está conectado.

Parámetros utilizados con la herramienta:

- s (snaplen): Truncado de paquetes. Sirve para limitar la captura del tráfico a una determinada cantidad de bytes, en este caso, a 1500 bytes.
- w: Escribe en un fichero los resultados, en este caso, en datagrams.txt

Iniciamos la captura con Comando: **tcpdump -s 1500 -w datagrams.txt**

Mientras que se está ejecutando TCPDump, abrimos el navegador de internet y visitamos estas dos webs: [www.google.cat](http://www.google.cat) y [www.urv.cat](http://www.urv.cat).

Finalizamos la captura con: **Crtl+c**

Mostramos el contenido del fichero con el comando: **cat datagrams.txt**

```
e0]g0BB!E0@E40@=0M000
P0!0e00000Po0
00g00000re0]00BB^E400@0f
0000P000!0è0^
000000g0e0]00BB!E0@E40@=0L000
P0!0%000S00P'
00g0000te0]:0BB^E400@pL
0000P00&!0%000
000000g0e0]f0BB!E0@E40@=0K000
P0!00090o00P{0
00g0000ue0]00BB!E0@E40@=0J000
P0!00^0r0000P0
00g0000re0]'BB^E4h@x
0000P90o!000
000000g0e0];0BB^E40_@0

0000P0r00!00 000
000000g0e0]a0BB!E0@E40 @=0I000
P0!0e00000P^0
00g00000e0]30BB!E0@E40!@=0H000
P0!0%000'00P0
00g00000e0]00BB!E0@E40"@=0G000
P0!000p00Pj0
00g00000e0]0BB!E0@E40#@=0F000
P0!00 0r0000P00
00g00000e0]`0!E0@E0K>0X
-
5000iJ0000www    instagramcom0
                                , z-p42-instagramc10facebook000/*(0000ú
                                D 0?ans0?0?b0y0w*E00
                                0w*(000
                                0ú
                                000-E00
```

La salida que se escribe en el fichero corresponde a caracteres extraños e ininteligibles. Esto se debe a que al escribir el resultado en el fichero, no se detecta bien el texto ASCII y aparece de esta forma. En la siguiente tarea, aplicamos filtros para que se muestren los paquetes de forma que se puedan leer y entender.

Si ejecutásemos la misma comanda sin la opción “-w”, observaríamos los paquetes directamente desde la pantalla de comandos.

## Tarea 2 Mostrar los paquetes capturados con TCPDump

En esta fase el objetivo es mostrar de una forma más pulida los paquetes capturados.

Para esto utilizamos los siguientes parámetros del comando tcpdump:

**-tn:** -t indica que no imprima la hora de captura de cada trama

-n indica que no se conviertan las direcciones de salida

**-r:** indicamos el fichero de lectura, en este caso, datagrams.txt

Mostramos ahora datagrams.txt de manera que sea entendible gracias al comando:

**tcpdump -tn -r datagrams.txt**

```
ARP, Request who-has 10.21.11.10 tell 10.21.11.6, length 46
IP 193.144.16.177.80 > 10.21.1.4.53774: Flags [F.], seq 23613, ack 405, win 49232, options [nop,nop,TS val 2546231254 ecr 4272201586], length 0
IP 10.21.1.4.53774 > 193.144.16.177.80: Flags [F.], seq 405, ack 23614, win 606, options [nop,nop,TS val 4272205954 ecr 2546231254], length 0
IP 193.144.16.177.80 > 10.21.1.4.53780: Flags [F.], seq 22620, ack 409, win 49232, options [nop,nop,TS val 2546231254 ecr 4272201588], length 0
IP 10.21.1.4.53780 > 193.144.16.177.80: Flags [F.], seq 409, ack 22621, win 649, options [nop,nop,TS val 4272205955 ecr 2546231254], length 0
IP 193.144.16.177.80 > 10.21.1.4.53778: Flags [F.], seq 22707, ack 406, win 49232, options [nop,nop,TS val 2546231254 ecr 4272201589], length 0
IP 193.144.16.177.80 > 10.21.1.4.53776: Flags [F.], seq 12733, ack 428, win 49232, options [nop,nop,TS val 2546231254 ecr 4272201586], length 0
IP 10.21.1.4.53778 > 193.144.16.177.80: Flags [F.], seq 406, ack 22708, win 649, options [nop,nop,TS val 4272205955 ecr 2546231254], length 0
IP 10.21.1.4.53776 > 193.144.16.177.80: Flags [F.], seq 428, ack 12734, win 486, options [nop,nop,TS val 4272205955 ecr 2546231254], length 0
IP 193.144.16.177.80 > 10.21.1.4.53774: Flags [.], ack 406, win 49232, options [nop,nop,TS val 2546231254 ecr 4272205954], length 0
IP 193.144.16.177.80 > 10.21.1.4.53780: Flags [.], ack 410, win 49232, options [nop,nop,TS val 2546231254 ecr 4272205955], length 0
IP 193.144.16.177.80 > 10.21.1.4.53778: Flags [.], ack 407, win 49232, options [nop,nop,TS val 2546231254 ecr 4272205955], length 0
IP 193.144.16.177.80 > 10.21.1.4.53776: Flags [.], ack 429, win 49232, options [nop,nop,TS val 2546231254 ecr 4272205955], length 0
IP 10.45.1.2.53 > 10.21.1.4.60356: 37621 2/2/4 CNAME z-p42-instagram.c10r.facebook.com., AAAA 2a03:2880:f204:e5:face:b00c:0:4420 (230)
ARP, Request who-has 10.21.102.14 tell 10.21.0.2, length 46
milax@d104:~$
```

Como se puede observar, ahora se han traducido los caracteres del comando anterior a unas líneas las cuales ya podemos entender, indicando:

**src > dst: Flags [tcpflags], seq data-seqno, ack ackno, win window, urg urgent, options [opts], length len**

Siendo el significado de estos parámetros:

- **src > dst:** La IP de origen y de destino.
- **Flags [tcpflags]:** Los flags empleados en la trama: S (SYN), F (FIN), P (PUSH), R (RST), U (URG), W (ECN CWR), E (ECN-Echo) o '.' (ACK), o 'none'.
- **seq data-seqno:** La porción del espacio de la secuencia que ha sido utilizado por los datos en este paquete.
- **ack ackno:** secuencia de números que indica el siguiente dato esperado en esta conexión.
- **win window:** número de bytes disponible del buffer de recepción de la otra dirección en la conexión
- **urg urgent:** indica si hay datos urgentes en el paquete.
- **options [opts]:** son las opciones del TCP.
- **length len:** longitud de los datos del paquete.

### Tarea 3 Filtros con TCPDump

Para esta tarea aplicamos diferentes filtros para observar los diferentes tipos de datagramas.

#### tcpdump -tn -r datagrams.txt tcp port 80

Con este comando capturamos únicamente los paquetes que utilizan el puerto 80, que es el puerto por el cual un servidor http escucha la petición de un cliente. Por lo tanto, estamos capturando el tráfico web.

```
IP 193.144.16.86.80 > 10.21.1.4.46080: Flags [P.], seq 1331840:1333288, ack 868, win 49232, options [nop,nop,TS val 475676871 ecr 498179818], length 1448: HTTP
IP 193.144.16.86.80 > 10.21.1.4.46080: Flags [.], seq 1333288:1334736, ack 868, win 49232, options [nop,nop,TS val 475676871 ecr 498179819], length 1448: HTTP
IP 193.144.16.86.80 > 10.21.1.4.46080: Flags [P.], seq 1334736:1336184, ack 868, win 49232, options [nop,nop,TS val 475676871 ecr 498179819], length 1448: HTTP
IP 10.21.1.4.46080 > 193.144.16.86.80: Flags [.], ack 1336184, win 13777, options [nop,nop,TS val 498179823 ecr 475676871], length 0
IP 193.144.16.86.80 > 10.21.1.4.46080: Flags [P.], seq 1336184:1338208, ack 868, win 49232, options [nop,nop,TS val 475676871 ecr 498179819], length 2024: HTTP
IP 10.21.1.4.46080 > 193.144.16.86.80: Flags [.], ack 1338208, win 13777, options [nop,nop,TS val 498179823 ecr 475676871], length 0
IP 93.184.220.29.80 > 10.21.1.4.43180: Flags [P.], seq 1:789, ack 372, win 122, options [nop,nop,TS val 370454327 ecr 1332483076], length 788: HTTP: HTTP/1.1 200 OK
IP 10.21.1.4.43180 > 93.184.220.29.80: Flags [.], ack 789, win 241, options [nop,nop,TS val 1332483220 ecr 370454327], length 0
IP 52.84.167.240.80 > 10.21.1.4.38888: Flags [P.], seq 1:489, ack 325, win 122, options [nop,nop,TS val 370454378 ecr 1444393204], length 488: HTTP: HTTP/1.1 200 OK
```

#### tcpdump -tn -r datagrams.txt tcp port 80 and dst host [www.urv.cat](http://www.urv.cat)

Con este comando filtramos aún más las tramas, captando únicamente los que utilicen el puerto 80 y cuyo destino sea la dirección host de la página [www.urv.cat](http://www.urv.cat)

```
IP 10.21.1.4.46386 > 193.144.16.86.80: Flags [S], seq 3159780140, win 29200, options [mss 1460,sackOK,TS val 500324518 ecr 0,nop,wscale 7], length 0
IP 10.21.1.4.46386 > 193.144.16.86.80: Flags [.], ack 846832459, win 229, options [nop,nop,TS val 500324520 ecr 42888255], length 0
IP 10.21.1.4.46388 > 193.144.16.86.80: Flags [S], seq 1971641319, win 29200, options [mss 1460,sackOK,TS val 500324521 ecr 0,nop,wscale 7], length 0
IP 10.21.1.4.46390 > 193.144.16.86.80: Flags [S], seq 3126829586, win 29200, options [mss 1460,sackOK,TS val 500324521 ecr 0,nop,wscale 7], length 0
IP 10.21.1.4.46392 > 193.144.16.86.80: Flags [S], seq 651480039, win 29200, options [mss 1460,sackOK,TS val 500324521 ecr 0,nop,wscale 7], length 0
IP 10.21.1.4.46394 > 193.144.16.86.80: Flags [S], seq 3376199763, win 29200, options [mss 1460,sackOK,TS val 500324521 ecr 0,nop,wscale 7], length 0
```

#### tcpdump -tn -r datagrams.txt udp port 53 or tcp port 80

Con este comando filtramos los paquetes a aquellos que utilicen el protocolo udp (user datagram protocol) a través del puerto 53 (puerto del servicio DNS), es decir, captamos las peticiones de DNS de usuarios; y también captamos las tramas a través del puerto 80 (peticiones http) con protocolo tcp.

```
IP 10.21.1.4.41542 > 52.94.234.174.80: Flags [S], seq 1410558977, win 29200, options [mss 1460,sackOK,TS val 4242291692 ecr 0,nop,wscale 7], length 0
IP 52.94.234.174.80 > 10.21.1.4.41542: Flags [S.], seq 1433566835, ack 1410558978, win 14480, options [mss 1460,sackOK,TS val 370668890 ecr 4242291692,nop,wscale 7], length 0
IP 10.21.1.4.41542 > 52.94.234.174.80: Flags [.], ack 1, win 229, options [nop,nop,TS val 4242291697 ecr 370668890], length 0
IP 10.21.1.4.41542 > 52.94.234.174.80: Flags [P.], seq 1:289, ack 1, win 229, options [nop,nop,TS val 4242291698 ecr 370668890], length 288: HTTP: GET /x.png HTTP/1.1
IP 52.94.234.174.80 > 10.21.1.4.41542: Flags [.], ack 289, win 122, options [nop,nop,TS val 370668890 ecr 4242291698], length 0
```

#### tcpdump -tn -r datagrams.txt not \((dst host [www.google.cat](http://www.google.cat))\)

Este comando filtra y muestra todas las tramas exceptuando aquellas que tengan como destino el host de [www.google.cat](http://www.google.cat)

```
IP 172.217.168.162.443 > 10.21.1.4.42290: Flags [P.], seq 5409:5440, ack 2065, win 162, options [nop,nop,TS val 370725065 ecr 2013790752], length 31
IP 10.21.1.4.42290 > 172.217.168.162.443: Flags [.], ack 5440, win 356, options [nop,nop,TS val 2013790791 ecr 370725065], length 0
IP 172.217.168.162.443 > 10.21.1.4.42290: Flags [P.], seq 5440:5479, ack 2065, win 162, options [nop,nop,TS val 370725065 ecr 2013790752], length 39
IP 10.21.1.4.42290 > 172.217.168.162.443: Flags [P.], seq 2065:2104, ack 5479, win 356, options [nop,nop,TS val 2013790791 ecr 370725065], length 39
IP 172.217.168.162.443 > 10.21.1.4.42290: Flags [.], ack 2104, win 162, options [nop,nop,TS val 370725065 ecr 2013790791], length 0
IP 10.21.1.4.42274 > 172.217.168.162.443: Flags [P.], seq 1628:1820, ack 6019, win 386, options [nop,nop,TS val 2013790795 ecr 370725061], length 192
IP 172.217.168.162.443 > 10.21.1.4.42274: Flags [.], ack 1820, win 157, options [nop,nop,TS val 370725065 ecr 2013790795], length 0
IP 172.217.168.162.443 > 10.21.1.4.42274: Flags [P.], seq 6019:6103, ack 1820, win 157, options [nop,nop,TS val 370725069 ecr 2013790795], length 84
IP 172.217.168.162.443 > 10.21.1.4.42274: Flags [P.], seq 6103:6134, ack 1820, win 157, options [nop,nop,TS val 370725069 ecr 2013790795], length 31
IP 172.217.168.162.443 > 10.21.1.4.42274: Flags [P.], seq 6134:6173, ack 1820, win 157, options [nop,nop,TS val 370725069 ecr 2013790795], length 39
IP 10.21.1.4.42274 > 172.217.168.162.443: Flags [.], ack 6173, win 386, options [nop,nop,TS val 2013790834 ecr 370725069], length 0
IP 10.21.1.4.42274 > 172.217.168.162.443: Flags [P.], seq 1820:1859, ack 6173, win 386, options [nop,nop,TS val 2013790834 ecr 370725069], length 39
```



## 4. Realización de las tareas con Wireshark

### Tarea 4 Captura con Wireshark

En esta tarea vamos a capturar los paquetes mediante la herramienta Wireshark. Para ello, abrimos el programa y lo ejecutamos en la red por la que se enviarán y recibirán los paquetes, en nuestro caso la enp0s3. Una vez comenzamos las capturas, abrimos el navegador y accedemos a las páginas <http://www.example.com> y <https://www.example.com>.

17	5.672729530	10.21.1.4	10.45.1.2	DNS	75 Standard query 0x31b7 A www.example.com
18	5.674177800	10.45.1.2	10.21.1.4	DNS	195 Standard query response 0x31b7 A www.example.com A 93.184.216.34 NS b.iana-servers.net NS a.iana-servers.net AAAA 2001:500:8d::53 AAAA 2001:500:8f::53
19	5.675536556	10.21.1.4	93.184.216.34	TCP	74 40440 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK PERM=1 TSval=3356381213 TSecr=0 WS=128
20	5.677897049	93.184.216.34	10.21.1.4	TCP	74 80 → 40440 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK PERM=1 TSval=370803042 TSecr=3356381213 WS=128
21	5.678000115	10.21.1.4	93.184.216.34	TCP	66 40440 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3356381215 TSecr=370803042
22	5.681649938	10.21.1.4	10.45.1.2	DNS	75 Standard query 0x3ea7 A www.example.com
23	5.681705400	10.21.1.4	10.45.1.2	DNS	75 Standard query 0x8fe2 AAAA www.example.com

294	68.325020252	10.21.0.2	224.0.0.18	VRRP	60 Announcement (v2)
295	68.396560347	10.21.1.4	10.45.1.2	DNS	80 Standard query 0x61fe A diaridigital.urv.cat
296	68.396613004	10.21.1.4	10.45.1.2	DNS	80 Standard query 0x4c39 AAAA diaridigital.urv.cat
297	68.398016421	10.45.1.2	10.21.1.4	DNS	166 Standard query response 0x61fe A diaridigital.urv.cat A 193.144.16.177 NS aliga.urv.cat NS dns.urv.cat A 193.144.16.4 A 193.144.16.9
298	68.398057203	10.45.1.2	10.21.1.4	DNS	131 Standard query response 0x4c39 AAAA diaridigital.urv.cat 50A dns.urv.cat
299	69.325802321	10.21.0.2	224.0.0.18	VRRP	60 Announcement (v2)
300	70.012611272	Entropy 7d:b8:9b	Spanning tree (for...) STP	60 RST. Root = 32768/0/00:01:f4:7d:b8:80 Cost = 0 Port = 0x801c	
301	70.192096296	216.58.201.174	10.21.1.4	TLSv1.2	120 Application Data
302	70.234038488	10.21.1.4	216.58.201.174	TCP	66 46648 → 443 [ACK] Seq=503 Ack=489 Win=6241 Len=0 TSval=3719888076 TSecr=370809494
303	70.326349263	10.21.0.2	224.0.0.18	VRRP	60 Announcement (v2)
304	71.217209304	10.21.1.4	10.45.1.2	DNS	75 Standard query 0xb1f1 A www.example.com
305	71.217261292	10.21.1.4	10.45.1.2	DNS	75 Standard query 0xfc2b AAAA www.example.com
306	71.218813101	10.45.1.2	10.21.1.4	DNS	207 Standard query response 0xfc2b AAAA www.example.com AAAA 2606:2800:220:1:248:1893:25c0:1946 NS b.iana-servers.net NS a.iana-servers.net AAAA 2001:500:8d::53 AAAA 2001:500:8f::53
307	71.218855952	10.45.1.2	10.21.1.4	DNS	195 Standard query response 0xb1f1 A www.example.com A 93.184.216.34 NS b.iana-servers.net NS a.iana-servers.net AAAA 2001:500:8d::53 AAAA 2001:500:8f::53
308	71.327267480	10.21.0.2	224.0.0.18	VRRP	60 Announcement (v2)
309	72.326276719	10.21.0.2	224.0.0.18	VRRP	60 Announcement (v2)
310	72.000662400	Entropy 7d:b8:9b	Spanning tree (for...) STP	60 RST. Root = 32768/0/00:01:f4:7d:b8:80 Cost = 0 Port = 0x801c	

Estas son las capturas correspondientes a los paquetes que hemos capturado, respectivamente.

## Tarea 5 Filtrado de paquetes con Wireshark

En este apartado aplicaremos una serie de filtros de búsqueda. Realizamos el mismo proceso, navegando por las páginas <http://www.example.com> y <https://www.example.com>, parando la ejecución del programa y aplicando los filtros.

Primero, aplicamos el filtro para que solo se muestren las peticiones DNS:

Time	Source	Destination	Protocol	Length	Info
17.5.672729530	10.21.1.4	10.45.1.2	DNS	75	Standard query 0x31b7 A www.example.com
18.5.674177800	10.45.1.2	10.21.1.4	DNS	195	Standard query response 0x31b7 A www.example.com A 93.184.216.34 NS b.iana-servers.net NS a.iana-servers.net AAAA 2001:500:8d::53 AAAA 2001:500:8f::53
22.5.681649938	10.21.1.4	10.45.1.2	DNS	75	Standard query 0x3ea7 A www.example.com
23.5.681705400	10.21.1.4	10.45.1.2	DNS	75	Standard query 0x8fe2 AAAA www.example.com
24.5.683241793	10.45.1.2	10.21.1.4	DNS	195	Standard query response 0x3ea7 A www.example.com A 93.184.216.34 NS b.iana-servers.net NS a.iana-servers.net AAAA 2001:500:8d::53 AAAA 2001:500:8f::53
25.5.683287668	10.45.1.2	10.21.1.4	DNS	207	Standard query response 0x8fe2 AAAA www.example.com AAAA 2001:500:8d::53 AAAA 2001:500:8f::53

A continuación, desactivamos el filtro anterior y buscamos una petición de http. No encontramos ninguna petición http por lo que realizamos un ping a google.com para asegurarnos de generar una petición.

Seleccionamos la petición y pedimos que reconstruya la conversación, es decir, que nos muestre las peticiones y respuestas que ha tenido con el servidor al que hemos contactado.

Primero, le hacemos una petición para que nos muestre el html de la página web, por lo que nos responde con el código en html. Seguidamente, le pedimos otro html que no posee, por lo que responde con un not found y nos vuelve a enviar el html inicial.

No.	Time	Source	Destination	Protocol	Length	Info
1683	29.324082820	10.21.1.4	172.217.16.227	OCSP	439	Request
1818	29.459077907	172.217.16.227	10.21.1.4	OCSP	767	Response
3467	30.995870184	10.21.1.4	172.217.16.227	OCSP	439	Request
3481	31.227780963	172.217.16.227	10.21.1.4	OCSP	767	Response
4569	33.635190000	10.21.1.4	172.217.16.227	OCSP	440	Request
4574	33.638536077	10.21.1.4	172.217.16.227	OCSP	440	Request
4603	33.757746726	172.217.16.227	10.21.1.4	OCSP	768	Response
4606	33.772615470	172.217.16.227	10.21.1.4	OCSP	768	Response
6203	113.208179434	10.21.1.4	93.184.220.29	OCSP	437	Request
6207	113.425544789	93.184.220.29	10.21.1.4	OCSP	854	Response
7328	213.304909470	10.21.1.4	93.184.216.34	HTTP	389	GET / HTTP/1.1
7330	213.512676381	93.184.216.34	10.21.1.4	HTTP	1075	HTTP/1.1 200 OK (text/html)
7334	213.585804483	10.21.1.4	93.184.216.34	HTTP	321	GET /favicon.ico HTTP/1.1
7336	213.691370484	93.184.216.34	10.21.1.4	HTTP	1066	HTTP/1.1 404 Not Found (text/html)

Time	Source	Destination	Protocol	Length	Info
78.12.934755815	10.21.1.4	10.45.1.2	DNS	75	Standard query 0x38e8 A docs.google.com
79.12.934810659	10.21.1.4	10.45.1.2	DNS	75	Standard query 0x9524 AAAA docs.google.com
83.12.936216787	10.45.1.2	10.21.1.4	DNS	339	Standard query response 0x38e8 A docs.google.com A 216.58.201.174 NS ns1.google.com NS ns2.google.com
84.12.936269649	10.45.1.2	10.21.1.4	DNS	351	Standard query response 0x9524 AAAA docs.google.com AAAA 2a00:1450:4003:80b::200e NS ns3.google.com
113.14.203479570	10.21.1.4	10.45.1.2	DNS	75	Standard query 0x4339 A play.google.com
114.14.203612801	10.21.1.4	10.45.1.2	DNS	75	Standard query 0xe2df AAAA play.google.com
118.14.205262613	10.45.1.2	10.21.1.4	DNS	339	Standard query response 0x4339 A play.google.com A 216.58.211.46 NS ns3.google.com NS ns1.google.com
119.14.205294721	10.45.1.2	10.21.1.4	DNS	351	Standard query response 0xe2df AAAA play.google.com AAAA 2a00:1450:4003:802::200e NS ns1.google.com
133.15.212015544	10.21.1.4	10.45.1.2	DNS	75	Standard query 0xb9f6 A play.google.com
134.15.212081859	10.21.1.4	10.45.1.2	DNS	75	Standard query 0xf136 AAAA play.google.com
139.15.213641937	10.45.1.2	10.21.1.4	DNS	351	Standard query response 0xf136 AAAA play.google.com AAAA 2a00:1450:4003:802::200e NS ns3.google.com
140.15.213686896	10.45.1.2	10.21.1.4	DNS	339	Standard query response 0xb9f6 A play.google.com A 216.58.211.46 NS ns3.google.com NS ns2.google.com
156.20.285225609	10.21.1.4	10.45.1.2	DNS	70	Standard query 0x43f6 A google.com
157.20.285288622	10.21.1.4	10.45.1.2	DNS	70	Standard query 0x522e AAAA google.com
158.20.286811225	10.45.1.2	10.21.1.4	DNS	334	Standard query response 0x43f6 A google.com A 172.217.168.174 NS ns4.google.com NS ns1.google.com
159.20.337794435	10.45.1.2	10.21.1.4	DNS	346	Standard query response 0x522e AAAA google.com AAAA 2a00:1450:4003:80a::200e NS ns2.google.com NS
162.20.356593040	10.21.1.4	10.45.1.2	DNS	88	Standard query 0xacac PTR 174.168.217.172.in-addr.arpa
163.20.358388929	10.45.1.2	10.21.1.4	DNS	385	Standard query response 0xacac PTR 174.168.217.172.in-addr.arpa PTR mad07s10-in-f14.1e100.net NS n

En el apartado de información que se muestra sobre cada paquete capturado, se puede observar el número del paquete, el tiempo, la IP destino y fuente, el tipo de protocolo empleado e información adicional sobre el paquete.

Esta información sigue el siguiente formato:

**Standard query response [a hex number] [record type] [domain] (CNAME [canonical domain]) + [record type] [IP Address] ([record type] [IP Address])**

Para la dos siguientes fotografías: No se encuentra ninguna petición https ya que este no es un protocolo diferente al http. La ‘s’ indica que existe una protección, integridad y confidencialidad en la transferencia de datos entre los ordenadores de usuario y el sitio web.



```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 10 Dec 2019 10:51:25 GMT
Etag: "3147526947"
Expires: Tue, 17 Dec 2019 10:51:25 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (bsa/EB15)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 648
```

```
<!doctype html>
<html>
<head>
  <title>Example Domain</title>

  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
    }
    div {
      width: 600px;
      margin: 5em auto;
      padding: 2em;
      background-color: #fdfdff;
      border-radius: 0.5em;
      box-shadow: 2px 3px 7px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
      color: #38488f;
      text-decoration: none;
    }
    @media (max-width: 700px) {
      div {
        margin: 0 auto;
        width: auto;
      }
    }
  </style>
```

```
<p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

```
GET /favicon.ico HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
```

```
HTTP/1.1 404 Not Found
Content-Encoding: gzip
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 10 Dec 2019 10:51:25 GMT
Expires: Tue, 17 Dec 2019 10:51:25 GMT
Last-Modified: Thu, 05 Dec 2019 04:59:59 GMT
Server: ECS (bsa/EB21)
Vary: Accept-Encoding
X-Cache: 404-HIT
Content-Length: 648
```

```
<!doctype html>
<html>
<head>
  <title>Example Domain</title>

  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
    }
    div {
      width: 600px;
      margin: 5em auto;
      padding: 2em;
      background-color: #fdfdff;
      border-radius: 0.5em;
      box-shadow: 2px 3px 7px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
      color: #38488f;
      text-decoration: none;
    }
    @media (max-width: 700px) {
      div {
        margin: 0 auto;
        width: auto;
      }
    }
  </style>
```

## Tarea 6 Análisis del tráfico total

Realizamos Ping a otras máquinas del laboratorio y capturamos los paquetes que se envían y reciben durante 10 segundos. Paramos la captura de paquetes.

Realizamos ping a la maquina con dirección IP 10.21.1.9 desde la maquina 10.21.1.7.

Para crear un filtro que nos muestre todos los paquetes que pertenecen al broadcast, utilizamos como condición que la dirección ethernet de destino (eth.dst) sea ff.ff.ff.ff.ff, que es la dirección de broadcast a nivel de enlace, es decir, la dirección MAC de broadcast.

**(eth.dst == ff.ff.ff.ff.ff)**

(eth.dst == ff.ff.ff.ff.ff)						
No.	Time	Source	Destination	Protocol	Length	Info
2908	20.584418353	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x501a[Malformed Packet]
3495	24.587864339	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x501a[Malformed Packet]
951	6.769473003	10.21.1.1	10.21.255.255	NBNS	92	Name query NB ISATAP<00>
1065	7.518553379	10.21.1.1	10.21.255.255	NBNS	92	Name query NB ISATAP<00>
6	0.061842765	10.21.1.12	10.21.255.255	NBNS	92	Name query NB WPAD<00>
105	0.803082508	10.21.1.12	10.21.255.255	NBNS	92	Name query NB WPAD<00>
219	1.557853693	10.21.1.12	10.21.255.255	NBNS	92	Name query NB WPAD<00>
130	0.931184135	10.21.1.17	255.255.255.255	SNMP	115	get-request 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0 1.3.6.1.2.1.2.2.1.6.1
268	1.931188787	10.21.1.17	255.255.255.255	SNMP	115	get-request 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0 1.3.6.1.2.1.2.2.1.6.1
563	4.062324018	10.21.1.17	255.255.255.255	UDP	60	47317 → 3289 Len=15
723	5.147261962	10.21.1.17	255.255.255.255	UDP	60	51234 → 3289 Len=15
930	6.501573814	10.21.1.17	255.255.255.255	UDP	79	42682 → 1124 Len=37
654	4.628071073	EncantoN_00:23:00	Broadcast	ARP	60	Who has 10.21.10.18? Tell 10.21.0.23
795	5.643596846	EncantoN_00:23:00	Broadcast	ARP	60	Who has 10.21.10.18? Tell 10.21.0.23
939	6.667817427	EncantoN_00:23:00	Broadcast	ARP	60	Who has 10.21.10.18? Tell 10.21.0.23
4475	31.690557310	EncantoN_01:00:00	Broadcast	ARP	60	Who has 169.254.188.118? Tell 0.0.0.0
790	5.550887136	EncantoN_01:01:01	Broadcast	ARP	60	Who has 10.21.0.1? Tell 10.21.1.1
922	6.456918028	EncantoN_01:01:01	Broadcast	ARP	60	Who has 10.21.0.1? Tell 10.21.1.1
2406	17.192393653	EncantoN_01:09:01	Broadcast	ARP	60	Who has 10.21.1.7? Tell 10.21.1.9
3918	27.497914919	Enterasy_a1:40:77	Broadcast	ARP	60	Who has 10.21.102.14? Tell 10.21.0.2
4058	28.495946430	Enterasy_a1:40:77	Broadcast	ARP	60	Who has 10.21.102.14? Tell 10.21.0.2
20	0.131005185	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.13.10? Tell 10.21.11.19
21	0.131194076	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.13.18? Tell 10.21.11.19
22	0.131679618	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.12.10? Tell 10.21.11.19
23	0.131860620	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.12.3? Tell 10.21.11.19
24	0.132226307	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.12.11? Tell 10.21.11.19
36	0.195908688	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.13.17? Tell 10.21.11.19
37	0.196085554	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.12.19? Tell 10.21.11.19
40	0.227826176	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.13.7? Tell 10.21.11.19
41	0.227946792	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.16.0? Tell 10.21.11.19
42	0.228380845	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.12.16? Tell 10.21.11.19
49	0.292287208	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.13.19? Tell 10.21.11.19
54	0.323583042	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.13.9? Tell 10.21.11.19
59	0.387548922	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.11.18? Tell 10.21.11.19
73	0.483416581	Giga-Byt_22:dc:b9	Broadcast	ARP	60	Who has 10.21.12.0? Tell 10.21.11.19
▶ Frame 2406: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0						
▶ Ethernet II, Src: EncantoN_01:09:01 (00:10:21:01:09:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)						
▶ Address Resolution Protocol (request)						

Durante los 10 segundo capturamos un total de 1165 paquetes, de los cuales 936 eran de broadcast, lo cual corresponde a entorno al 80% del tráfico.

Este tráfico broadcast se corresponde al protocolo ARP. El protocolo de ARP (Address Resolution Protocol) pertenece a la capa de enlace. Se encarga de detectar la dirección MAC (o dirección física) de una máquina correspondiente a una dirección IP. La máquina emisora difunde un paquete ARP request por broadcast con la IP a la que necesita enviar cierto paquete. La máquina de la red LAN a la que corresponde dicha IP responde con una ARP replay que contiene su dirección MAC (identificador del dispositivo de red de la máquina). La primera máquina guardará dicha IP y la MAC asociada en una caché, la tabla de ARP, para futuras transferencias. Por ello, a la hora de realizar la tarea, debemos asegurarnos de que las IP objetivo no están en la tabla.

((ip.addr == 10.21.1.7))(((eth.dst == ff.ff.ff.ff.ff.ff)))						
No.	Time	Source	Destination	Protocol	Length	Info
519	6.305744669	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.16.0? Tell 10.21.17.8
520	6.305955288	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.12.9? Tell 10.21.17.8
521	6.322917109	Giga-Byt de:09:9e	Broadcast	ARP	60	Who has 10.21.12.6? Tell 10.21.11.10
522	6.335835316	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.0? Tell 10.21.11.4
523	6.336051348	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.12.13? Tell 10.21.11.4
524	6.336329557	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.3? Tell 10.21.11.4
525	6.336572046	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.13? Tell 10.21.11.4
526	6.336851079	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.12.9? Tell 10.21.11.4
527	6.341335462	10.21.1.7	216.58.201.170	TCP	66	58306 → 443 [ACK] Seq=1 Ack=1 Win=930 Len=0 TSval=986860333 TSecr=430903603
528	6.345372293	216.58.201.170	10.21.1.7	TCP	66	[TCP ACKed unseen segment] 443 → 58306 [ACK] Seq=1 Ack=2 Win=653 Len=0 TSval=430908129 TSecr=986815040
529	6.400321713	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.11.1? Tell 10.21.11.4
530	6.400970737	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.11.3? Tell 10.21.17.8
531	6.431711266	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.19? Tell 10.21.11.4
532	6.449116807	10.21.1.7	10.21.1.9	ICMP	98	Echo (ping) request id=0x0c4e, seq=1/256, ttl=64 (reply in 533)
533	6.451838211	10.21.1.9	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0c4e, seq=1/256, ttl=64 (request in 532)
534	6.482641024	Giga-Byt de:09:9e	Broadcast	ARP	60	Who has 10.21.12.12? Tell 10.21.11.10
535	6.499870797	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.12.10? Tell 10.21.11.4
536	6.527100036	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.10? Tell 10.21.11.4
537	6.528417964	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.16.0? Tell 10.21.11.4
538	6.806499454	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.12.16? Tell 10.21.11.4
539	6.806511369	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.17? Tell 10.21.11.4
540	6.807097811	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.12.0? Tell 10.21.11.4
541	6.819989820	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.2? Tell 10.21.11.4
542	6.848914261	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.18.2? Tell 10.21.17.8
543	6.848929410	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.13.0? Tell 10.21.17.8
544	6.849339702	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.12.0? Tell 10.21.17.8
546	6.880608310	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.11.11? Tell 10.21.11.4
547	6.880368148	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.11.5? Tell 10.21.11.4
548	6.880536962	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.17.10? Tell 10.21.17.8
549	6.880548143	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.16? Tell 10.21.11.4
550	6.880751933	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.13.3? Tell 10.21.17.8
551	6.880986490	Giga-Byt 2d:36:12	Broadcast	ARP	60	Who has 10.21.13.9? Tell 10.21.11.4
552	6.880997295	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.17.7? Tell 10.21.17.8
553	6.881267868	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.11.1? Tell 10.21.17.8
554	6.881450926	Giga-Byt 16:77:5b	Broadcast	ARP	60	Who has 10.21.11.11? Tell 10.21.17.8
▶ Frame 610: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0						
▶ Ethernet II, Src: EncantoN 01:07:01 (00:10:21:01:07:01), Dst: EncantoN 01:09:01 (00:10:21:01:09:01)						
▶ Internet Protocol Version 4, Src: 10.21.1.7, Dst: 10.21.1.9						
▶ Internet Control Message Protocol						

Una vez hemos hecho esto, volvemos a capturar el tráfico y hacemos nuevamente ping a estas máquinas. En esta ocasión, observamos que ya no se emite un paquete broadcast preguntando cuál es la IP destino y la IP origen de las máquinas que realizan el ping. Esto es debido a que las direcciones se encuentran ya en la cache de ARP. Así, no tendrá que preguntar, sino que lo consultará en la tabla y enviará el paquete.

## Tarea 7 Análisis de un ping

Hacemos ping a una máquina Linux y a una máquina Windows al mismo tiempo.

(ip.addr == 10.21.1.7)

No.	Time	Source	Destination	Protocol	Length	Info
37	0.832448391	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=392/34817, ttl=64 (reply in 38)
38	0.832500957	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=392/34817, ttl=64 (request in 37)
87	1.857071215	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=393/35073, ttl=64 (reply in 88)
88	1.857111364	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=393/35073, ttl=64 (request in 87)
133	2.880525456	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=394/35329, ttl=64 (reply in 134)
134	2.880553068	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=394/35329, ttl=64 (request in 133)
139	2.992028040	10.21.1.7	216.58.211.46	TCP	66	33746 → 443 [ACK] Seq=1 Ack=1 Win=1444 Len=0 TSval=1171614282 TSecr=431108900
140	2.993378061	216.58.211.46	10.21.1.7	TCP	66	[TCP ACKed unseen segment] 443 → 33746 [ACK] Seq=1 Ack=2 Win=1263 Len=0 TSval=431113406 TSecr=1171477855
172	3.583293994	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=1/256, ttl=64 (reply in 173)
173	3.583937907	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=1/256, ttl=64 (request in 172)
183	3.904495017	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=395/35585, ttl=64 (reply in 184)
184	3.904523081	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=395/35585, ttl=64 (request in 183)
190	4.124959013	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=1/256, ttl=64 (reply in 191)
191	4.125504225	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=1/256, ttl=128 (request in 190)
243	4.590432734	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=2/512, ttl=64 (reply in 244)
244	4.591164791	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=2/512, ttl=64 (request in 243)
258	4.928458749	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=396/35841, ttl=64 (reply in 259)
259	4.928506483	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=396/35841, ttl=64 (request in 258)
265	5.133507778	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=2/512, ttl=64 (reply in 266)
266	5.134110920	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=2/512, ttl=128 (request in 265)
313	5.600506844	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=3/768, ttl=64 (reply in 314)
314	5.601292574	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=3/768, ttl=64 (request in 313)
327	5.952494938	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=397/36097, ttl=64 (reply in 328)
328	5.952526470	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=397/36097, ttl=64 (request in 327)
332	6.160806495	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=3/768, ttl=64 (reply in 333)
333	6.160628719	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=3/768, ttl=128 (request in 332)
373	6.608106368	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=4/1024, ttl=64 (reply in 374)
374	6.608788860	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=4/1024, ttl=64 (request in 373)
388	6.976806056	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=398/36353, ttl=64 (reply in 389)
389	6.976938843	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=398/36353, ttl=64 (request in 388)
392	7.184068323	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=4/1024, ttl=64 (reply in 393)
393	7.184666424	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=4/1024, ttl=128 (request in 392)
407	7.632073236	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=5/1280, ttl=64 (reply in 408)

▶ Frame 333: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
▶ Ethernet II, Src: EncantoN\_01:06:01 (00:10:21:01:06:01), Dst: EncantoN\_01:07:01 (00:10:21:01:07:01)  
▶ Internet Protocol Version 4, Src: 10.21.1.6, Dst: 10.21.1.7  
    0100 .... = Version: 4  
    .... 0101 = Header Length: 20 bytes (5)  
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
        Total Length: 84  
        Identification: 0x1226 (4646)  
    ▶ Flags: 0x0000  
Time to live: 128  
Protocol: ICMP (1)  
Header checksum: 0x124d [validation disabled]  
[Header checksum status: Unverified]  
Source: 10.21.1.6  
Destination: 10.21.1.7  
▶ Internet Control Message Protocol

0000 00 10 21 01 07 01 00 10 21 01 06 01 08 00 45 00 ..!.....!....E.  
0010 00 54 12 26 00 00 80 01 12 4d 0a 15 01 06 0a 15 .T.6...M.....

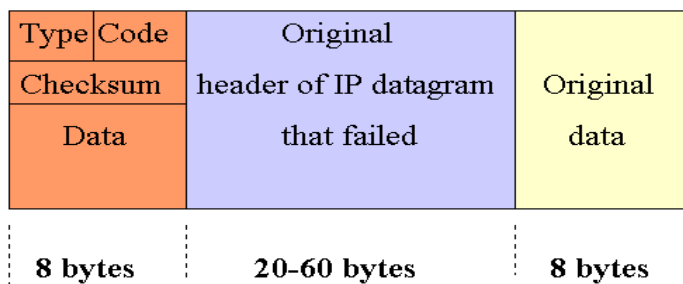
(ip.addr == 10.21.1.7)

No.	Time	Source	Destination	Protocol	Length	Info
37	0.832448391	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=392/34817, ttl=64 (reply in 38)
38	0.832500957	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=392/34817, ttl=64 (request in 37)
87	1.857071215	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=393/35073, ttl=64 (reply in 88)
88	1.857111364	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=393/35073, ttl=64 (request in 87)
133	2.880525456	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=394/35329, ttl=64 (reply in 134)
134	2.880553068	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=394/35329, ttl=64 (request in 133)
139	2.992028040	10.21.1.7	216.58.211.46	TCP	66	33746 → 443 [ACK] Seq=1 Ack=1 Win=1444 Len=0 TSval=1171614282 TSecr=431108900
140	2.993378061	216.58.211.46	10.21.1.7	TCP	66	[TCP ACKed unseen segment] 443 → 33746 [ACK] Seq=1 Ack=2 Win=1263 Len=0 TSval=431113406 TSecr=1171477855
172	3.583293994	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=1/256, ttl=64 (reply in 173)
173	3.583937907	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=1/256, ttl=64 (request in 172)
183	3.904495017	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=395/35585, ttl=64 (reply in 184)
184	3.904523081	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=395/35585, ttl=64 (request in 183)
190	4.124959013	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=1/256, ttl=64 (reply in 191)
191	4.125504225	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=1/256, ttl=128 (request in 190)
243	4.590432734	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=2/512, ttl=64 (reply in 244)
244	4.591164791	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=2/512, ttl=64 (request in 243)
258	4.928458749	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=396/35841, ttl=64 (reply in 259)
259	4.928506483	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=396/35841, ttl=64 (request in 258)
265	5.133507778	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=2/512, ttl=64 (reply in 266)
266	5.134110920	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=2/512, ttl=128 (request in 265)
313	5.600506844	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=3/768, ttl=64 (reply in 314)
314	5.601292574	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=3/768, ttl=64 (request in 313)
327	5.952494938	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=397/36097, ttl=64 (reply in 328)
328	5.952526470	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=397/36097, ttl=64 (request in 327)
332	6.160806495	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=3/768, ttl=64 (reply in 333)
333	6.160628719	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=3/768, ttl=128 (request in 332)
373	6.608106368	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=4/1024, ttl=64 (reply in 374)
374	6.608788860	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f97, seq=4/1024, ttl=64 (request in 373)
388	6.976806056	10.21.1.8	10.21.1.7	ICMP	98	Echo (ping) request id=0x0b0a, seq=398/36353, ttl=64 (reply in 389)
389	6.976938843	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) reply id=0x0b0a, seq=398/36353, ttl=64 (request in 388)
392	7.184068323	10.21.1.7	10.21.1.6	ICMP	98	Echo (ping) request id=0x0f98, seq=4/1024, ttl=64 (reply in 393)
393	7.184666424	10.21.1.6	10.21.1.7	ICMP	98	Echo (ping) reply id=0x0f98, seq=4/1024, ttl=128 (request in 392)
407	7.632073236	10.21.1.7	10.21.1.8	ICMP	98	Echo (ping) request id=0x0f97, seq=5/1280, ttl=64 (reply in 408)

▶ Frame 314: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
▶ Ethernet II, Src: EncantoN\_01:08:01 (00:10:21:01:08:01), Dst: EncantoN\_01:07:01 (00:10:21:01:07:01)  
▶ Internet Protocol Version 4, Src: 10.21.1.8, Dst: 10.21.1.7  
    0100 .... = Version: 4  
    .... 0101 = Header Length: 20 bytes (5)  
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
        Total Length: 84  
        Identification: 0x67ab (26539)  
    ▶ Flags: 0x0000  
Time to live: 64  
Protocol: ICMP (1)  
Header checksum: 0xfcc5 [validation disabled]  
[Header checksum status: Unverified]  
Source: 10.21.1.8  
Destination: 10.21.1.7  
▶ Internet Control Message Protocol

El protocolo que se utiliza es ICMP ya que el comando Ping utiliza este protocolo. Es un subprotocolo de IP para intercambiar datos de estado o errores, por lo que acompaña a este. La máquina que realiza el Ping envía un mensaje IP con cabecera ICMP de tipo Echo Request, y la máquina destino responde con un Echo Reply cuando recibe el paquete. Si el destino no recibe el paquete, el último nodo disponible envía un paquete ICMP Destination Unreachable al origen.

### *Format of the ICMP Error Message*



Los datos que transporta el datagrama son: el tipo de mensaje que es; el código asociado a este paquete, que especifica características del tipo de mensaje; el identificador junto con una secuencia numérica; la checksum, que garantiza que no se haya producido ninguna modificación accidental en el datagrama; y por último los datos que envíe para cada caso en específico.

El campo TTL es un campo de 8 bits que se encuentra en la cabecera IP. Cada sistema operativo establece un TTL por defecto a sus datagramas IP. Se trata de un contador que se decrementa a medida que el datagrama pasa por nodos de Internet. Su objetivo es evitar que los datos circulen indefinidamente por la red.

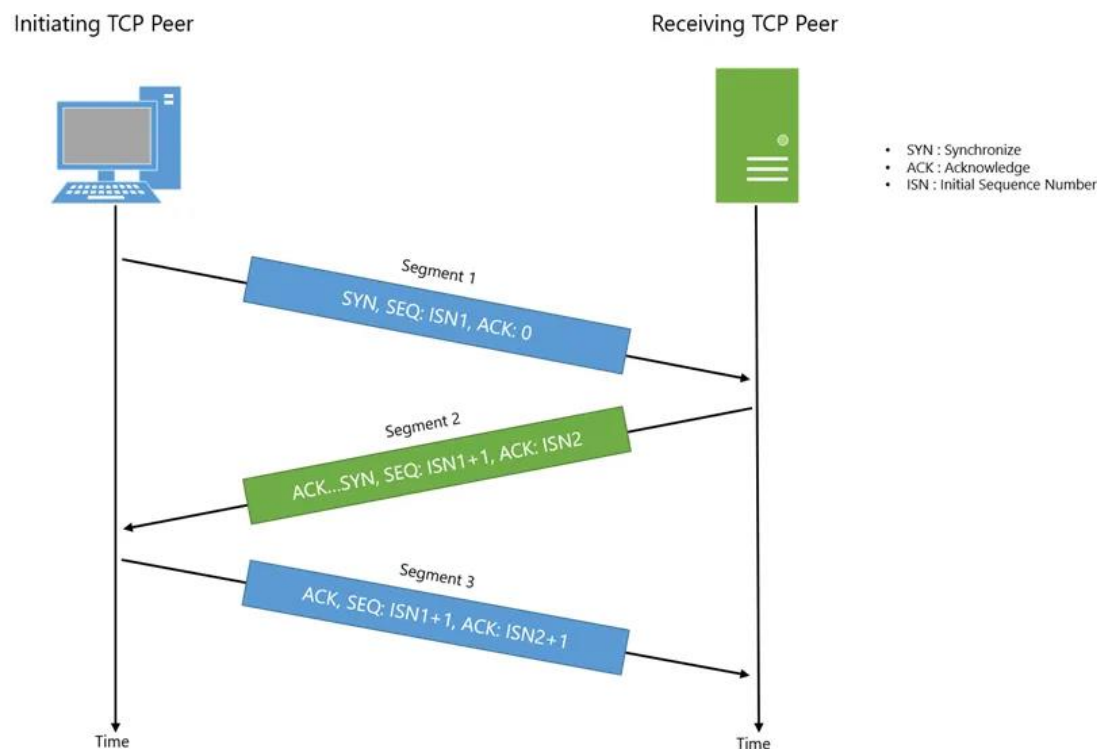
La diferencia en el TTL entre Windows y Linux es que Windows 7 fija un TTL de 128, mientras que Linux lo fija en 64.



Tarea 8 Análisis de una conexión TCP

Realizamos la conexión a la URL <http://www.example.com> y capturamos los paquetes que corresponden a esta conexión.

Imagen explicativa del three-way handshake



El primer paquete que corresponde a nuestra conexión es una petición DNS desde nuestra IP (10.21.1.7) a la dirección 10.45.1.2 (dirección del servidor DNS local). A continuación, este nos responde con la dirección IP de la página que hemos buscado, que es 93.184.216.34.

No.	Time	Source	Destination	Protocol	Length	Info
10	0.006428036	10.21.1.7	10.45.1.2	TCP	60	66 [RTP ACKed unseen segment] 443 → 46722 [ACK] Seq=1 Ack=2 Win=256 Len=0 TSval=431211923 TSecr=3320124776
11	0.289962470	Giga-Byt de:09:9e	Broadcast	ARP	60	60 Who has 10.21.17.7? Tell 10.21.11.10
12	0.89866449	10.21.0.2	224.0.0.18	VRRP	60	60 Announcement (v2)
13	1.314414798	Giga-Byt de:09:9e	Broadcast	ARP	60	60 Who has 10.21.17.7? Tell 10.21.11.10
14	1.899193790	10.21.0.2	224.0.0.18	VRRP	60	60 Announcement (v2)
15	2.086030524	173.194.76.189	10.21.1.7	TLSv1.2	119	Application Data
16	2.086091948	10.21.1.7	173.194.76.189	TCP	66	66 45472 → 443 [ACK] Seq=1 Ack=54 Win=264 Len=0 TSval=2179794248 TSecr=431212131
17	2.266563675	Enterasy 7d:b8:95	Spanning-tree (for-...	STP	60	60 RST - Root = 32769/0/00-01:14:7d:b8:80 Cost = 0 Port = 0x8016
18	2.338693274	Giga-Byt de:09:9e	Broadcast	ARP	60	60 Who has 10.21.17.7? Tell 10.21.11.10
19	2.898943956	10.21.0.2	224.0.0.18	VRRP	60	60 Announcement (v2)
20	3.362516556	Giga-Byt de:09:9e	Broadcast	ARP	60	60 Who has 10.21.17.7? Tell 10.21.11.10
21	3.898944004	10.21.0.2	224.0.0.18	VRRP	60	60 Announcement (v2)
22	4.705502283	10.21.1.7	172.217.17.4	TLSv1.2	250	Application Data
23	4.706173201	172.217.17.4	10.21.1.7	TCP	66	66 443 → 39924 [ACK] Seq=1 Ack=185 Win=1277 Len=0 TSval=431212393 TSecr=4283643009
24	4.719336843	10.21.1.7	10.45.1.2	DNS	75	Standard query 0x11c0 A www.example.com
25	4.715688362	10.45.1.2	10.21.1.7	DNS	195	Standard query response 0x11c0 A www.example.com A 93.184.216.34 NS b.iana-servers.net NS a.iana-servers.net AAAA 2001:500:8d::53 AAAA 2001:500:8f::53
26	4.716059057	10.21.1.7	93.184.216.34	TCP	74	59194 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1293541986 TSecr=0 WS=128
27	4.718452555	93.184.216.34	10.21.1.7	TCP	74	80 → 59194 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=431212395 TSecr=1293541986 WS=128
28	4.718500932	10.21.1.7	93.184.216.34	TCP	66	59194 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1293541988 TSecr=431212395
29	4.719167209	10.21.1.7	93.184.216.34	HTTP	485	GET / HTTP/1.1
30	4.719930396	93.184.216.34	10.21.1.7	TCP	66	80 → 59194 [ACK] Seq=1 Ack=420 Win=15616 Len=0 TSval=431212395 TSecr=1293541989
31	4.750072363	172.217.17.4	10.21.1.7	TLSv1.2	97	Application Data
32	4.750099481	172.217.17.4	10.21.1.7	TLSv1.2	105	Application Data
33	4.750109131	172.217.17.4	10.21.1.7	TLSv1.2	105	Application Data
34	4.750211567	10.21.1.7	172.217.17.4	TCP	66	39924 → 443 [ACK] Seq=185 Ack=143 Win=3742 Len=0 TSval=4283643054 TSecr=431212398
35	4.751494447	10.21.1.7	172.217.17.4	TLSv1.2	105	Application Data
36	4.752165895	172.217.17.4	10.21.1.7	TCP	66	443 → 39924 [ACK] Seq=143 Ack=224 Win=1278 Len=0 TSval=431212398 TSecr=4283643055
37	4.899243956	10.21.0.2	224.0.0.18	VRRP	60	60 Announcement (v2)
38	4.921464277	93.184.216.34	10.21.1.7	HTTP	1080	HTTP/1.1 200 OK (text/html)
39	4.921488430	10.21.1.7	93.184.216.34	TCP	66	59194 → 80 [ACK] Seq=420 Ack=1015 Win=31232 Len=0 TSval=1293542191 TSecr=431212415
40	4.958086797	Giga-Byt de:09:62	Broadcast	ARP	60	60 Who has 10.21.17.7? Tell 10.21.11.8
41	4.965695527	10.21.1.7	10.45.1.2	DNS	72	Standard query 0x4e91 A www.iana.org
42	4.980105883	10.21.1.7	93.184.216.34	HTTP	421	GET /favicon.ico HTTP/1.1

Entonces se realiza el **three-way handshake**, el cual se basa en una serie de paquetes para establecer la conexión. El lado cliente de una conexión realiza una apertura activa de un puerto enviando un paquete SYN inicial al servidor como parte de la negociación en tres pasos. En el lado del servidor se comprueba si el puerto está abierto, es decir, si existe algún proceso escuchando en ese puerto, pues se debe verificar que el dispositivo de destino tenga este



servicio activo y esté aceptando peticiones en el número de puerto que el cliente intenta usar para la sesión. En caso de no estarlo, se envía al cliente un paquete de respuesta con el bit RST activado, lo que significa el rechazo del intento de conexión. En caso de que sí se encuentre abierto el puerto, el lado servidor respondería a la petición SYN válida con un paquete SYN/ACK. Finalmente, el cliente debería responderle al servidor con un ACK, completando así la negociación en tres pasos (SYN, SYN/ACK y ACK) y la fase de establecimiento de conexión.

Le mandamos un datagrama con protocolo TCP SYN (bit seq=0) a la dirección IP de la página. El servidor nos contesta TCP SYN ACK (bits seq=1 y ack=1). Por último, le contestamos TCP ACK (bits seq=1 y ack=1). De esta forma, queda establecida la conexión.

22	4.785502283	10.21.1.7	172.217.17.4	TLSv1.2	250 Application Data
23	4.786173201	172.217.17.4	10.21.1.7	TCP	66 443 → 39924 [ACK] Seq=1 Ack=185 Win=1277 Len=0 TSval=431212393 TSecr=4283643009
24	4.713636843	10.21.1.7	10.45.1.2	DNS	75 Standard query 0x11c0 A www.example.com
25	4.715688362	10.45.1.2	10.21.1.7	DNS	195 Standard query response 0x11c0 A www.example.com A 93.184.216.34 NS b.iana-servers.net NS a.iana-servers.net AAAA 26
26	4.716050557	10.21.1.7	93.184.216.34	TCP	74 59194 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1293541986 TSecr=0 WS=128
27	4.718452455	93.184.216.34	10.21.1.7	TCP	74 80 → 59194 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=431212395 TSecr=1293541986 WS=128
28	4.718500932	10.21.1.7	93.184.216.34	TCP	66 59194 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1293541988 TSecr=431212395
29	4.719167209	10.21.1.7	93.184.216.34	HTTP	485 GET / HTTP/1.1
30	4.719930396	93.184.216.34	10.21.1.7	TCP	66 80 → 59194 [ACK] Seq=1 Ack=420 Win=15616 Len=0 TSval=431212395 TSecr=1293541989
31	4.750072363	172.217.17.4	10.21.1.7	TLSv1.2	138 Application Data
32	4.750099481	172.217.17.4	10.21.1.7	TLSv1.2	97 Application Data
33	4.750109131	172.217.17.4	10.21.1.7	TLSv1.2	105 Application Data
34	4.750211567	172.217.17.4	172.217.17.4	TCP	66 39924 → 443 [ACK] Seq=185 Ack=143 Win=3742 Len=0 TSval=4283643054 TSecr=431212398
35	4.751494447	10.21.1.7	172.217.17.4	TLSv1.2	105 Application Data

▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
 Total Length: 60

Una vez se ha realizado esto, le mandamos una petición GET para obtener los datos HTTP. El resto de la conversación es la misma a la explicada en la tarea 5.

29	4.719167209	10.21.1.7	93.184.216.34	HTTP	485 GET / HTTP/1.1
30	4.719930396	93.184.216.34	10.21.1.7	TCP	66 80 → 59194 [ACK] Seq=1 Ack=420 Win=15616 Len=0 TSval=431212395 TSecr=1293541989
31	4.750072363	172.217.17.4	10.21.1.7	TLSv1.2	138 Application Data
32	4.750099481	172.217.17.4	10.21.1.7	TLSv1.2	97 Application Data
33	4.750109131	172.217.17.4	10.21.1.7	TLSv1.2	105 Application Data
34	4.750211567	10.21.1.7	172.217.17.4	TCP	66 39924 → 443 [ACK] Seq=185 Ack=143 Win=3742 Len=0 TSval=4283643054 TSecr=431212398
35	4.751494447	10.21.1.7	172.217.17.4	TLSv1.2	105 Application Data
36	4.752165095	172.217.17.4	10.21.1.7	TCP	66 443 → 39924 [ACK] Seq=143 Ack=224 Win=1278 Len=0 TSval=431212398 TSecr=4283643055
37	4.899243956	10.21.0.2	224.0.0.18	VRRP	60 Announcement (v2)
38	4.921464277	93.184.216.34	10.21.1.7	HTTP	1066 HTTP/1.1 200 OK (text/html)
39	4.921488430	10.21.1.7	93.184.216.34	TCP	66 59194 → 80 [ACK] Seq=420 Ack=1015 Win=31232 Len=0 TSval=1293542191 TSecr=431212415
40	4.958086797	Giga-Byt de:09:62	Broadcast	ARP	60 Who has 10.21.17.7? Tell 10.21.11.8
41	4.965695527	10.21.1.7	10.45.1.2	DNS	72 Standard query 0x4e91 A www.iana.org
42	4.980105883	10.21.1.7	93.184.216.34	HTTP	421 GET /favicon.ico HTTP/1.1
43	4.980867804	93.184.216.34	10.21.1.7	TCP	66 80 → 59194 [ACK] Seq=1015 Ack=775 Win=16640 Len=0 TSval=431212421 TSecr=1293542250
44	5.083164573	93.184.216.34	10.21.1.7	HTTP	1066 HTTP/1.1 404 Not Found (text/html)

▼ Internet Protocol Version 4, Src: 93.184.216.34, Dst: 10.21.1.7  
 0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes (5)  
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
 Total Length: 1066  
 Identification: 0x93c7 (37831)  
 Flags: 0x4000, Don't fragment  
 Time to live: 60

Por lo tanto, las direcciones IP que intervienen son la nuestra (10.21.1.7), la del servidor DNS local (10.45.1.2) y la de la página a la que nos conectamos (93.184.216.34)

Los puertos que intervienen son: el puerto 59194, que es el puerto que se ha asignado para recibir los paquetes TCP del three-way handshake, y el puerto 80, que es el puerto por defecto para las conexiones TCP de los servidores HTTP.

El paquete más grande es el HTTP 200 OK, que presenta un tamaño de 1066 B. En una conexión, la tecnología que se utiliza define el tamaño máximo de los paquetes que se pueden transferir. Esto se define como MTU (Maximum Transfer Unit). En nuestro caso, la conexión se realiza por Ethernet, el cual tiene un tamaño máximo de 1500 B, mayor que el mayor paquete que hemos transferido. Si el paquete de IP tuviera un tamaño mayor al MTU, se tendría que fragmentar el datagrama IP en partes con un tamaño menor al MTU. Tiene sentido que el más grande sea este ya que contienen el código HTML que le hemos pedido a la página, por lo tanto, ocupará más espacio.

## Tarea 9 Análisis de una petición HTTP

Activamos un servidor apache2 (como realizamos en la práctica anterior). Desde otra máquina, intentamos conectarnos a este servidor y capturamos los paquetes durante la conexión.

El **flag PSH** es un bit que se encuentra dentro del campo del código en el protocolo TCP. Cuando este flag 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. Esto ocurre porque a veces llegan varios segmentos que transportan datos y no tienen activado el bit PSH. Entonces el receptor almacenará esos datos, pero no los entregará a la aplicación receptora hasta que reciba un segmento con el PSH activado.

No.	Time	Source	Destination	Protocol	Length	Info
37	2.304705714	EncantoN 01:06:01	Broadcast	ARP	60	Who has 10.21.1.7? Tell 10.21.1.6
38	2.304718866	EncantoN 01:07:01	EncantoN 01:06:01	ARP	42	10.21.1.7 is at 00:10:21:01:07:01
39	2.305362191	10.21.1.6	10.21.1.7	TCP	74	80 → 53050 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=4138738955 TSecr=2694502343 WS=128
40	2.305400738	10.21.1.7	10.21.1.6	TCP	66	53050 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2694502345 TSecr=4138738955
41	2.305530898	151.101.241.194	10.21.1.7	TCP	66	443 → 45080 [FIN, ACK] Seq=1 Ack=2 Win=132 Len=0 TSval=431370493 TSecr=1099792619
42	2.305560664	10.21.1.7	151.101.241.194	TCP	66	45080 → 443 [ACK] Seq=2 Ack=2 Win=1175 Len=0 TSval=1099792621 TSecr=431370493
43	2.305772472	10.21.1.7	10.21.1.6	HTTP	479	GET / HTTP/1.1
44	2.306392000	10.21.1.6	10.21.1.7	TCP	66	80 → 53050 [ACK] Seq=1 Ack=414 Win=30080 Len=0 TSval=4138738957 TSecr=2694502346
45	2.331142916	10.21.1.6	10.21.1.7	HTTP	3446	HTTP/1.1 200 OK (text/html)
46	2.331205573	10.21.1.7	10.21.1.6	TCP	66	53050 → 80 [ACK] Seq=414 Ack=3381 Win=35968 Len=0 TSval=2694502371 TSecr=4138738981
47	2.372188868	10.21.1.7	10.21.1.6	HTTP	419	GET /icons/openlogo-75.png HTTP/1.1
48	2.384583487	10.21.1.6	10.21.1.7	TCP	2962	80 → 53050 [ACK] Seq=3381 Ack=767 Win=31104 Len=2896 TSval=4138739035 TSecr=2694502412 [TCP segment of a reassembled PDU]
49	2.384713169	10.21.1.7	10.21.1.6	TCP	66	53050 → 80 [ACK] Seq=767 Ack=6277 Win=41856 Len=0 TSval=2694502424 TSecr=4138739035
50	2.384882528	10.21.1.6	10.21.1.7	HTTP	3210	HTTP/1.1 200 OK (PNG)
51	2.384897279	10.21.1.7	10.21.1.6	TCP	66	53050 → 80 [ACK] Seq=767 Ack=9421 Win=48128 Len=0 TSval=2694502425 TSecr=4138739035
52	2.391207946	10.21.1.7	10.45.1.2	DNS	75	Standard query 0xc370 A bugs.debian.org
53	2.391448556	10.21.1.7	10.45.1.2	DNS	76	Standard query 0xb7d6 A httpd.apache.org
54	2.393353501	10.45.1.2	10.21.1.7	DNS	328	Standard query response 0xc370 A bugs.debian.org A 209.87.16.39 A 140.211.166.201 NS dns4.easydns.info NS sec1.rcode0.net NS
55	2.393373005	10.45.1.2	10.21.1.7	DNS	303	Standard query response 0xb7d6 A httpd.apache.org A 80.70.70.1 A 85.216.24.32 NS ns3.cloudflare.nl NS ns1.no-ip.com NS ns4.no-
Source Port: 80 Destination Port: 53050 [Stream index: 8] [TCP Segment Len: 3380] Sequence number: 1 (relative sequence number) [Next sequence number: 3381 (relative sequence number)] Acknowledgment number: 414 (relative ack number) 1000 .... = Header Length: 32 bytes (8) ▼ Flags: 0x018 (PSH, ACK) 0000 .... = 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 ....1 .... = Push: Set ....0 .... = Reset: Not set ....0 .... = Syn: Not set ....0 .... = Fin: Not set [TCP Flags: .....AP...]						
0020	01 07 00 50 cf 3a bb 1a 10 9e 56 b9 52 44 80 18	...P:..-V-RD:1				
0030	00 eb 23 91 00 00 01 01 08 0a f6 b0 25 25 a0 9a	..#.....-%%..				
0040	d7 ca 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f	..HTTP/1.1 200 0				
0050	4b 0d 0a 44 61 74 65 3a 20 54 75 65 2c 20 31 37	K-Date: Tue, 17				
0060	20 44 65 63 20 32 30 31 39 20 31 30 3a 33 39 3a	Dec 201 9 10:39:				
0070	32 30 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20	20 GMT-Server:				
0080	41 70 61 63 68 65 2f 32 2e 3a 2e 33 38 20 28 44	Apache/2.4.38 (D				
0090	65 62 69 61 6e 29 0d 0a 4c 61 73 74 2d 4d 6f 64	ebian)-Last-Mod				
00a0	69 66 69 65 64 3a 20 57 65 64 2c 20 32 37 20 4e	ified: Wed, 27 N				
00b0	6f 76 20 32 30 31 39 20 30 39 3a 35 34 3a 30 37	ov 2019 09:54:07				
00c0	20 47 4d 54 0d 0a 45 54 61 67 3a 20 22 32 39 63	GMT-ETag: "29c				
00d0	64 2d 35 39 38 35 30 66 61 3a 38 38 64 35 31 2d	d-59850fa488d51-				
00e0	67 7a 69 70 22 0d 0a 41 63 63 65 70 74 2d 52 61	gzip"-Accept-Ra				
00f0	6e 67 65 73 3a 20 62 79 74 65 73 0d 0a 56 61 72	nges: bytes Var				
0100	79 3a 20 41 63 63 65 70 74 2d 45 6e 63 6f 64 69	y: Accept-Encodi				
0110	6e 67 0d 0a 43 6f 6e 74 65 6e 74 2d 45 6e 63 6f	ng-Content-Enco				
0120	64 69 6e 67 3a 20 67 7a 69 70 0d 0a 43 6f 6e 74	ding: gzip-Cont				
0130	65 6e 74 2d 4c 65 6e 67 74 68 3a 20 33 30 34 31	ent-Leng th: 3041				
0140	0d 0a 4b 65 65 70 2d 41 6c 69 76 65 3a 20 74 69	-Keep-A live: ti				

Nosotros hemos encontrado el flag PSH activado (1) en el paquete HTTP/1.1 200 OK. Esto se corresponde con lo explicado anteriormente ya que el **código 200 OK** (código de estado de la comunicación HTTP) indica que la solicitud ha tenido éxito. En nuestro caso, al ser una petición de http mediante un GET, indica que se ha solicitado un recurso que se ha encontrado y nos lo ha respondido. Así, el flag PSH nos está indicando que esta información se debe enviar al receptor.

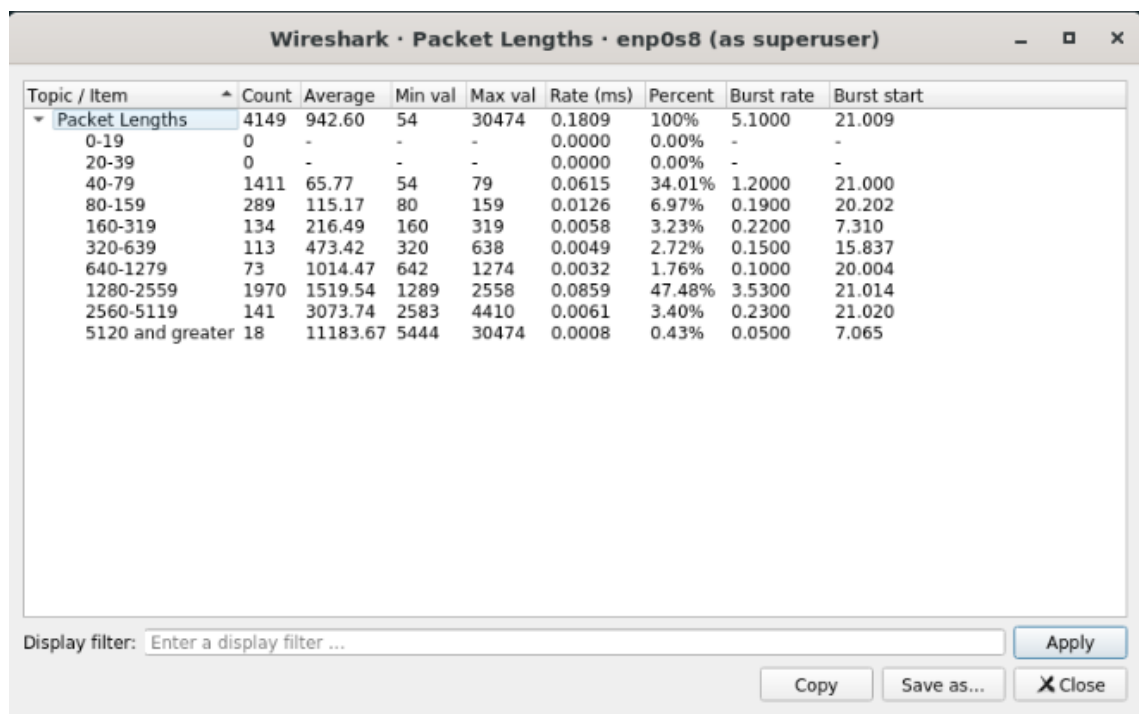
## Tarea 10 Herramientas de estadística

Realizamos una captura con WireShark, mientras que accedemos a 3 páginas aleatorias de internet. Probaremos 4 herramientas estadísticas.

### *Packet Lengths*

Esta herramienta nos ofrece un resumen en forma de tabla de los tamaños de los paquetes que se han esnifado. La herramienta define unos rangos de tamaño de paquete y los clasifica ofreciendo datos estadísticos sobre los rangos (porcentaje, tamaño medio, porcentaje respecto al total).

Con burst rate, la herramienta nos indica el número máximo de paquetes enviados por intervalo de tiempo, el momento en el que se envió el mayor número de paquetes, y este intervalo de tiempo se puede modificar desde (burst rate resolution).



The image shows the 'Wireshark · Packet Lengths · enp0s8 (as superuser)' window. It displays a table of packet length statistics. The table has columns for Topic / Item, Count, Average, Min val, Max val, Rate (ms), Percent, Burst rate, and Burst start. The data is organized into rows representing different packet size ranges. At the bottom, there is a 'Display filter' input field and buttons for 'Apply', 'Copy', 'Save as...', and 'Close'.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Packet Lengths	4149	942.60	54	30474	0.1809	100%	5.1000	21.009
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	1411	65.77	54	79	0.0615	34.01%	1.2000	21.000
80-159	289	115.17	80	159	0.0126	6.97%	0.1900	20.202
160-319	134	216.49	160	319	0.0058	3.23%	0.2200	7.310
320-639	113	473.42	320	638	0.0049	2.72%	0.1500	15.837
640-1279	73	1014.47	642	1274	0.0032	1.76%	0.1000	20.004
1280-2559	1970	1519.54	1289	2558	0.0859	47.48%	3.5300	21.014
2560-5119	141	3073.74	2583	4410	0.0061	3.40%	0.2300	21.020
5120 and greater	18	11183.67	5444	30474	0.0008	0.43%	0.0500	7.065

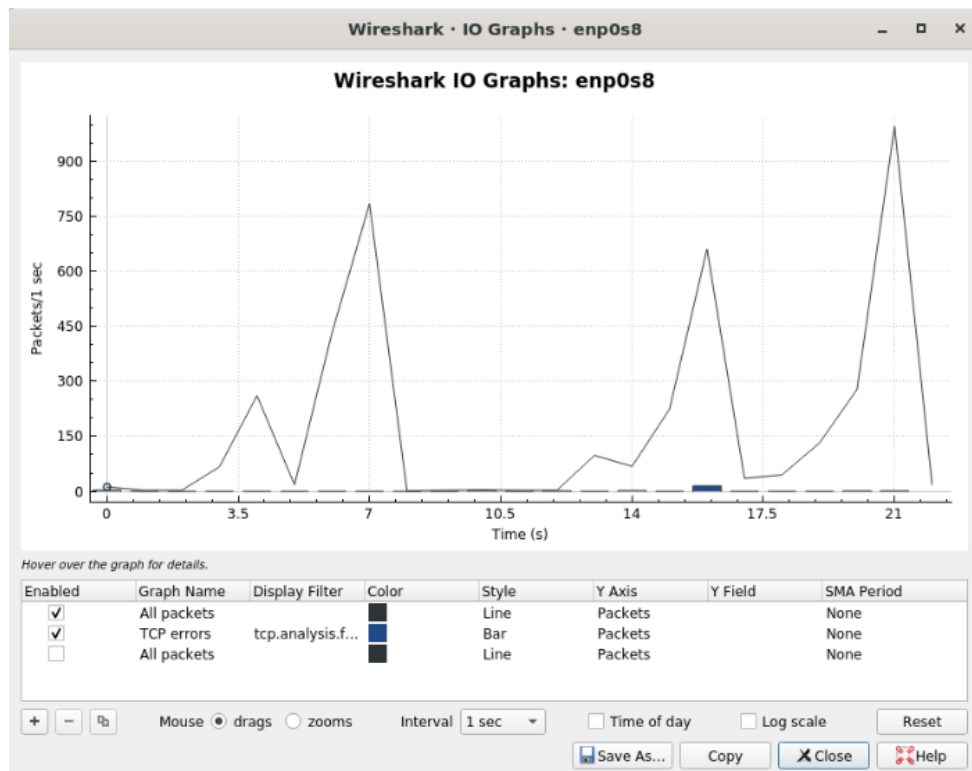
## Flow Graph

Esta herramienta nos representa gráficamente el flujo de paquetes entre emisores y receptores. Nos muestra el camino entre la IP local hasta las diferentes IPs con las que se han interactuado (incluyendo los destinos de broadcast). Nos indica los nombres de las máquinas que han recibido o enviado los paquetes, así como los puertos de entrada y de salida.



## IO Graphs

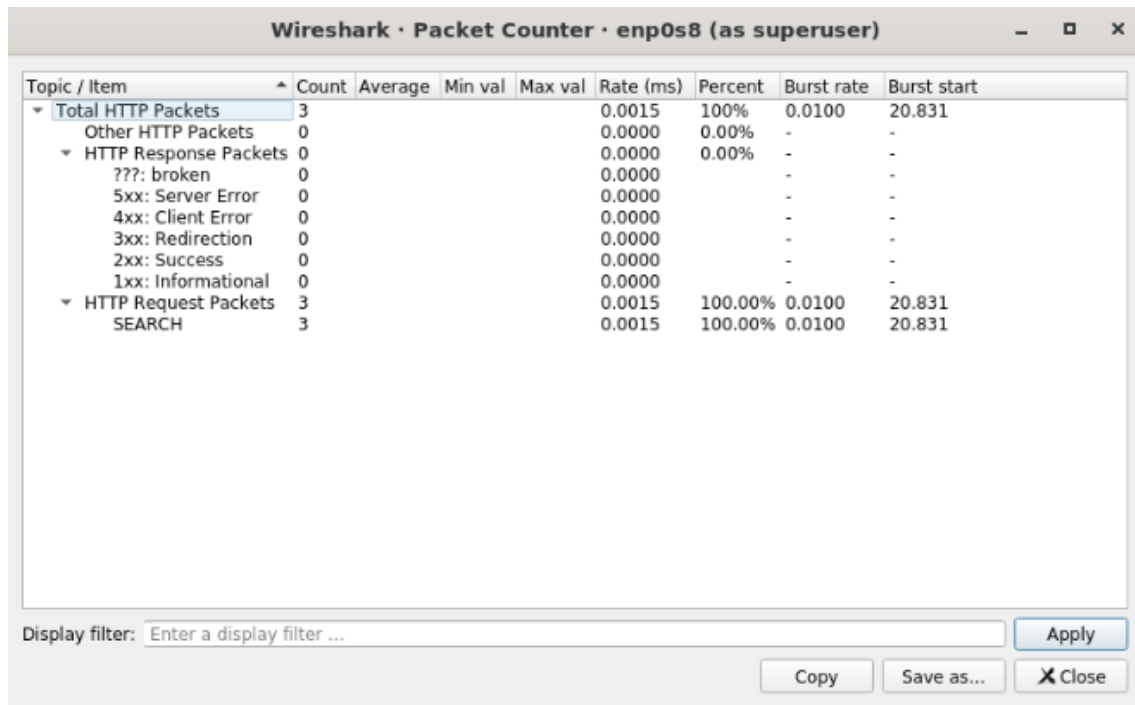
Esta herramienta nos muestra el tráfico general realizado durante una captura que se mide generalmente en velocidad: paquetes o bytes por segundo. Por defecto, encontramos en el eje X los segundos, y en Y la velocidad. Esta herramienta sirve para ver la dinámica del tráfico más de cerca y poder detectar errores de picos y caídas en dicho tráfico.



### HTTP / Packet Counter

Esta herramienta nos permite evaluar estadísticamente los paquetes HTTP esnifados, y nos ofrece un recuento de los paquetes y los tipos de paquetes como error o éxito, como la señal 200 OK vista anteriormente.

Esta herramienta nos divide los paquetes de petición según el protocolo, en este caso solo se realizaron peticiones GET al acceder a las 3 webs.



The image shows the 'Wireshark · Packet Counter · enp0s8 (as superuser)' window. It displays a table of HTTP statistics. The table has columns for Topic / Item, Count, Average, Min val, Max val, Rate (ms), Percent, Burst rate, and Burst start. The data is organized into a tree structure under 'Total HTTP Packets'.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Total HTTP Packets	3				0.0015	100%	0.0100	20.831
Other HTTP Packets	0				0.0000	0.00%	-	-
▼ HTTP Response Packets	0				0.0000	0.00%	-	-
??? : broken	0				0.0000	-	-	-
5xx: Server Error	0				0.0000	-	-	-
4xx: Client Error	0				0.0000	-	-	-
3xx: Redirection	0				0.0000	-	-	-
2xx: Success	0				0.0000	-	-	-
1xx: Informational	0				0.0000	-	-	-
▼ HTTP Request Packets	3				0.0015	100.00%	0.0100	20.831
SEARCH	3				0.0015	100.00%	0.0100	20.831

At the bottom of the window, there is a 'Display filter:' field with the placeholder text 'Enter a display filter ...'. To the right of this field are three buttons: 'Apply', 'Copy', and 'Save as...'. At the bottom right corner, there is a 'Close' button with a red X icon.