

Presentación

¿Sabías que muchas empresas de Internet como Twitter, Facebook, Flickr o Wikipedia tienen una API – un conjunto de definiciones y protocolos para que dos aplicaciones puedan interactuar – desde la que podemos descargar datos? Por ejemplo, datos sobre qué usuarios están twitteando o sobre qué temas se están twitteando.

En esta actividad se te propone utilizar una web API con el objetivo puesto en el análisis de protocolos de red. Es decir, no nos fijaremos en el tratamiento de los datos, sino que usaremos una herramienta de software libre muy popular entre los técnicos de red (*Wireshark*). Esta herramienta nos permitirá detectar el tráfico (o hacer *traffic sniffing* en inglés) y poder así analizar los paquetes que pasan por la interfaz de red de nuestro ordenador. A su vez, os planteamos una serie de preguntas teóricas sobre los contenidos del reto.

Competencias

En esta PEC se trabajan las siguientes competencias del Grado de Ciencia de Datos Aplicada:

- Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
- Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
- Uso y aplicación de las TIC en el ámbito académico y profesional.
- Administrar y gestionar los sistemas operativos y de comunicaciones de los componentes de una red de ordenadores.

Objetivos

Los objetivos específicos de esta PEC son:

- Ser capaz de comprender las funciones asociadas a las capas superiores del modelo OSI (transporte y aplicación) y diferenciarlas de las destacadas en el reto anterior.
- Entender la importancia de las redes como herramienta para obtener datos.
- Conocer las principales aplicaciones orientadas a Internet.
- Conseguir analizar un flujo de datos y encontrar los elementos estudiados en los tres primeros retos.
- Dominar el uso de una herramienta para analizar protocolos de red.

Descripción de la PEC a realizar

Esta PEC hará que pongas en práctica los conocimientos y procedimientos asociados a este reto. Partiremos del mismo navegador que utilizamos cada día para navegar por Internet y teclearemos una dirección web para analizar qué pasa en la red todas las veces que ejecutamos una operación tan básica y cotidiana. Seguidamente, haremos lo mismo accediendo a una API de la que podríamos decidir extraer datos. En esta PEC no nos centraremos en la extracción de datos, que ya se tratará en otros cursos/asignaturas, sino que sólo observaremos y analizaremos lo que pasa en la red.

En cuanto a los ejercicios de esta PEC, se combinan preguntas teóricas referentes a los principales conceptos de este reto, junto con cuestiones referentes al análisis de red a través de una herramienta de software libre: Wireshark.

NOTA: El navegador Google Chrome suele utilizar un protocolo particular llamado QUIC que puede dificultarnos la visión de los conceptos a tratar. Por ello, se recomienda usar otros navegadores (p. ej. Mozilla Firefox). En caso de que queráis usar Google Chrome, os dejamos [este enlace](#) que explica cómo deshabilitar QUIC.

Recursos

Recursos Básicos

- Las redes como medio para obtener datos

- Las capas inferiores del modelo OSI
- El nivel de red
- El nivel de transporte
- Las capas superiores del modelo OSI
- El papel de las redes de computadoras en el ciclo de vida de los datos

Recursos Complementarios

- Manual de Wireshark
- Captura Wireshark 'sip-rtp-g711.pcap'
- Computer Networking: A Top-Down Approach, Kurose, 7th edition

Criterios de valoración

- La PEC debe resolverse de manera individual.
- Es necesario justificar todas las respuestas a los ejercicios propuestos en la PEC.
- La puntuación (sobre un total de 10 puntos) asociada a cada ejercicio de esta PEC se indica al inicio de cada enunciado.

Los ejercicios de esta PEC constituyen la parte correspondiente del cómputo de la evaluación continua de la asignatura: $AC = 0,2 \cdot PEC1 + 0,3 \cdot PEC2 + 0,3 \cdot PEC3 + 0,2 \cdot PEC4$

Para más información sobre el modelo de evaluación de la asignatura os remitimos al Plan Docente.

Formato y fecha de entrega

La entrega de esta PEC constará de un archivo .pdf siguiendo el formato específico **PEC3Apellido1Apellido2Nombre.pdf** e incluirá:

- La memoria de la práctica en formato .pdf que contenga las respuestas correspondientes a todos los Ejercicios planteados.

Este archivo .pdf deberá enviarse a través de la herramienta REC (Registro de Evaluación Continua) del aula **antes de las 23:59 del día 10/12/2021**.

IMPORTANTE: Recordad que la PEC es individual. La detección de falta de originalidad será penalizada conforme a la normativa vigente de la UOC. Además, al hacer la entrega aseguraos de comprobar que el fichero entregado es el correcto, pues es responsabilidad del alumno realizar las entregas correctamente. No se aceptarán entregas fuera de plazo.

1. Primer contacto con Wireshark

1.1. Iniciar Wireshark

Wireshark se ha convertido en el principal analizador de red utilizado por los ingenieros de redes. Este software de código abierto está disponible para muchos sistemas operativos diferentes, incluidos Windows, MAC y Linux. Wireshark se puede descargar desde www.wireshark.org.

Utiliza el recurso *Manual de Wireshark* para instalar y obtener una primera visión de conjunto del aplicativo. Elige la versión de software que necesites según la arquitectura y el sistema operativo de tu PC o laptop. Por ejemplo, si tienes un PC de 64 bits con Windows, selecciona Windows Installer (64-bit) (Instalador de Windows [64 bits]).

Para capturar datos de la red activa, Wireshark requiere la instalación de WinPcap o de Npcap. Si WinPcap o Npcap ya están instalados en el PC, la casilla de verificación Install (Instalar) estará desactivada. Si la versión instalada de estos programas es anterior a la versión que incluye Wireshark, se recomienda que permitas que la versión más reciente se instale haciendo clic en la casilla de verificación Install WinPcap o Npcap.

1.2. Selecciona una interfaz de red

Una vez finalizada la instalación, haz doble clic en el icono de Wireshark en tu escritorio o navega hasta el directorio que contiene el archivo ejecutable e inicia Wireshark. En la pantalla inicial de Wireshark se listarán todas las interfaces de red con las que el Wireshark puede capturar datos. Selecciona la interfaz por la cual estás conectado a Internet. Es muy posible que estés conectado a través de Wi-Fi. En ese caso, selecciona la tarjeta de red Wi-Fi. En caso de que estés conectado a través de Ethernet, selecciona la interfaz correspondiente.

Los nombres de las interfaces dependerán del modelo de tarjetas que lleven vuestros equipos. Una manera de ver qué interfaces nos están proveyendo de tráfico es a través de la pequeña gráfica situada justo al lado del nombre de las interfaces listadas justo al abrir Wireshark. En las interfaces con tráfico se apreciarán curvas variantes, mientras que en las que no tienen tráfico las gráficas se mantendrán constantes en el origen. En el ejemplo de la Figura 1 se observa actividad en la interfaz de Wi-Fi *Wi-Fi 2*, que es la que nos provee de conexión a Internet.¹

En el menú **Capture**, puedes elegir en la opción *Options* algunos parámetros para especificar la captura. Por ejemplo, en la pestaña *Options*, puedes configurar que la captura se pare automáti-

¹Obviad la interfaz de red virtual de loopback.

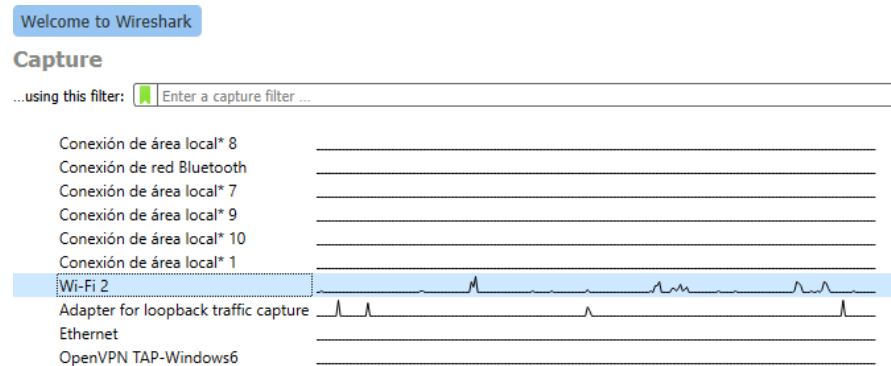


Figura 1: Ejemplo de posibles interfaces listadas en Wireshark.

camente después de un cierto tiempo o después de haber capturado una cierta cantidad de datos. Cuando pulses sobre *Start*, el programa empezará a capturar todos los paquetes que pasen a través de la interfaz que hayas seleccionado. Seguramente observaremos que, aunque no estemos haciendo nada en particular, el programa ya empieza a capturar paquetes. Veamos qué es lo que Wireshark nos está mostrando:

1. *No.:* La primera columna representa un identificador de paquete generado por el programa para tener un orden secuencial de los paquetes en base al orden de llegada.
2. *Time:* La segunda columna muestra el tiempo (por defecto, en segundos) que transcurre entre que se captura un paquete y el tiempo origen (considerado como 0). Por ejemplo, una entrada que indica un *Time* de 3 segundos corresponde a un paquete capturado 3 segundos después de iniciar la captura (es decir, de hacer clic en *Start*).
3. *Source:* La tercera columna muestra la dirección IP que ha generado el paquete.
4. *Destination:* La cuarta columna muestra la dirección IP a la que va dirigido el paquete.
5. *Protocol:* La quinta columna muestra el protocolo asociado al paquete.
6. *Length:* La sexta columna muestra la longitud del paquete en bytes.
7. *Info:* La última columna describe información adicional de los protocolos de comunicación utilizados, como por ejemplo valores específicos de ciertas cabeceras de interés.

En la casilla *Apply a display filter*, ubicada en la parte superior de la pantalla principal de Wireshark, podemos introducir filtros de visualización que nos ayuden a leer e interpretar los datos. Algunos de los filtros de visualización más comunes de Wireshark y que utilizaremos en esta práctica se listan a continuación a través de ejemplos. Puedes encontrar más información sobre los filtros de visualización en: <https://gitlab.com/wireshark/wireshark/-/wikis/DisplayFilters>.

- `ip.addr==192.168.0.1`: Filtra todo el tráfico con dirección IP 192.168.0.1 como origen o destino.
- `tcp.port==80`: Filtra todo el tráfico con el puerto 80 como origen o destino.
- `ip.src==192.168.0.1 and ip.dst==10.100.1.1`: Filtra todo el tráfico que tiene como origen 192.168.0.1 y destino 10.100.1.1.
- `icmp`: Filtra sólo el tráfico del protocolo ICMP.
- `http`: Filtra sólo el tráfico del protocolo HTTP.
- `dns`: Filtra sólo el tráfico del protocolo DNS.
- `sip`: Filtra sólo el tráfico del protocolo SIP.

1.3. ICMP y Ping

Empecemos con un primer ejercicio para contextualizar.

Ejercicio 1 [0,25p]: *¿Cuáles son las direcciones MAC e IPv4 de tu ordenador? Realiza una captura de pantalla del símbolo de sistema (terminal), donde se puedan leer todas las direcciones IPv4 y MAC de tu equipo. ¿Cuál es el comando que has utilizado? Recuerda que para acceder al símbolo de sistema, si estás usando un sistema operativo Windows, escribe en el buscador cmd y presiona Enter. En MacOS puedes abrir la carpeta Aplicaciones, luego Utilidades y hacer doble clic en Terminal.*

Mediante el comando `ipconfig /all` Podemos obtener la dirección IPv4 192.168.64.1 y la dirección MAC o dirección física 50-EB-71-F8-87-97

```

Símbolo del sistema
Configuración automática habilitada . . . : sí

Adaptador de Ethernet Conexión de red Bluetooth 2:

Estado de los medios. . . . . : medios desconectados
Sufijo DNS específico para la conexión. . :
Descripción . . . . . : Bluetooth Device (Personal Area Network) #2
Dirección física. . . . . : 50-EB-71-F8-87-97
DHCP habilitado . . . . . : sí
Configuración automática habilitada . . . : sí

Adaptador de Ethernet vEthernet (Default Switch):

Sufijo DNS específico para la conexión. . :
Descripción . . . . . : Hyper-V Virtual Ethernet Adapter
Dirección física. . . . . : 00-15-5D-B5-55-DF
DHCP habilitado . . . . . : no
Configuración automática habilitada . . . : sí
Vínculo: dirección IPv6 local. . . : fe80::fc13:9bdf:50eb:9097%70(Preferido)
Dirección IPv4. . . . . : 192.168.64.1(Preferido)
Máscara de subred . . . . . : 255.255.240.0
Puerta de enlace predeterminada . . . . :
IAID DHCPv6 . . . . . : 1174410589
DUID de cliente DHCPv6. . . . . : 00-01-00-01-27-A2-0D-5E-A8-5E-45-51-95-07
Servidores DNS. . . . . : fec0:0:0:ffff::1%1
                          fec0:0:0:ffff::2%1
                          fec0:0:0:ffff::3%1
NetBIOS sobre TCP/IP. . . . . : habilitado

C:\Users\aland>

```

Sigamos con una primera captura de los paquetes generados al realizar un ping. En concreto, vamos a realizar un ping desde el símbolo de sistema (o terminal). Una vez abierta la consola, ejecuta el comando `ping www.uoc.edu` (o la página web que prefieras). Si estás utilizando Linux o MacOS, debes abrir el terminal y ejecutar el comando `ping -c 4 www.uoc.edu` (donde el parámetro `c` del comando `ping` permite especificar el número de paquetes ping a enviar). Después de haber ejecutado el ping correctamente, se puede detener la captura en Wireshark pulsando el botón rojo que indica *Stop*.

Analicemos ahora el intercambio de mensajes ICMP en Wireshark al realizar un ping. Escribe `icmp` en el campo *Filter* en la ventana principal de Wireshark. A continuación, haz clic en *Apply* a la derecha del filtro. Wireshark pasa a mostrar solo los mensajes relativos al protocolo ICMP.

Ejercicio 2 [0,25p]: ¿Cuántos paquetes ICMP se observan? Adjunta una captura de pantalla al respecto. ¿Cómo se identifica un mensaje ICMP en el datagrama IP? ¿Cuál es el tamaño de las peticiones que manda por defecto el comando ping? Indica cuál sería el comando para generar 6 peticiones de 64 bytes cada una.

Se obtienen 4 paquetes del tipo request con sus correspondientes 4 paquetes de tipo replay

The screenshot shows the Wireshark interface with the filter `icmp` applied. The packet list displays 8 packets, alternating between requests and replies. The packet details for packet 456 (a request) are expanded, showing the Ethernet II header, the Internet Protocol Version 4 header, and the ICMP Echo (ping) request structure. The ICMP structure shows an ID of 0x0001, a sequence number of 77, and a TTL of 128. The payload is a 64-byte ping request.

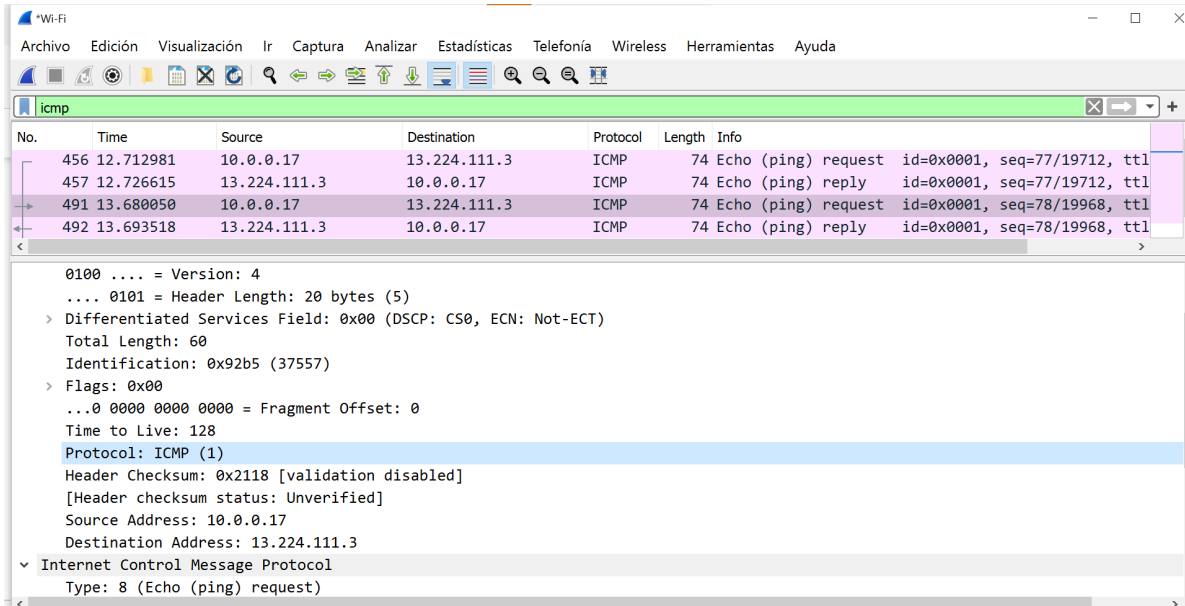
No.	Time	Source	Destination	Protocol	Length	Info
456	12.712981	10.0.0.17	13.224.111.3	ICMP	74	Echo (ping) request id=0x0001, seq=77/19712, ttl=128
457	12.726615	13.224.111.3	10.0.0.17	ICMP	74	Echo (ping) reply id=0x0001, seq=77/19712, ttl=239
491	13.680050	10.0.0.17	13.224.111.3	ICMP	74	Echo (ping) request id=0x0001, seq=78/19968, ttl=128
492	13.693518	13.224.111.3	10.0.0.17	ICMP	74	Echo (ping) reply id=0x0001, seq=78/19968, ttl=239
493	14.688275	10.0.0.17	13.224.111.3	ICMP	74	Echo (ping) request id=0x0001, seq=79/20224, ttl=128
494	14.701809	13.224.111.3	10.0.0.17	ICMP	74	Echo (ping) reply id=0x0001, seq=79/20224, ttl=239
498	15.697409	10.0.0.17	13.224.111.3	ICMP	74	Echo (ping) request id=0x0001, seq=80/20480, ttl=128
500	15.717605	13.224.111.3	10.0.0.17	ICMP	74	Echo (ping) reply id=0x0001, seq=80/20480, ttl=239

Frame 456: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{AA8A064A-6B55-455F-8D73-DD3FE24A30} Ethernet II, Src: a6:97:f1:64:5d:ab (a6:97:f1:64:5d:ab), Dst: 00:ff:49:a7:a8:69 (00:ff:49:a7:a8:69) Internet Protocol Version 4. Src: 10.0.0.17. Dst: 13.224.111.3

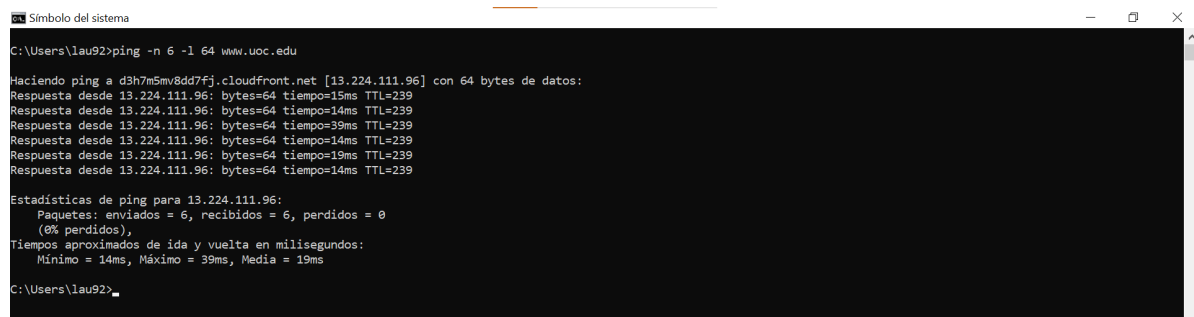
```

0000  00 ff 49 a7 a8 69 a6 97 f1 64 5d ab 08 00 45 00  ..I..i...d]...E.
0010  00 3c 92 b4 00 00 80 01 21 19 0a 00 00 11 0d e0  -<.....!.....
0020  6f 03 08 00 4d 0e 00 01 00 4d 61 62 63 64 65 66  o...M...Mabcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67 68 69  wabcdefg hi
  
```

El header de ICMP comienza después del encabezado IPv4 y se identifica con el número de protocolo IP '1'.



El tamaño de los paquetes enviados al ejecutar el comando ping es por defecto de 32 bytes, podemos modificar este tamaño u otras variables mediante el uso de parámetros, para enviar 6 peticiones de 64 bytes cada una podemos usar el comando : ping -n 6 -l 64, donde el parámetro -n indica el numero de paquetes, y -l indica su longitud .



2. Observar el comportamiento al navegar por Internet

Para garantizar la interoperabilidad entre implementaciones de distintos fabricantes (por ejemplo, para que clientes Web como pueden ser Chrome o Firefox se puedan 'entender' con servidores de Apache o Microsoft), Internet se basa en el uso de protocolos estándares y abiertos - cualquiera puede acceder a la especificación (se encuentran en Internet) e implementarla. La principal organización que se encarga de definir estos estándares para Internet es la Internet Engineering Task Force (IETF). La IETF se organiza en Working Groups (WG) en los que, personas voluntarias, colaboran para definir conjuntamente estos estándares y los publican en las llamadas RFC (Request for Comments). Como ejemplo, el protocolo TCP inicialmente se definió en la RFC número 793, publicado en 1981, y posteriormente se fueron añadiendo funcionalidades adicionales en nuevas RFCs.

Ejercicio 3 [0,25p]: Indica cuáles son los números de RFC para los siguientes protocolos y, para cada uno de ellos, indica el año de publicación:

UDP	RFC 768	agosto 1980
HTTP version 1.0	RFC 1945	mayo 1996
HTTP version 2.0	RFC 7540	mayo 2015
QUIC	RFC 9000	mayo 2021

Tal y como recogen los materiales del curso, mas alla de UDP o TCP existen otros protocolos de transporte como SCTP.

Ejercicio 4 [0,25p]: Indica una característica del protocolo SCTP que lo diferencie de TCP. ¿Qué significa que el protocolo de transporte SCTP tiene la capacidad de multi-homing?

El protocolo SCTP es una alternativa al protocolo al TCP manteniendo sus características de fiabilidad, control de flujo y secuenciación. A diferencia de este puede enviar mensajes desordenadamente similares al protocolo UDP.

Otra diferencia es la propiedad multi-homing que permite una tolerancia de fallos a nivel de red gracias a que se permite que un host comparta a través de múltiples direcciones IP, lo que permite múltiples rutas de IP

Veremos ahora los paquetes generados al abrir una página web. Es momento de introducir el protocolo HTTP.

2.1. Comportamiento del protocolo HTTP

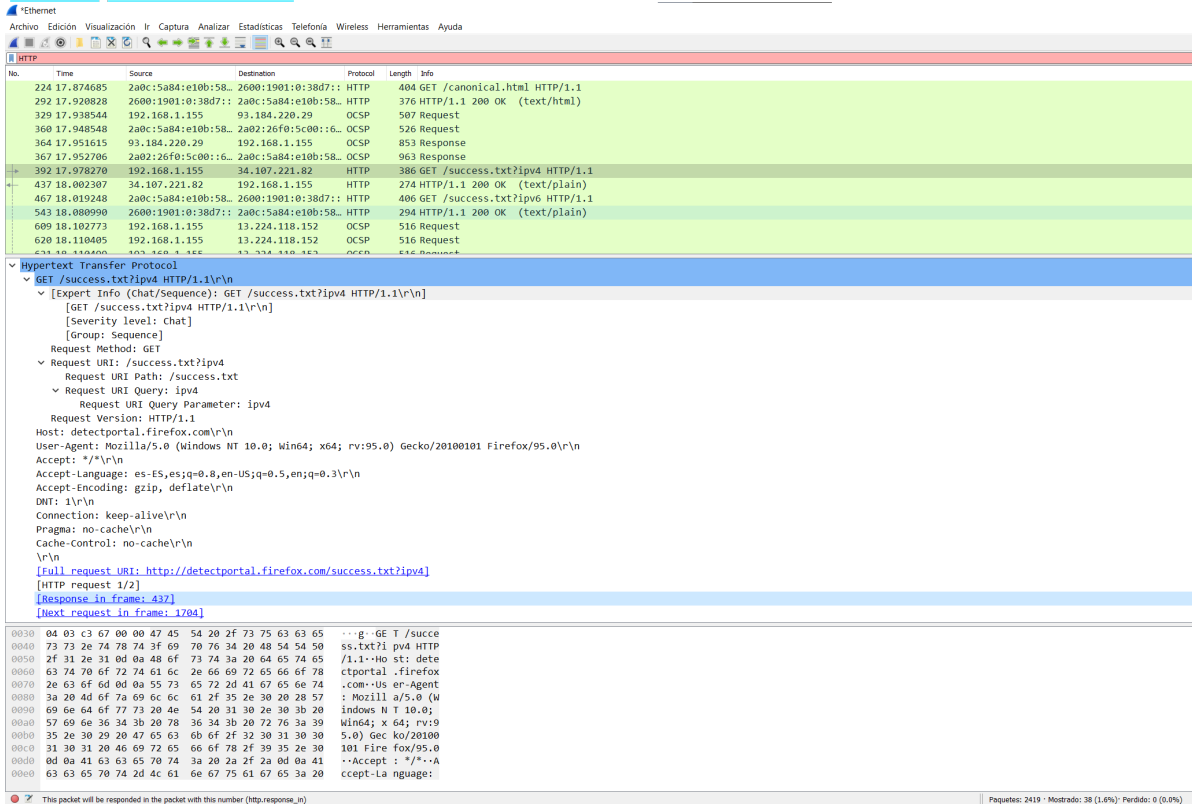
El protocolo HTTP (Hypertext Transfer Protocol) es un protocolo de aplicación para transmitir información en la web a través de una arquitectura cliente-servidor. Para dotar de seguridad al intercambio de datos entre cliente y servidor, el protocolo HTTP se suele transmitir (cada vez más) en combinación con otros protocolos, como por ejemplo el SSL (Secure Sockets Layer) y más recientemente el TLS (Transport Layer Security). A nivel de la arquitectura de red, el protocolo TLS se sitúa entre la capa TCP/IP y el HTTP. Esta combinación da lugar al protocolo HTTPS, también conocido como HTTP over TLS.

Para generar tráfico HTTP tan solo tenemos que introducir en nuestro navegador de Internet habitual la URL de una página web HTTP (y no HTTPS). Un ejemplo de web HTTP es `http://www.washington.edu/`.

Cuando un navegador intenta acceder a una página web realiza una petición a un servidor HTTP. El protocolo HTTP tiene dos métodos principales para que el cliente (navegador) realice esta petición al servidor: el método POST y el método GET. Si todo ha ido bien, tras recibir esta petición, el servidor web donde se encuentra alojada la página web contesta satisfactoriamente a través del mensaje 200 OK. Para ello, encapsula en un mensaje HTTP el código HTML que contiene la información requerida.

Inicia una captura de Wireshark y filtra por protocolo HTTP (para ello, escribe `http` en el campo *Filter* en la ventana principal del Wireshark y haz clic en *Apply*). Verás que al navegar por alguna de estas páginas el número de paquetes HTTP aumenta sensiblemente.² Después de haber capturado paquetes durante aproximadamente un par de minutos detén la captura de Wireshark. Vamos a utilizar esta captura para los ejercicios que siguen [5-11].

Ejercicio 5 [0,75p]: Proporciona una captura de pantalla del resultado del filtro HTTP, donde se pueda ver una petición POST o GET y su respuesta correspondiente. Contesta a las siguientes preguntas: 1) ¿Qué mensaje HTTP es el mensaje de respuesta a la petición GET o POST? 2) ¿Cuánto tiempo ha pasado entre la petición del cliente y la respuesta del servidor? 3) ¿Cuál es el número asignado por Wireshark al mensaje de petición y al de respuesta? 4) ¿Puedes identificar la dirección IP del cliente? ¿Y la dirección IP del servidor? 5) Por último, indica qué versión del protocolo HTTP estamos utilizando.



Wireshark 2.4.19 (64-bit) - Ethernet

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

Filter: HTTP

No.	Time	Source	Destination	Protocol	Length	Info
224	17.874685	2a0c:5a84:e10b:58...	2600:1901:0:38d7::	HTTP	404	GET /canonical.html HTTP/1.1
252	17.920828	2600:1901:0:38d7::	2a0c:5a84:e10b:58...	HTTP	376	HTTP/1.1 200 OK (text/html)
329	17.938544	192.168.1.155	93.184.220.29	OCSP	507	Request
360	17.948548	2a0c:5a84:e10b:58...	2a02:26f0:5c00::16...	OCSP	526	Request
364	17.951615	93.184.220.29	192.168.1.155	OCSP	853	Response
367	17.952706	2a02:26f0:5c00::16...	2a0c:5a84:e10b:58...	OCSP	963	Response
392	17.978270	192.168.1.155	34.107.221.82	HTTP	386	GET /success.txt?ipv4 HTTP/1.1
437	18.002307	34.107.221.82	192.168.1.155	HTTP	274	HTTP/1.1 200 OK (text/plain)
467	18.019248	2a0c:5a84:e10b:58...	2600:1901:0:38d7::	HTTP	406	GET /success.txt?ipv6 HTTP/1.1
543	18.080990	2600:1901:0:38d7::	2a0c:5a84:e10b:58...	HTTP	294	HTTP/1.1 200 OK (text/plain)
609	18.102773	192.168.1.155	13.224.118.152	OCSP	516	Request
620	18.110405	192.168.1.155	13.224.118.152	OCSP	516	Request
631	18.110405	192.168.1.155	13.224.118.152	OCSP	516	Request

Hypertext Transfer Protocol

GET /success.txt?ipv4 HTTP/1.1\r\n

[Expert Info (Chat/Sequence): GET /success.txt?ipv4 HTTP/1.1\r\n]

[GET /success.txt?ipv4 HTTP/1.1\r\n]

[Severity level: Chat]

[Group: Sequence]

Request Method: GET

Request URI: /success.txt?ipv4

Request URI Path: /success.txt

Request URI Query: ipv4

Request URI Query Parameter: ipv4

Request Version: HTTP/1.1

Host: detectportal.firefox.com\r\n

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0\r\n

Accept: */*\r\n

Accept-Language: es-ES,en;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

Pragma: no-cache\r\n

Cache-Control: no-cache\r\n

\r\n

[Full request URI: http://detectportal.firefox.com/success.txt?ipv4]

[HTTP request 1/2]

[Response in frame 437]

[Next request in frame 1704]

This packet will be responded in the packet with this number (http.response_in)

Paquetes: 2419 · Mostrados: 38 (1.6%) · Perdidos: 0 (0.0%)

2Si no consigues capturar ningún paquete HTTP al entrar en estas páginas, borra el historial de cookies de tu navegador e inténtalo de nuevo.

Wireshark capture of an HTTP 200 OK response. The packet list shows a GET request from 192.168.1.155 to 34.107.221.82 on port 80. The packet details pane shows the full HTTP response structure, including status code 200, content type text/plain, and various headers like Server: nginx, Date, and Cache-Control. The packet bytes pane shows the raw data in hexadecimal and ASCII.

- 1) En la línea de estado podemos encontrar la versión del protocolo en este caso HTTP 1.1, además de la respuesta HTTP cuyo código es 200, este quiere decir que es código de respuesta de estado satisfactorio, indica que la solicitud ha tenido éxito.
- 2) El tiempo de respuesta ha sido de 0.02403 segundos
- 3) el número signado por Wireshark al mensaje de petición es 392 y el número asignado al mensaje de respuesta es 437
- 4) la dirección IP del cliente es 192.168.1.155 y la del servidor 34.107.221.82
- 5) Usamos la versión HTTP 1.1

Ejercicio 6 [0,5p]: Selecciona un paquete HTTP de tu captura Wireshark y busca los detalles de la cabecera IP. Incluye una captura de pantalla con los detalles del paquete seleccionado. Ayudándote de los documentos de la capa de red que hemos visto en el reto 2, identifica los elementos de la cabecera IP y descríbelos.

The image shows a Wireshark packet capture analysis of an HTTP GET request. The packet list on the left shows a packet from 192.168.1.155 to 93.184.220.29. The packet details pane on the right shows the structure of the Ethernet II, Internet Protocol Version 4, and Hypertext Transfer Protocol layers. The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII.

Elementos cabecera IP

- 1) Versión : IPv4
- 2) Longitud del encabezado 20bytes
- 3) Tipo de servicio : Differentiated Services Code Point : DEFAULT (0)
- 4) Tamaño del Datagrama : suma del encabezado y los datos : 260 bytes
- 6) Señalador (Flags) se utiliza para controlar si los enrutadores pueden fragmentar un paquete un paquete
- 7) Desplazamiento del fragmento (fragmento offset):0
- 8) Indica la posición del paquete actual mediante 13 bits
- 9) Tiempo de vida: 123 (número de saltos que hacer paquete, disminuido por la mayoría de los enrutadores; se usa para evitar bucles de enrutamiento accidentales
- 10) Protocolo TCP (6) indica el tipo de paquete transportado
- 11) Header cheksum 0xc2c5 : sirve para detectar errores de tranmision en el header del paquete ip
- 12) Direccion de origen 34.107.221.82

13) Direccion de destino 192.168.1.155

Ejercicio 7 [0,25p]: Filtra ahora los mensajes capturados para visualizar sólo los que tengan tu dirección IP como origen e incluye una captura de pantalla.

The screenshot shows the Wireshark interface with the filter 'src==192.168.1.155' applied. The packet list shows several TCP and UDP packets. The selected packet is a TCP ACK from 192.168.1.155 to 52.182.143.210. The packet details pane shows the following information:

- Ethernet II:** Src: ASUSIEK_51:95:07 (aa:5e:45:51:95:07), Dst: zte_a0:55:38 (98:00:6a:a0:55:38)
- Internet Protocol Version 4:** Src: 192.168.1.155, Dst: 52.182.143.210
- Transmission Control Protocol:** Src Port: 60171, Dst Port: 80, Seq: 333, Ack: 221, Len: 332

The packet bytes pane shows the raw data in hexadecimal and ASCII. The ASCII part shows the text 'GET / HTTP/1.1'.

Ejercicio 8 [0,75p]: Manteniendo el filtro de tu dirección IP, identifica la dirección MAC de tu ordenador. ¿Cuál es? En los varios paquetes filtrados, salientes de tu ordenador, identifica también la dirección MAC de destino. ¿A qué dispositivo corresponde? Incluye una captura de pantalla donde se pueda ver la dirección MAC de origen y destino.

Ethernet

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

ip.src == 192.168.1.155

No.	Time	Source	Destination	Protocol	Length	Info
1676	72.116418	192.168.1.155	52.182.143.210	TCP	54	65148 → 443 [ACK] Seq=198 Ack=1453 Win=262144 Len=0
1681	72.116466	192.168.1.155	52.182.143.210	TCP	66	[TCP Dup ACK 1676#1] 65148 → 443 [ACK] Seq=198 Ack=1453 Win=262144 Len=0 SLE=2905 SRE=4357
1682	72.116478	192.168.1.155	52.182.143.210	TCP	66	[TCP Dup ACK 1676#2] 65148 → 443 [ACK] Seq=198 Ack=1453 Win=262144 Len=0 SLE=2905 SRE=5809
1683	72.116489	192.168.1.155	52.182.143.210	TCP	66	[TCP Dup ACK 1676#3] 65148 → 443 [ACK] Seq=198 Ack=1453 Win=262144 Len=0 SLE=2905 SRE=6210
1684	72.116507	192.168.1.155	52.182.143.210	TCP	54	65148 → 443 [ACK] Seq=198 Ack=6210 Win=262144 Len=0
1685	72.126042	192.168.1.155	52.182.143.210	TLSv1...	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
1687	72.261977	192.168.1.155	52.182.143.210	TCP	66	[TCP Dup ACK 1684#1] 65148 → 443 [ACK] Seq=356 Ack=6210 Win=262144 Len=0 SLE=1453 SRE=2905
1689	72.273096	192.168.1.155	52.182.143.210	TCP	54	65148 → 443 [ACK] Seq=356 Ack=6261 Win=261888 Len=0
1690	72.274492	192.168.1.155	52.182.143.210	TLSv1...	978	Application Data
1692	72.430484	192.168.1.155	52.182.143.210	TCP	54	65148 → 443 [ACK] Seq=1280 Ack=6743 Win=261376 Len=0
1700	72.997338	192.168.1.155	192.168.1.255	UDP	305	54915 → 54915 Len=263
1704	73.037223	192.168.1.155	34.107.221.82	HTTP	386	GET /success.txt?ip=4 HTTP/1.1

> Frame 1689: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{19077ECA-470F-4993-A06D-4F79756B4D61}, id 0

> Ethernet II, Src: ASUSTekC_51:95:07 (a8:5e:45:51:95:07), Dst: zte_a0:55:38 (98:00:6a:a0:55:38)

Destination: zte_a0:55:38 (98:00:6a:a0:55:38)

Address: zte_a0:55:38 (98:00:6a:a0:55:38)

.... 00 = LG bit: Globally unique address (factory default)

.... 00 = IG bit: Individual address (unicast)

Source: ASUSTekC_51:95:07 (a8:5e:45:51:95:07)

Address: ASUSTekC_51:95:07 (a8:5e:45:51:95:07)

.... 00 = LG bit: Globally unique address (factory default)

.... 00 = IG bit: Individual address (unicast)

Type: IPv4 (0x0800)

> Internet Protocol Version 4, Src: 192.168.1.155, Dst: 52.182.143.210

> Transmission Control Protocol, Src Port: 65148, Dst Port: 443, Seq: 356, Ack: 6261, Len: 0

Source Port: 65148

Destination Port: 443

[Stream index: 53]

[Conversation completeness: Complete, WITH_DATA (47)]

[TCP Segment Len: 0]

La dirección MAC o dirección física es ASUSTekC_51:95:07 (a8:5e:45:51:95:07), y la dirección de destino zte_a0:55:38 (98:00:6a:a0:55:38)

Ejercicio 9 [0,5p]: Selecciona un paquete HTTP de tu captura Wireshark y busca los detalles de la cabecera TCP. Incluye una captura de pantalla con los detalles del paquete seleccionado. Ayudándote con los documentos de la capa de transporte que hemos visto en el reto 3, identifica los elementos de la cabecera TCP y descríbelos brevemente.

Ethernet

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

ip.src==192.168.1.155

No.	Time	Source	Destination	Protocol	Length	Info
1704	73.037223	192.168.1.155	34.107.221.82	HTTP	386	GET /success.txt?ip=192.168.1.155 HTTP/1.1
1713	73.115061	192.168.1.155	34.107.221.82	TCP	54	60171 → 80 [ACK] Seq=665 Ack=441 Win=262656 Len=0
1715	73.221815	192.168.1.155	107.20.137.119	TCP	54	[TCP Dup ACK 52#4] [TCP ACKed unseen segment] 58226 → 443 [ACK] Seq=1 Ack=2 Win=1025 Len=0
1716	73.996780	192.168.1.155	192.168.1.255	UDP	305	54915 → 54915 Len=263
1730	74.997566	192.168.1.155	192.168.1.255	UDP	305	54915 → 54915 Len=263
1731	75.998713	192.168.1.155	192.168.1.255	UDP	305	54915 → 54915 Len=263
1732	76.014051	192.168.1.155	13.32.91.4	TLSv1	100	Application Data
1735	76.075752	192.168.1.155	13.32.91.4	TCP	54	60167 → 443 [ACK] Seq=896 Ack=9949 Win=263424 Len=0
1745	76.995928	192.168.1.155	52.182.143.210	TLSv1	1161	Application Data, Application Data
1746	76.997231	192.168.1.155	192.168.1.255	UDP	305	54915 → 54915 Len=263
1748	77.027086	192.168.1.155	34.117.237.239	TLSv1	93	Application Data
1749	77.027087	192.168.1.155	34.120.208.123	TLSv1	93	Application Data
1753	77.033371	192.168.1.155	34.120.208.123	TLSv1	100	Application Data

Type: IPv4 (0x0800)

> Internet Protocol Version 4, Src: 192.168.1.155, Dst: 52.182.143.210

▼ Transmission Control Protocol, Src Port: 65148, Dst Port: 443, Seq: 198, Ack: 6210, Len: 158

Source Port: 65148
Destination Port: 443
[Stream index: 53]
[Conversation completeness: Complete, WITH_DATA (47)]
[TCP Segment Len: 158]
Sequence Number: 198 (relative sequence number)
Sequence Number (raw): 3516946158
[Next Sequence Number: 356 (relative sequence number)]
Acknowledgment Number: 6210 (relative ack number)
Acknowledgment number (raw): 2831003614
0101 = Header Length: 20 bytes (5)
▼ Flags: 0x018 (PSH, 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
.... .1... = Push: Set
.... .0.. = Reset: Not set
.... .0.. = Syn: Not set
.... .0.. = Fin: Not set
[TCP Flags:AP...]
Window: 1024
[Calculated window size: 262144]
[Window size scaling factor: 256]
Checksum: 0x8794 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
▼ [Timestamps]
[Time since first frame in this TCP stream: 0.304679000 seconds]
[Time since previous frame in this TCP stream: 0.009535000 seconds]
▼ [SEQ/ACK analysis]

0020 8f d2 fe 7c 01 bb d1 a0 56 ee a8 bd af de 50 18 ..|....V.....P2
0030 04 00 87 84 00 00 16 03 03 00 66 10 00 00 62 61|...f...ba

Transmission Control Protocol (tcp), 20 byte(s)

Elementos cabecera TCP

- 1) Puerto de origen 65148 puerto de envío
- 2) Puerto de destino 443 puerto de recepción
- 3) Número de secuencia (32 bits): 198 identifica el primer byte del campo de datos
- 4) Número de acuse de recibo (32 bits): 6210 indica el numero de secuencia del sgmento que se espera recibir a continuación
- 5) Longitud de la cabecera 20 indica la longitud en múltiplos de 4 bytes
- 7) Punto de urgencia (16 bits): 0 si la bandera URG está activada, este campo de 16 bits es un desplazamiento del número de secuencia que indica el último byte de datos urgentes.

- 8) Opciones y carácter de relleno
- 9) Datos
- 10)

Ejercicio 10 [0,25p]: Identifica el puerto destino que usa el protocolo de la capa de transporte en el primer paquete GET o POST de la conexión HTTP que has iniciado. ¿Hay alguna razón por la que se use ese puerto? ¿Cambiará ese puerto si en vez de conectarnos contra un servidor HTTP lo hiciéramos contra un servidor HTTPS?

Wireshark interface showing a packet capture on Ethernet. The selected packet is a GET request from 192.168.1.155 to 2600:1901:0:38d7::2a0c:5a84:e10b:5800 on port 80.

No.	Time	Source	Destination	Protocol	Length	Info
224	17.874685	2a0c:5a84:e10b:5800	2600:1901:0:38d7::2a0c:5a84:e10b:5800	HTTP	404	GET /canonical.html HTTP/1.1
292	17.920828	2600:1901:0:38d7::2a0c:5a84:e10b:5800	2a0c:5a84:e10b:5800	HTTP	376	HTTP/1.1 200 OK (text/html)

Packet details for the selected packet (No. 224):

- Ethernet II, Src: ASUSTekC_51:95:07 (a8:5e:45:51:95:07), Dst: zte_a0:55:38 (98:00:6a:a0:55:38)
- Internet Protocol Version 6, Src: 2a0c:5a84:e10b:5800:dfc:2b8a:a3a3:bd7a, Dst: 2600:1901:0:38d7::
- Transmission Control Protocol, Src Port: 60159, Dst Port: 80, Seq: 1, Ack: 1, Len: 330
 - Source Port: 60159
 - Destination Port: 80
 - [Stream index: 14]
 - [Conversation completeness: Incomplete, DATA (15)]
 - [TCP Segment Len: 330]
 - Sequence Number: 1 (relative sequence number)
 - Sequence Number (raw): 908871129
 - [Next Sequence Number: 331 (relative sequence number)]
 - Acknowledgment Number: 1 (relative ack number)
 - Acknowledgment number (raw): 980308537
 - 0101 = Header Length: 20 bytes (5)
 - Flags: 0x018 (PSH, ACK)
 - Window: 1029
 - [Calculated window size: 263424]
 - [Window size scaling factor: 256]
 - Checksum: 0xd17d [unverified]
 - [Checksum Status: Unverified]
 - Urgent Pointer: 0
 - Timestamps
 - [Time since first frame in this TCP stream: 0.048622000 seconds]
 - [Time since previous frame in this TCP stream: 0.000822000 seconds]
 - SEQ/ACK analysis
 - [RTT: 0.047800000 seconds]
 - [Bytes in flight: 330]
 - [Bytes sent since last PSH flag: 330]
 - TCP payload (330 bytes)
- Hypertext Transfer Protocol
 - GET /canonical.html HTTP/1.1\r\n
 - [Expert Info (Chat/Sequence): GET /canonical.html HTTP/1.1\r\n]
 - [GET /canonical.html HTTP/1.1\r\n]
 - [Severity level: Chat]
 - [Group: Sequence]

El puerto de destino es el puerto 80 , el cual es el puerto utilizado por defecto para el protocolo HTTP, en cambio el protocolo HTTPS que es un protocolo que garantiza transacciones seguras mediante el uso de certificados digitales, utiliza por defecto el puerto 443.

Ejercicio 11 [0,5p]: En el menú *Statistics-Protocol Hierarchy* de Wireshark podrás obtener una lista porcentual de los diferentes paquetes capturados. En particular podrás analizar qué porcentaje de paquetes pertenece a cada protocolo. Vamos a fijarnos en los protocolos de transporte. Contesta a las siguientes preguntas: 1) De los paquetes que has capturado, ¿qué porcentaje de paquetes usa TCP como transporte, y qué porcentaje usa UDP? Incluye una captura de pantalla que muestre esta estadística. 2) Teniendo en cuenta esta estadística, ¿qué podemos deducir respecto al protocolo de transporte dominante cuando navegamos por Internet?

En la captura podemos ver que el porcentaje de paquetes es claramente mayor para el protocolo TCP con un 85.1 versus un 5.1 para el protocolo UDP. Como sabemos el protocolo TCP es el predominante pues nos proporciona ordenamiento de datos seguridad y fiabilidad; mientras el protocolo UDP resulta eficiente cuando se quiere transferir una gran cantidad de dato sen poco tiempo como el caso de streaming.

Wireshark · Estadísticas de jerarquía de protocolo · Ethernet

Protocolo	Porcentaje de paquetes	Paquetes	Porcentaje de bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	689	100.0	601693	226 k	0	0	0
▼ Ethernet	100.0	689	1.6	9646	3623	0	0	0
▼ Internet Protocol Version 6	9.3	64	0.4	2560	961	0	0	0
▼ User Datagram Protocol	0.4	3	0.0	24	9	0	0	0
Multicast Domain Name System	0.4	3	0.0	135	50	3	135	50
▼ Transmission Control Protocol	8.9	61	0.7	4311	1619	42	1067	400
Transport Layer Security	1.7	12	0.7	4172	1567	12	4172	1567
▼ Internet Protocol Version 4	90.1	621	2.1	12420	4665	0	0	0
▼ User Datagram Protocol	5.1	35	0.0	280	105	0	0	0
Simple Service Discovery Protocol	1.5	10	0.2	1250	469	10	1250	469
Multicast Domain Name System	0.3	2	0.0	106	39	2	106	39
Data	3.3	23	1.0	5830	2190	23	5830	2190
▼ Transmission Control Protocol	85.1	586	93.9	564713	212 k	544	514011	193 k
Transport Layer Security	0.9	6	0.4	2574	966	6	2574	966
Hypertext Transfer Protocol	5.1	35	8.1	48893	18 k	34	47470	17 k
Data	1.6	11	0.2	1351	507	11	1523	572
Address Resolution Protocol	0.3	2	0.0	74	27	2	74	27

Vamos a analizar ahora algunos aspectos básicos del protocolo TCP. Cuando dos nodos se comunican mediante TCP, se establece una conexión antes de que estos puedan intercambiar los datos. Este procedimiento toma el nombre de *three-way handshake* y se puede consultar en el libro *Computer Networking: A Top Down approach*.

Inicia una nueva captura Wireshark y realiza una búsqueda a través de tu navegador web. Filtra los paquetes del protocolo TCP y, para mayor claridad, filtralos con `tcp.port==80`. Se visualizarán solo los paquetes del protocolo TCP que pasan por el puerto 80, relativos al protocolo HTTP.

Ejercicio 12 [0,75p]: 1) Describe el procedimiento del *three-way handshake* teniendo en cuenta el material de lectura propuesto y las trazas capturadas con Wireshark. 2) ¿Cuál es el número de secuencia del segmento TCP SYN? (Identifica el *sequence number raw*, y no el relativo.) 3) ¿Cuál es el número de secuencia (*raw*) del SYN ACK y el número de acknowledgement enviado por el servidor? ¿Cómo se han determinado estos valores? 4) ¿Cuál es el número de acknowledgement del siguiente ACK? ¿Cómo se ha determinado este valor? Acompaña tus respuestas de una captura de pantalla donde se pueda observar lo descrito.

El cliente envía un pequeño segmento TCP al servidor, cuyo segmento SYN tiene el valor de 1 añadido de un initial sequence number (*client_isn*) escogido de forma aleatoria como número de secuencia, y son enviados al servidor.

El servidor reconoce y responde con un pequeño segmento de TCP, el bit SYN se establece en 1, el campo de reconocimiento del encabezado (ACK#) del segmento TCP se establece en *client_isn + 1* y añade su propio initial sequence number (*server_isn*)

Finalmente el cliente recibe estos datos conocidos en su conjunto como SYNACK segment. Y envía un último segmento al servidor con el SYN con valor de 0 el número de reconocimiento de encabezado con el valor *server_isn+1*. La conexión está establecida

2) TCP SYN

Obtenemos el número de forma aleatoria 2852913009

Ethernet

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port==80

No.	Time	Source	Destination	Protocol	Length	Info
122	8.788694	192.168.1.155	217.138.218.98	TCP	240	51262 → 80 [PSH, ACK] Seq=1 Ack=187 Win=1026 Len=186
123	8.812360	217.138.218.98	192.168.1.155	TCP	60	80 → 51262 [ACK] Seq=1 Ack=187 Win=155 Len=0
124	8.962049	217.138.218.98	192.168.1.155	TCP	248	80 → 51262 [PSH, ACK] Seq=1 Ack=187 Win=155 Len=194
126	9.003806	192.168.1.155	217.138.218.98	TCP	54	51262 → 80 [ACK] Seq=187 Ack=195 Win=1025 Len=0
132	9.569894	192.168.1.155	217.138.218.98	TCP	66	63622 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
139	9.599546	217.138.218.98	192.168.1.155	TCP	66	80 → 63622 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1452 SACK_PERM=1
142	9.599653	192.168.1.155	217.138.218.98	TCP	54	63622 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
144	9.601353	192.168.1.155	217.138.218.98	TCP	571	63622 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656 Len=517
147	9.630556	217.138.218.98	192.168.1.155	TCP	60	80 → 63622 [ACK] Seq=1 Ack=518 Win=30720 Len=0
150	9.630556	217.138.218.98	192.168.1.155	TCP	1506	80 → 63622 [ACK] Seq=1 Ack=518 Win=30720 Len=1452
151	9.630556	217.138.218.98	192.168.1.155	TCP	1506	80 → 63622 [ACK] Seq=1453 Ack=518 Win=30720 Len=1452
152	9.630556	217.138.218.98	192.168.1.155	TCP	1506	80 → 63622 [ACK] Seq=2905 Ack=518 Win=30720 Len=1452

> Frame 132: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{19077ECA-470F-4993-A06D-4F79756B4D61}, id 0

> Ethernet II, Src: ASUSTekC_51:95:07 (a8:5e:45:51:95:07), Dst: zte_a0:55:38 (98:00:6a:a0:55:38)

> Internet Protocol Version 4, Src: 192.168.1.155, Dst: 217.138.218.98

▼ Transmission Control Protocol, Src Port: 63622, Dst Port: 80, Seq: 0, Len: 0

Source Port: 63622

Destination Port: 80

[Stream index: 10]

[Conversation completeness: Complete, WITH_DATA (63)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 2852913009

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 0

Acknowledgment number (raw): 0

1000 = Header Length: 32 bytes (8)

▼ Flags: 0x002 (SYN)

000. = Reserved: Not set

...0 = Nonce: Not set

...0... = Congestion Window Reduced (CWR): Not set

...0... = ECN-Echo: Not set

...0... = Urgent: Not set

...0... = Acknowledgment: Not set

...0... = Push: Not set

...0... = Reset: Not set

>1. = Syn: Set

...0... = Fin: Not set

[TCP Flags:S.]

Window: 64240

[calculated window size: 64240]

Checksum: 0x7657 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted

> [Timestamps]

3)SYN+ACK

Como hemos descrito anteriormente el valor de reconocimiento (raw) se genera de forma aleatoria y es 1677759825, en caso del ACK es igual al client_isn+1 = 2852913010

*Ethernet

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port==805

No.	Time	Source	Destination	Protocol	Length	Info
122	8.788694	192.168.1.155	217.138.218.98	TCP	240	51262 → 80 [PSH, ACK] Seq=1 Ack=1 Win=1026 Len=186
123	8.812360	217.138.218.98	192.168.1.155	TCP	60	80 → 51262 [ACK] Seq=1 Ack=187 Win=155 Len=0
124	8.962049	217.138.218.98	192.168.1.155	TCP	248	80 → 51262 [PSH, ACK] Seq=1 Ack=187 Win=155 Len=194
126	9.003806	192.168.1.155	217.138.218.98	TCP	54	51262 → 80 [ACK] Seq=187 Ack=195 Win=1025 Len=0
132	9.569894	192.168.1.155	217.138.218.98	TCP	66	63622 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
139	9.599546	217.138.218.98	192.168.1.155	TCP	66	80 → 63622 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1452 SACK_PERM=1
142	9.599653	192.168.1.155	217.138.218.98	TCP	54	63622 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
144	9.601353	192.168.1.155	217.138.218.98	TCP	571	63622 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656 Len=517
147	9.630556	217.138.218.98	192.168.1.155	TCP	60	80 → 63622 [ACK] Seq=1 Ack=518 Win=30720 Len=0
150	9.630556	217.138.218.98	192.168.1.155	TCP	1506	80 → 63622 [ACK] Seq=1 Ack=518 Win=30720 Len=1452
151	9.630556	217.138.218.98	192.168.1.155	TCP	1506	80 → 63622 [ACK] Seq=1453 Ack=518 Win=30720 Len=1452
152	9.630556	217.138.218.98	192.168.1.155	TCP	1506	80 → 63622 [ACK] Seq=2905 Ack=518 Win=30720 Len=1452

> Frame 139: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{19077ECA-470F-4993-A06D-4F79756B4D61}, id 0

> Ethernet II, Src: zte_a0:55:38 (98:00:6a:a0:55:38), Dst: ASUSTekC_51:95:07 (a8:5e:45:51:95:07)

> Internet Protocol Version 4, Src: 217.138.218.98, Dst: 192.168.1.155

▼ Transmission Control Protocol, Src Port: 80, Dst Port: 63622, Seq: 0, Ack: 1, Len: 0

Source Port: 80

Destination Port: 63622

[Stream index: 10]

[Conversation completeness: Complete, WITH_DATA (63)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 1677759825

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 2852913010

1000 = Header Length: 32 bytes (8)

▼ Flags: 0x012 (SYN, 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

> 1... = Syn: Set

.... 0... = Fin: Not set

[TCP Flags:A..S.]

Window: 29200

[Calculated window size: 29200]

Checksum: 0xeb1e [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale

> [Timestamps]

> [SEQ/ACK analysis]

0020 01 9b 00 50 f8 86 64 00 95 51 aa 0b ff 72 80 12 ...P...d...0.....

4) Siguiente ACK

Calculado de sequence number del servidor +1 = 1677759826

Wireshark packet capture analysis of a TCP connection. The top pane shows a list of packets, with packet 142 selected. The middle pane shows the packet details for the selected packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The bottom pane shows the raw packet data in hexadecimal and ASCII.

2.2. Comportamiento del protocolo DNS (Domain Name System)

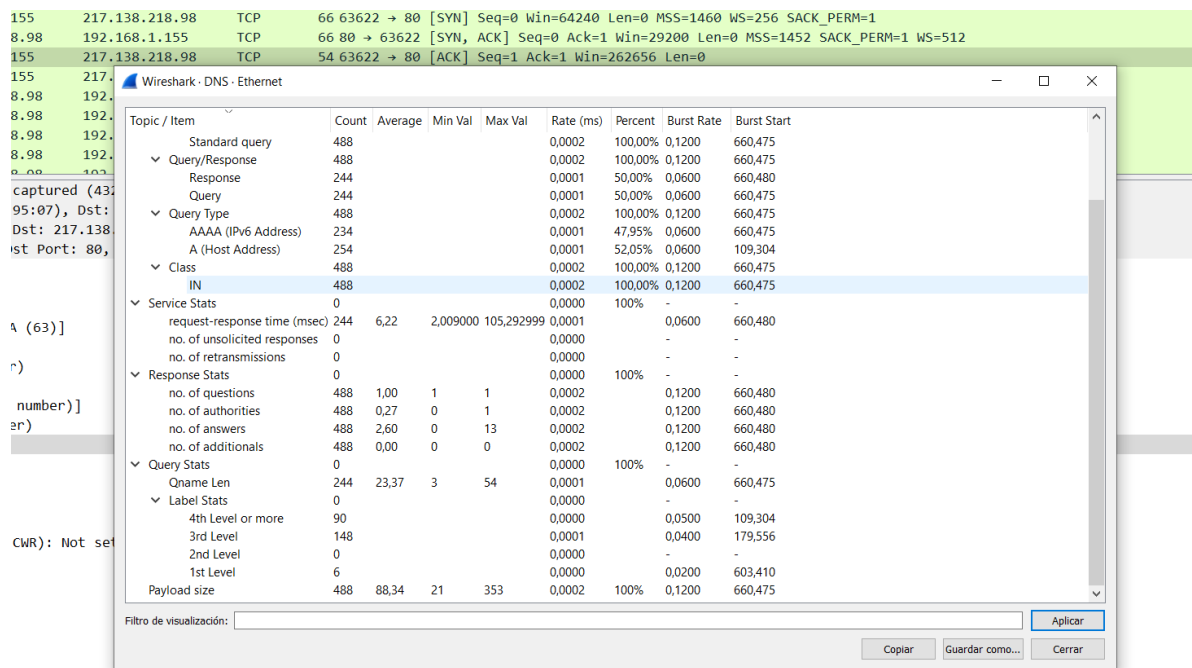
Volvamos a Wireshark y preparémoslo para una nueva captura. Antes de proceder, es recomendable vaciar la caché del DNS para asegurarnos de que la petición se realice. Puedes usar los siguientes comandos: en Windows `ipconfig /flushdns`; en MacOS `sudo killall -HUP mDNSResponder` y en Linux `sudo /etc/init.d/nscd restart`.

Inicia la captura y accede a distintas páginas web como pueden ser <https://webrtchacks.com/>, <https://www.ietf.org/>, <http://phdcomics.com/> u otras. Una vez tengas varias páginas abiertas, detén la captura de Wireshark.

Ejercicio 13 [0,25p]: Describe el principal objetivo del protocolo DNS. Con la ayuda del menú Statistics de Wireshark, explica cuántos paquetes DNS has capturado en total y su tamaño mínimo, medio y máximo. Incluye una captura de pantalla de la información estadística relacionada.

El protocolo DNS cumple de función de directorio telefónico de permitiendo acceder a la web a partir de nombres de dominio que son más fácil de recordar ,el DNS traduce los nombres de dominio a direcciones IP para que los navegadores puedan cargar los recursos de Internet.

Se han capturado 488 paquetes con un tamaño medio de 88.34 mínimo de 21 y máximo de 353



Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Standard query	488				0,0002	100,00%	0,1200	660,475
Query/Response	488				0,0002	100,00%	0,1200	660,475
Response	244				0,0001	50,00%	0,0600	660,480
Query	244				0,0001	50,00%	0,0600	660,475
Query Type	488				0,0002	100,00%	0,1200	660,475
AAAA (IPv6 Address)	234				0,0001	47,95%	0,0600	660,475
A (Host Address)	254				0,0001	52,05%	0,0600	109,304
Class	488				0,0002	100,00%	0,1200	660,475
IN	488				0,0002	100,00%	0,1200	660,475
Service Stats	0				0,0000	100%	-	-
request-response time (msec)	244	6,22	2,009000	105,292999	0,0001	-	0,0600	660,480
no. of unsolicited responses	0				0,0000	-	-	-
no. of retransmissions	0				0,0000	-	-	-
Response Stats	0				0,0000	100%	-	-
no. of questions	488	1,00	1	1	0,0002	-	0,1200	660,480
no. of authorities	488	0,27	0	1	0,0002	-	0,1200	660,480
no. of answers	488	2,60	0	13	0,0002	-	0,1200	660,480
no. of additional	488	0,00	0	0	0,0002	-	0,1200	660,480
Query Stats	0				0,0000	100%	-	-
Qname Len	244	23,37	3	54	0,0001	-	0,0600	660,475
Label Stats	0				0,0000	-	-	-
4th Level or more	90				0,0000	-	0,0500	109,304
3rd Level	148				0,0001	-	0,0400	179,556
2nd Level	0				0,0000	-	-	-
1st Level	6				0,0000	-	0,0200	603,410
Payload size	488	88,34	21	353	0,0002	100%	0,1200	660,475

Filtra ahora la captura por el protocolo DNS.

Ejercicio 14 [0,5p]: Identifica las peticiones DNS y sus mensajes de respuesta. 1) ¿Se envían a través de TCP o de UDP? 2) ¿Cuál es el puerto de destino de las peticiones DNS? ¿Cuál es el puerto de origen de los mensajes DNS de respuesta? Adjunta una captura de pantalla donde se pueda apreciar lo observado.

Usamos el filtro “DNS” para identificar las peticiones sDNS vemos que estas se envían a través de UDP siendo para las peticiones DNS el puerto de origen el 53195 y el puerto de destino el 53

The screenshot shows a Wireshark capture of network traffic. The packet list pane displays several DNS packets. Packet 45 is selected, and its details are shown in the packet details pane. The details pane shows the following structure:

- Ethernet II, Src: ASUSTekc_51:95:07 (a8:5e:45:51:95:07), Dst: zte_a0:55:38 (98:00:6a:a0:55:38)
- Internet Protocol Version 6, Src: 2a0c:5a84:e10b:5800:dfc:2b8a:a3a3:bd7a, Dst: 2a0c:5a80:0:2::1
- User Datagram Protocol, Src Port: 53195, Dst Port: 53
- Domain Name System (query)
 - Transaction ID: 0xc341
 - Flags: 0x0100 Standard query
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries
 - [Response In: 47]

Mientras que para las respuestas el puerto de origen es 53 y el de destino es 56736 en este caso

Wireshark interface showing a packet capture of DNS traffic. The packet list shows a query for 'api.windscribe.com' from source 2a0c:5a84:e10b:58... to destination 2a0c:5a80:0:2::1. The packet details pane shows the following information:

- Frame 46: 161 bytes on wire (1288 bits), 161 bytes captured (1288 bits) on interface \Device\NPF_{19077ECA-470F-4993-A06D-4F79756B4D61}, id 0
- Ethernet II, Src: zte_a0:55:38 (98:00:6a:a0:55:38), Dst: ASUSTekC_51:95:07 (a8:5e:45:51:95:07)
- Internet Protocol Version 6, Src: 2a0c:5a80:0:2::1, Dst: 2a0c:5a84:e10b:5800:dfc:2b8a:a3a3:bd7a
 - 0110 = Version: 6
 - 0000 0000 = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - 1010 0100 1101 0111 = Flow Label: 0xa4db7
 - Payload Length: 107
 - Next Header: UDP (17)
 - Hop Limit: 60
 - Source Address: 2a0c:5a80:0:2::1
 - Destination Address: 2a0c:5a84:e10b:5800:dfc:2b8a:a3a3:bd7a
- User Datagram Protocol, Src Port: 53, Dst Port: 56736
 - Source Port: 53
 - Destination Port: 56736
 - Length: 107
 - Checksum: 0xf6ea [unverified]
 - [Checksum Status: Unverified]
 - [Stream index: 4]
 - [Timestamps]
 - UDP payload (99 bytes)
- Domain Name System (response)
 - Transaction ID: 0x9598
 - Flags: 0x8180 Standard query response, No error
 - Questions: 1
 - Answer RRs: 2
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries
 - Answers
 - [Request In: 441]
 - [Time: 0.003179000 seconds]

Ejercicio 15 [0,25p]: ¿Cuál es la dirección IP de tu servidor DNS? Justifica la respuesta acompañándola de una captura de pantalla de Wireshark.

*Ethernet

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

dns

No.	Time	Source	Destination	Protocol	Length	Info
20	6.966116	2a0c:5a84:e10b:58...	2a0c:5a80:0:2::1	DNS	98	Standard query 0x1c95 A api.windscribe.com
21	6.968961	2a0c:5a80:0:2::1	2a0c:5a84:e10b:58...	DNS	146	Standard query response 0x1c95 A api.windscribe.com A 104.20.26.2
44	7.321906	2a0c:5a84:e10b:58...	2a0c:5a80:0:2::1	DNS	105	Standard query 0x9598 AAAA chrome.cloudflare-dns.com
45	7.321929	2a0c:5a84:e10b:58...	2a0c:5a80:0:2::1	DNS	105	Standard query 0xc341 A chrome.cloudflare-dns.com
46	7.325085	2a0c:5a80:0:2::1	2a0c:5a84:e10b:58...	DNS	161	Standard query response 0x9598 AAAA chrome.cloudflare-dns.com AAA
47	7.325388	2a0c:5a80:0:2::1	2a0c:5a84:e10b:58...	DNS	137	Standard query response 0xc341 A chrome.cloudflare-dns.com A 104.
128	9.440596	2a0c:5a84:e10b:58...	2a0c:5a80:0:2::1	DNS	104	Standard query 0x62a8 A es-003.whiskergalaxy.com
129	9.471620	2a0c:5a84:e10b:58...	2a0c:5a84:0:2::1	DNS	104	Standard query 0x62a8 A es-003.whiskergalaxy.com
130	9.477388	2a0c:5a80:0:2::1	2a0c:5a84:e10b:58...	DNS	120	Standard query response 0x62a8 A es-003.whiskergalaxy.com A 217.1
131	9.501621	2a0c:5a84:0:2::1	2a0c:5a84:e10b:58...	DNS	120	Standard query response 0x62a8 A es-003.whiskergalaxy.com A 217.1
133	9.577074	2a0c:5a84:e10b:58...	2a0c:5a80:0:2::1	DNS	107	Standard query 0x8686 A safebrowsing.googleapis.com
134	9.577082	2a0c:5a84:e10b:58...	2a0c:5a80:0:2::1	DNS	107	Standard query 0x418e AAAA safebrowsing.googleapis.com A 14

> Frame 128: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface \Device\NPF_{19077ECA-470F-4993-A06D-4F79756B4D61}, id 0

> Ethernet II, Src: ASUSTekC_51:95:07 (a8:5e:45:51:95:07), Dst: zte_a0:55:38 (98:00:6a:a0:55:38)

Internet Protocol Version 6, Src: 2a0c:5a84:e10b:5800:dfc:2b8a:a3a3:bd7a, Dst: 2a0c:5a80:0:2::1

0110 = Version: 6

> 0000 0000 = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)

.... 0000 0000 0000 0000 = Flow Label: 0x000000

Payload Length: 50

Next Header: UDP (17)

Hop Limit: 64

Source Address: 2a0c:5a84:e10b:5800:dfc:2b8a:a3a3:bd7a

Destination Address: 2a0c:5a80:0:2::1

User Datagram Protocol, Src Port: 51794, Dst Port: 53

Source Port: 51794

Destination Port: 53

Length: 50

Checksum: 0xdd13 [unverified]

[Checksum Status: Unverified]

[Stream index: 7]

> [Timestamps]

UDP payload (42 bytes)

Domain Name System (query)

Transaction ID: 0x62a8

Flags: 0x0100 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 0

Queries

[Response In: 130]

2.3. Comportamiento del protocolo SIP (Session Initiation Protocol)

SIP (Session Initiation Protocol) es un protocolo del nivel de aplicación que permite inicializar, modificar y finalizar sesiones interactivas que impliquen elementos multimedia como vídeo, voz, mensajería instantánea, juegos en línea, realidad virtual, etc.

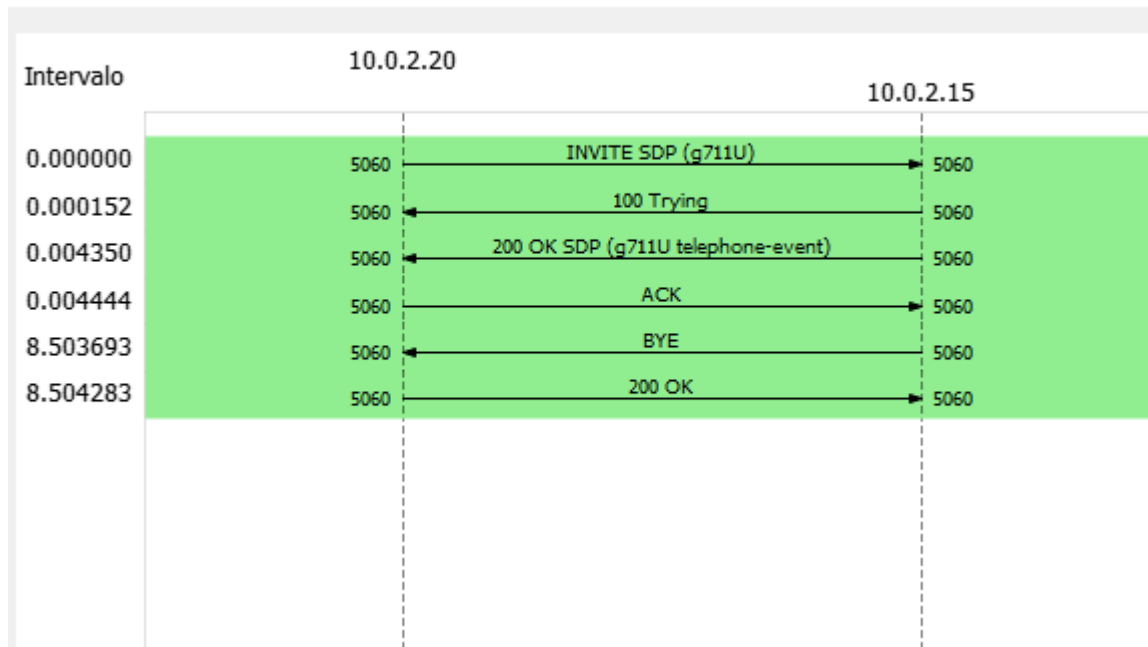
Las operadoras de telecomunicación usan el protocolo SIP para señalizar el tráfico en sus redes más modernas de telefonía (ya sean fijas o móviles) y lo acompañan de flujos RTP (Real-Time Protocol) que contienen la voz de los usuarios. Lo que sucede durante una llamada (quien ha llamado a quien, durante cuanto tiempo, si ha habido problemas, cuál ha sido la calidad del audio, etc) se reporta en los llamados CDR (Call Detailed Records) generando un alto volumen de datos. Esta información la utilizan las operadoras para tomar decisiones sobre cuestiones operativas (dimensionamientos y capacidades de red, identificar problemas, cambiar lógicas de enrutamiento) y para extraer conclusiones sobre el comportamiento de los usuarios.

Veamos ahora como funciona este protocolo utilizando la captura de una llamada de Voz sobre IP (VoIP). Abre con Wireshark el fichero 'sip-rtp-g711.pcap' que puedes encontrar en el Campus (junto con este enunciado) y aplica un filtro para que solo te muestre el protocolo SIP. En Wireshark, ves al menú 'Telefonía' y selecciona la opción 'Flujos SIP'. Selecciona el primer flujo (lo puedes identificar, ya que su estado está marcado como 'COMPLETED') y clicas en la opción 'Flow Sequence'.

Ejercicio 16 [0,25p]: ¿Cuántos mensajes SIP puedes ver en el diagrama mostrado? Indica cuáles son.

- 1) INVITE SDP
- 2) Estatus 100 Trying
- 3) 200 OK SDP
- 4) ACK
- 5) BYE
- 6) 200 Ok

Wireshark · Call Flow · sip-rtp-g711.pcap



Ejercicio 17 [0,25p]: Identifica cuáles son las direcciones IP de los dos teléfonos que participan en esta llamada.

Proviene de 10.0.2.20 a 10.0.2.15

Ejercicio 18 [0,25p]: ¿Cuánto tiempo dura el intercambio de paquetes de audio en esta llamada? Indica como lo has calculado.

Restando el intervalo desde SIP invite to SIP bye obtenemos 8,459253, como la duración de la llamada

Ejercicio 19 [0,5p]: Imaginemos ahora que, usando una aplicación basada en SIP, queremos iniciar un stream de video desde nuestra Webcam para conectarnos a la plataforma lanzada recientemente por Facebook, Meta. La Webcam genera una resolución de 640x480 píxeles por frame, donde cada píxel se representa con 3 colores, y cada color con 8 bits. El video resultante tiene una frame rate de 20 frames por segundo.

1. ¿Cuál es el ancho de banda necesario para transmitir el video sin compresión? ¿Cuál es si aplicamos un algoritmo de compresión con un ratio 40:1

$640 \times 480 \times 3 \times 8 \text{ bit (no comprimido)} = 7.372.800 \text{ bit /frame} \times 20 \text{ frames/sec} =$
 $147456000 \text{ bit/sec} = 144 \text{ Mbps}$

A la compresión 40:1

$144/40 \text{ Mbps} = 3.6 \text{ Mbps}$

2. ¿Qué ratio de compresión sería necesario si quisiéramos transmitir el video sobre un enlace de 512 kbps?

$144 \text{ Mbps} / 512 \text{ kbps (0.5 Mbps)} = 288 : 1$

3. Extracción de datos desde una API

API es el acrónimo en inglés de *application programming interface*. La API es básicamente una interfaz que habla por nosotros con un programa. Cuando un programador decide poner a disposición pública algún conjunto de datos generados por sus programas, lo puede hacer a través de una interfaz de este tipo. Otros programadores extraen estos datos de la aplicación construyendo URLs o a través de clientes HTTP. Una API está caracterizada por tres elementos:

- el **usuario**, que es la persona realizando la petición;
- el **cliente**, que es el ordenador que envía la petición al servidor;
- el **servidor**, que es el ordenador que responde a la petición.

Los datos se almacenan en un servidor y la documentación correspondiente se pone a disposición de los programadores para que puedan acceder. Un usuario externo entonces puede realizar peticiones de datos a este servidor. Ejemplos de estos sistemas son las redes sociales comunes como Twitter, Facebook, YouTube, LinkedIn, etc. Incluso portales como la NASA o el New York Times ponen sus datos a disposición para que los desarrolladores puedan acceder a ellos y manipularlos.

A través de una API conocida como RESTful API, podemos realizar peticiones (*queries*) utilizando el protocolo HTTP. ¿Cómo? Lo haremos realizando *HTTP-requests* y *HTTP-responses*. Concretamente, las APIs pueden realizar un número limitado de acciones entre las que encontramos:

- **GET**: solicita datos a un servidor;
- **POST**: envía cambios desde el cliente al servidor;
- **PUT**: revisa o añade a la información existente;
- **DELETE**: destruye información existente.

¿A qué tipos de datos podemos acceder? Por ejemplo, con Facebook podríamos obtener la lista de amigos entre un amigo nuestro y nosotros mismos, o el número de likes obtenidos en ciertas páginas. A Twitter le podríamos pedir la lista de temas y argumentos más tratados en una cierta área geográfica, por ejemplo en España. A través de la Google Direction API podemos obtener rutas para llegar de una dirección a otra. Por último, desde la Google Book API, podríamos obtener la lista de libros de un autor, etc.

La mayoría de aplicaciones mencionadas requieren gestionar una autorización que se tiene que pedir, concretándose en una contraseña o clave de la API. Las API públicas también tienen limitaciones en cuanto a cantidad de datos disponibles y número de peticiones por segundo posibles. Los datos que obtenemos se representan a través de un formato específico conocido como JSON (Java Script Object Notation). JSON es un formato estándar que se usa para transmitir datos entre un servidor y una aplicación web como alternativa al XML.

Vamos a utilizar *Fruityvice*, una API abierta y muy sencilla que nos permite obtener datos sobre distintos tipos de frutas. Iniciaremos seguidamente una nueva captura en Wireshark. Antes de empezar, es recomendable vaciar la caché del DNS (ver Sección 2.2).

Realicemos ahora una petición a la API que nos devuelva información sobre todas las frutas existentes en la base de datos: introduc. Una vez el navegador devuelva los datos solicitados podemos parar la captura Wireshark.

Vamos a ver primero como el protocolo DNS resuelve la dirección IP del servidor de la API FruityVice. Filtramos los paquetes capturados por DNS y buscamos la petición al servidor DNS local de la dirección IP del servidor www.fruityvice.com.

Ejercicio 20 [0,25p]: ¿Cuál es la dirección IP que resuelve el dominio www.fruityvice.com? Adjunta una captura de pantalla indicando el paquete donde se encuentra esta respuesta.

La dirección IP es 217.160.142.194

2194 42.838377	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	114 Standard query response 0x97a0 A www.fruityvice.com A 217.160.142.194
2201 42.839967	2a0c:5a80:0:2::1	DNS	98 Standard query 0x63d5 A www.fruityvice.com
2202 42.840590	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	114 Standard query response 0x97a0 A www.fruityvice.com A 217.160.142.194
2207 42.846834	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	114 Standard query response 0x63d5 A www.fruityvice.com A 217.160.142.194
2208 42.847948	2a0c:5a80:0:2::1	DNS	98 Standard query 0x446d AAAA www.fruityvice.com
2209 42.850890	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	171 Standard query response 0x446d AAAA www.fruityvice.com SOA ns1063.ui-d
2485 43.723178	2a0c:5a80:0:2::1	DNS	109 Standard query 0xca67 A v10.events.data.microsoft.com
2486 43.723323	2a0c:5a80:0:2::1	DNS	109 Standard query 0xcc9f AAAA v10.events.data.microsoft.com
2493 43.726391	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	243 Standard query response 0xca67 A v10.events.data.microsoft.com CNAME g
2494 43.726723	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	304 Standard query response 0xc9f AAAA v10.events.data.microsoft.com CNAME g
3152 48.764975	2a0c:5a80:0:2::1	DNS	106 Standard query 0x2e2d A token.services.mozilla.com
3153 48.765048	2a0c:5a80:0:2::1	DNS	106 Standard query 0xfef3 AAAA token.services.mozilla.com
3154 48.776344	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	200 Standard query response 0x2e2d A token.services.mozilla.com CNAME token
3155 48.776344	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	237 Standard query response 0xfef3 AAAA token.services.mozilla.com CNAME token
3157 48.777945	2a0c:5a80:0:2::1	DNS	112 Standard query 0xa2cd A token.r53-2.services.mozilla.com
3158 48.782147	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	160 Standard query response 0xa2cd A token.r53-2.services.mozilla.com A 34.
3159 48.783215	2a0c:5a80:0:2::1	DNS	112 Standard query 0xf18f AAAA token.r53-2.services.mozilla.com
3160 48.786026	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	197 Standard query response 0xf18f AAAA token.r53-2.services.mozilla.com S
3183 49.516546	2a0c:5a80:0:2::1	DNS	123 Standard query 0xa389 AAAA sync-1-us-west1-g.sync.services.mozilla.com
3184 49.519626	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	210 Standard query response 0xa389 AAAA sync-1-us-west1-g.sync.services.mo
3185 49.521192	2a0c:5a80:0:2::1	DNS	123 Standard query 0x4039 AAAA sync-1-us-west1-g.sync.services.mozilla.com
3187 49.538958	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	210 Standard query response 0x4039 AAAA sync-1-us-west1-g.sync.services.mo
3269 50.699144	2a0c:5a80:0:2::1	DNS	101 Standard query 0x070a A update.googleapis.com
3270 50.699144	2a0c:5a80:0:2::1	DNS	101 Standard query 0x202c AAAA update.googleapis.com
3271 50.702020	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	129 Standard query response 0x202c AAAA update.googleapis.com AAAA 2a00:14
3272 50.702528	2a0c:5a84:e10b:5800:c8e4:8900:2caf:af69	DNS	117 Standard query response 0x070a A update.googleapis.com A 142.250.200.6


```

Transaction ID: 0x97a0
> Flags: 0x8100 Standard query response, No error
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
v Queries
  www.fruityvice.com: type A, class IN
    Name: www.fruityvice.com
    [Name Length: 18]
    [Label Count: 3]
    Type: A (Host Address) (1)
    Class: IN (0x0001)
v Answers
  www.fruityvice.com: type A, class IN, addr 217.160.142.194

```

Ejercicio 21 [0,5p]: Teniendo en cuenta la información que has derivado en el ejercicio anterior (sobre la dirección IP que resuelve el dominio de interés), estima cuánto tiempo ha durado la descarga (en milisegundos) de los datos sobre las frutas desde esa dirección IP. Adjunta una captura de pantalla de Wireshark para justificar tu respuesta.

TimeSourceDestinationProtocolLengthInfo9168192.9984382a0c:5a84:e10b:5800:c84d:8900:2cfa:f69DNS146Standard query response 0x1fe aapl.roninchain.com A 104.22.31.32 A 104.22.30.19169193.0001382a0c:5a80:012:1:1DNS98Standard query 0x9229 AAAA aapl.roninchain.com99Wireshark - Packet 9249 - Ethernet910User Datagram Protocol, Src Port: 53, Dst Port: 50260910Source Port: 53910Destination Port: 50260910Length: 117910Checksum: 0x3774 [unverified]910[Checksum Status: Unverified]910[Stream index: 133]910Timestamps910[Time since first frame: 0.002932000 seconds]910[Time since previous frame: 0.002932000 seconds]910UDP payload (109 bytes)910Domain Name System (response)910Transaction ID: 0x3c28910Flags: 0x8180 Standard query response, No error9101... .. = Response: Message is a response910...000 0... .. = Opcode: Standard query (0)910... ..0... .. = Authoritative: Server is not an authority for domain910... ..0... .. = Truncated: Message is not truncated910... ..1... .. = Recursion desired: Do query recursively910... ..1... .. = Recursion available: Server can do recursive queries910... ..0... .. = Z: reserved (0)910... ..0... .. = Answer authenticated: Answer/authority portion was not authenticated by the server910... ..0... .. = Non-authenticated data: Unacceptable910... ..0000... .. = Reply code: No error (0)910Questions: 1910Answer RRs: 0910Authority RRs: 1910Additional RRs: 0100Queries100www.fruityvice.com: type AAAA, class IN100Name: www.fruityvice.com910000a8 5e 45 51 95 07 98 006a a0 55 38 86 d0 60 07->EQ... j-UB...910001080 15 00 75 11 3c 2a 0c5a 80 00 00 00 02 00 00...->... Z...910002000 00 00 00 00 01 2a 0c5a 84 e1 0b 58 00 c8 6d...->... Z...X...910003089 00 2c af af 69 00 35c4 54 00 75 37 74 3c 28...->... i-5 T-u7K<910004081 80 00 01 00 00 00 0100 00 03 77 77 77 0a 66...->... www-f910005072 75 69 74 79 76 69 6365 03 63 6f 6d 00 01 cfruityvic ecom910006000 01 c0 10 00 06 00 0100 00 00 43 00 3d 06 6e...->... C-n910007073 31 30 36 63 06 75 692d 64 6e 73 03 6f 72 6f...->... s1063-ui -dns-org910008000 0a 68 6f 73 74 6d 6173 74 65 72 05 31 75 6e...->... -hostma ster-lun910009064 31 03 63 6f 6d 00 7839 45 07 00 00 70 80 00d1-com-x 9...p-9100000 1c 00 00 3a 80 00 00 0000 02 68

fruityvice.com: tyName: fruityviceType: SOA (StarClass: IN (0x00Time to live: 6Data length: 61Primary name seResponsible autSerial Number:Refresh Interva

Ejercicio 22 [0,25p]: Explorando la captura del ejercicio anterior, indica si, teniendo visibilidad de lo que está sucediendo en la red (es decir, lo que vemos en Wireshark), es posible leer el contenido de la información que se está transmitiendo: en este caso la lista de las frutas con sus características. Justifica tu respuesta.

Como vemos al evaluar encontramos que los datos encuentra encriptados por un capa de seguridad la capa TSL , que permite la encriptación de los datos y transferencia segura entre el cliente y el servidor

Imaginemos ahora que hemos fundado nuestra propia startup para ofrecer, a través de Internet, una API REST que expone datos sobre la evolución, cotización y transacciones de distintos tipos de criptomoneda. Esta API está diseñada para que la consuman aplicaciones de trading que, con la información que extraigan de la API, van a generar recomendaciones de inversión personalizadas para cada uno de sus clientes. Llevamos poco tiempo con este proyecto y, aunque tenemos a muchos clientes que están probando nuestro servicio de forma gratuita, son pocos los que han dado el paso al servicio premium de pago. Con esto, tenemos que vigilar muy bien cuáles son nuestros costes de operación. Por este motivo, como se indica en la Figura 3, vamos a exponer nuestra API a través de un único servidor que va a disponer de una conexión a Internet limitada a un ancho de banda

de subida de $r_S = 100 \text{ Mb/s}$ (sentido servidor-clientes). El contenido completo de la información que expondremos vía nuestra API tiene un tamaño de $S = 200 \text{ MB}^4$. Imaginemos que Elon Musk, fundador de varias compañías entre las que se encuentra Tesla, publica un Tweet anunciando que a partir del próximo año Tesla va a dejar de aceptar moneda tradicional y los pagos deberán hacerse únicamente con Bitcoin.



Figura 2: Tweet imaginario de Elon Musk

Esto, automáticamente, genera un alto volumen de transacciones de gente intentando comprar Bitcoin en un espacio de tiempo muy corto. Todos nuestros clientes necesitan actualizar sus directrices de inversión inmediatamente, por lo que acceden de forma concurrente a nuestro servidor para descargar la información actualizada. Las computadoras de nuestros clientes se encuentran distribuidas a través de $N_A = 150$ redes de acceso, de forma que cada red tiene $N_C = 10$ computadoras.

Ejercicio 23 [0,5p]: Considera que cada computadora de cliente descarga la información actualizada una única vez y que cada petición de actualización devuelve el contenido completo de la API. Calcula:

1. El volumen total de tráfico enviado desde nuestro servidor, ...

Total de ordenadores= 1500 computadoras

Ancho de banda = 100Mb/s

Tamaño 200 MB

Trafico $1500 \times 200 \text{ MB} = 300000 \text{ MB}$

1. El volumen de tráfico recibido por cada una de las redes de acceso, $T_{cs,a}$

Volumen por red $150 \times 200 \text{ MB} = 30000 \text{ MB}$

2. La velocidad de descarga para cada cliente, $r_{cs,dl}$

100 Mb/s entre 1500 = 0.067 Mb/s

3. El tiempo total de descarga para cada cliente, t_{dl}

200 MB entre 0.067 Mb/s = 6,63 horas

Dada la experiencia en el ejercicio anterior, hemos recibido quejas de algunos clientes indicando que el tiempo de descarga de actualizaciones durante el incidente 'Elon Musk' ha sido demasiado

largo y no han podido ofrecer recomendaciones de inversión en tiempo real. Para ello, hablamos con nuestro equipo de ingeniería y decidimos que, para poder dar un buen nivel de servicio durante este tipo de eventos, debemos redefinir la arquitectura de red de nuestro servicio. Después de considerar distintas alternativas tecnológicas y sus costes asociados, decidimos llegar a un acuerdo con

³1 Mb/s = 10^6 bits/s.

⁴1 MB = 10^6 bytes = $8 \cdot 10^6$ bits.

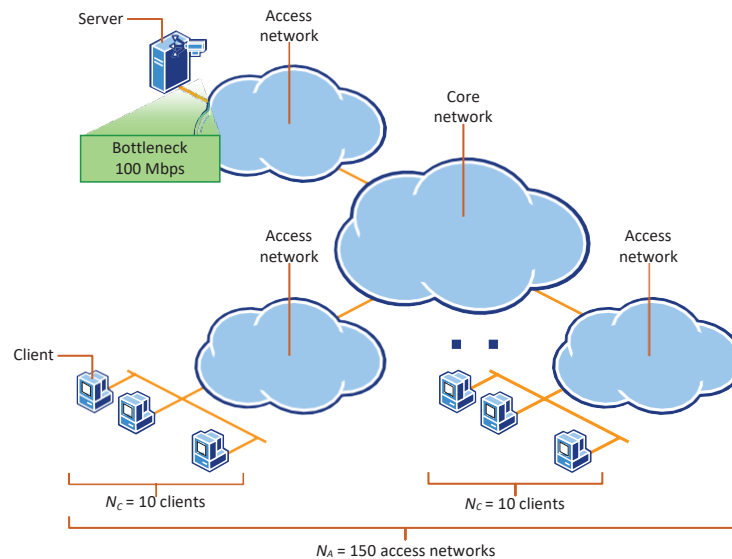


Figura 3: Arquitectura Cliente/Servidor.

una empresa de distribución de contenidos, también conocidas como CDN (Content Distribution Networks), tal y como se puede ver en la Figura 4. Esta empresa dispone de un servidor CDN en cada red de acceso, por lo que estos, en todo momento, van a mantener una copia actualizada del contenido de nuestra API. De este modo, las computadoras de nuestros clientes se van a conectar a estos servidores 'espejo' que residen en la red de acceso para descargar la información. La velocidad máxima de subida para estos servidores de la CDN es de $r_s = 100 \text{ Mb/s}$ (sentido servidorCDN-clientes).

Ejercicio 24 [0,5p]: calcula las mismas variables que en el ejercicio anterior, ahora denotadas como $T_{cdn,ul}$, $T_{cdn,av}$, $r_{cdn,db}$, $t_{cdn,dl}$.

Ejercicio 25 [0,5p]: Considera el caso de uso de los dos ejercicios anteriores. ¿Qué formato de datos seleccionarías para entregar los datos vía nuestra API REST? Justifica tu respuesta y proporciona un ejemplo de mensaje para este caso de uso.

Es muy común que las API exporten datos via formato JSON , unos de los formatos mas muy útil para el intercambio de información debido a ser muy ligero, su facilidad de lectura, de análisis y de generación. Están conformados por las claves y los valores,

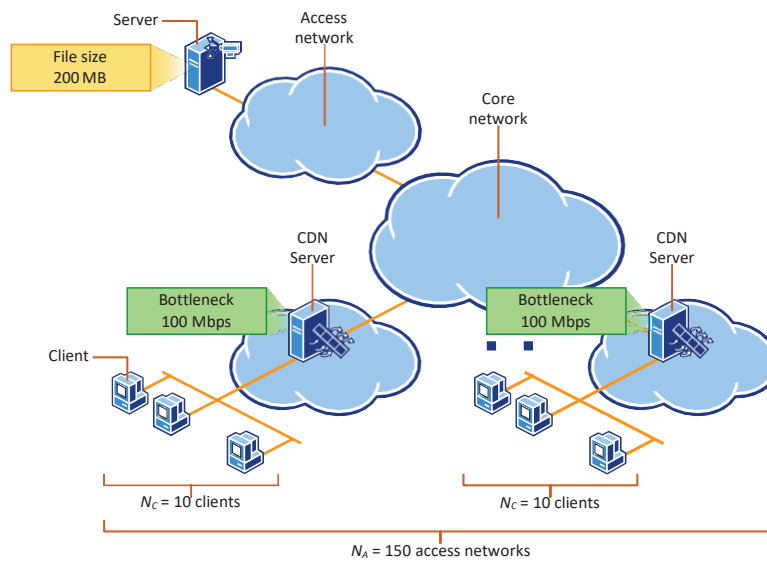


Figura 4: Arquitectura CDN.