# Presentaci´on

¿Sab´ıas 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 qu´e usuarios est´an twitteando o sobre qu´e temas se est´an twitteando.

En esta actividad se te propone utilizar una web API con el objetivo puesto en el an´alisis 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 t´ecnicos de red (Wireshark). Esta herramienta nos permitir´a detectar el tr´afico (o hacer *traffic sniffing* en ingl´es) y poder as´ı analizar los paquetes que pasan por la interfaz de red de nuestro ordenador. A su vez, os planteamos una serie de preguntas te´oricas 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 ´area de estudio que parte de la base de la educaci´on secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye tambi´en 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 vocaci´on de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboraci´on y defensa de argumentos y la resoluci´on de problemas dentro de su ´area de estudio.

Uso y aplicaci´on de las TIC en el ´ambito acad´emico y profesional.

Administrar y gestionar los sistemas operativos y de comunicaciones de los componentes de una red de ordenadores.

# Objetivos

Los objetivos espec´ıficos de esta PEC son:

Ser capaz de comprender las funciones asociadas a las capas superiores del modelo OSI (transporte y aplicaci´on) 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.

# Descripci´on de la PEC a realizar

Esta PEC har´a que pongas en pr´actica los conocimientos y procedimientos asociados a este reto. Partiremos del mismo navegador que utilizamos cada d´ıa para navegar por Internet y teclearemos una direcci´on web para analizar qu´e pasa en la red todas las veces que ejecutamos una operaci´on tan b´asica y cotidiana. Seguidamente, haremos lo mismo accediendo a una API de la que podr´ıamos decidir extraer datos. En esta PEC no nos centraremos en la extracci´on de datos, que ya se tratar´a en otros cursos/asignaturas, sino que s´olo observaremos y analizaremos lo que pasa en la red.

En cuanto a los ejercicios de esta PEC, se combinan preguntas te´oricas referentes a los principales conceptos de este reto, junto con cuestiones referentes al an´alisis de red a trav´es de una herramienta de software libre: Wireshark.

**NOTA**: El navegador Google Chrome suele utilizar un protocolo particular llamado QUIC que puede dificultarnos la visi´on de los conceptos a tratar. Por ello, se recomienda usar otros navega- dores (p. ej. Mozilla Firefox). En caso de que quer´ais usar Google Chrome, os dejamos [**este**](https://support.eset.com/es/kb6757-deshabilitar-el-protocolo-quic-en-el-navegador-google-chrome)[**enlace**](https://support.eset.com/es/kb6757-deshabilitar-el-protocolo-quic-en-el-navegador-google-chrome) que explica c´omo deshabilitar QUIC.

# Recursos Recursos B´asicos

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 valoraci´on

La PEC debe resolverse de manera individual.

Es necesario justificar todas las respuestas a los ejercicios propuestos en la PEC.

La puntuaci´on (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 c´omputo de la evaluaci´on con- tinua de la asignatura: *AC* = 0*,*2 *· PEC*1 + 0*,*3 *· PEC*2 + 0*,*3 *· PEC*3 + 0*,*2 *· PEC*4

Para m´as informaci´on sobre el modelo de evaluaci´on de la asignatura os remitimos al Plan Docente.

# Formato y fecha de entrega

La entrega de esta PEC constar´a de un archivo .pdf siguiendo el formato espec´ıfico

**PEC3*Apellido1Apellido2Nombre*.pdf** e incluir´a:

La memoria de la pr´actica en formato .pdf que contenga las respuestas correspondientes a todos los Ejercicios planteados.

Este archivo .pdf deber´a enviarse a trav´es de la herramienta REC (Registro de Evaluaci´on Con- tinua) del aula **antes de las 23:59 del d´ıa 10/12/2021**.

**IMPORTANTE**: Recordad que la PEC es individual. La detecci´on de falta de originalidad ser´a penalizada conforme a la normativa vigente de la UOC. Adem´as, al hacer la entrega ase- guraos de comprobar que el fichero entregado es el correcto, pues es responsabilidad del alumno realizar las entregas correctamente. No se aceptar´an entregas fuera de plazo.

# Primer contacto con Wireshark

## Iniciar Wireshark

Wireshark se ha convertido en el principal analizador de red utilizado por los ingenieros de re- des. Este software de c´odigo abierto est´a disponible para muchos sistemas operativos diferentes, incluidos Windows, MAC y Linux. Wireshark se puede descargar desde [www.wireshark.org.](http://www.wireshark.org/)

Utiliza el recurso *Manual de Wireshark* para instalar y obtener una primera visi´on de conjunto del aplicativo. Elige la versi´on de software que necesites segu´n 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 instalaci´on de WinPcap o de Npcap. Si WinPcap o Npcap ya est´an instalados en el PC, la casilla de verificaci´on Install (Instalar) estar´a desactivada. Si la versi´on instalada de estos programas es anterior a la versi´on que incluye Wireshark, se recomienda que permitas que la versi´on m´as reciente se instale haciendo clic en la casilla de verificaci´on Install WinPcap o Npcap.

## Selecciona una interfaz de red

Una vez finalizada la instalaci´on, 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 listar´an todas las interfaces de red con las que el Wireshark puede capturar datos. Selecciona la interfaz por la cual est´as conectado a Internet. Es muy posible que est´es conectado a trav´es de Wi-Fi. En ese caso, selecciona la tarjeta de red Wi-Fi. En caso de que est´es conectado a trav´es de Ethernet, selecciona la interfaz correspondiente.

Los nombres de las interfaces depender´an del modelo de tarjetas que lleven vuestros equipos. Una manera de ver qu´e interfaces nos est´an proveyendo de tr´afico es a trav´es de la pequen˜a gr´afica situada justo al lado del nombre de las interfaces listadas justo al abrir Wireshark. En las interfaces con tr´afico se apreciar´an curvas variantes, mientras que en las que no tienen tr´afico las gr´aficas se mantendr´an constantes en el origen. En el ejemplo de la Figura [1](#_bookmark1) se observa actividad en la interfaz de Wi-Fi *Wi-Fi 2*, que es la que nos provee de conexi´on a Internet.[1](#_bookmark0)

En el menu´ **Capture**, puedes elegir en la opci´on *Options* algunos par´ametros para especificar la captura. Por ejemplo, en la pestan˜a *Options*, puedes configurar que la captura se pare autom´ati-

1Obviad la interfaz de red virtual de loopback.

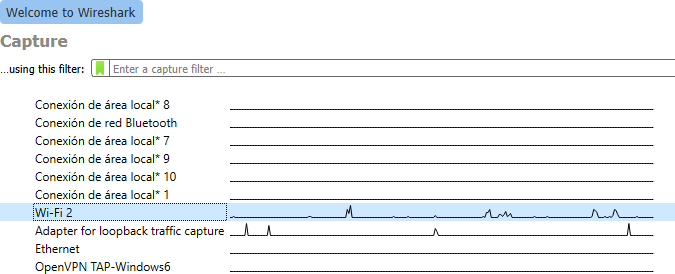


Figura 1: Ejemplo de posibles interfaces listadas en Wireshark.

camente despu´es de un cierto tiempo o despu´es de haber capturado una cierta cantidad de datos. Cuando pulses sobre *Start*, el programa empezar´a a capturar todos los paquetes que pasen a trav´es de la interfaz que hayas seleccionado. Seguramente observaremos que, aunque no estemos haciendo nada en particular, el programa ya empieza a capturar paquetes. Veamos qu´e es lo que Wireshark nos est´a 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 despu´es de iniciar la captura (es decir, de hacer clic en *Start* ).
3. *Source*: La tercera columna muestra la direcci´on IP que ha generado el paquete.
4. *Destination*: La cuarta columna muestra la direcci´on 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 u´ltima columna describe informaci´on adicional de los protocolos de comunicaci´on utilizados, como por ejemplo valores espec´ıficos de ciertas cabeceras de inter´es.

En la casilla *Apply a display filter*, ubicada en la parte superior de la pantalla principal de Wi- reshark, podemos introducir filtros de visualizaci´on que nos ayuden a leer e interpretar los datos. Algunos de los filtros de visualizaci´on m´as comunes de Wireshark y que utilizaremos en esta pr´actica se listan a continuaci´on a trav´es de ejemplos. Puedes encontrar m´as informaci´on so- bre los filtros de visualizaci´on en: [https://gitlab.com/wireshark/wireshark/-/wikis/](https://gitlab.com/wireshark/wireshark/-/wikis/DisplayFilters) [DisplayFilters](https://gitlab.com/wireshark/wireshark/-/wikis/DisplayFilters).

ip.addr==192.168.0.1: Filtra todo el tr´afico con direcci´on IP 192.168.0.1 como origen o destino.

tcp.port==80: Filtra todo el tr´afico con el puerto 80 como origen o destino.

ip.src==192.168.0.1 and ip.dst==10.100.1.1: Filtra todo el tr´afico que tiene co- mo origen 192.168.0.1 y destino 10.100.1.1.

icmp: Filtra s´olo el tr´afico del protocolo ICMP. http: Filtra s´olo el tr´afico del protocolo HTTP. dns: Filtra s´olo el tr´afico del protocolo DNS. sip: Filtra s´olo el tr´afico del protocolo SIP.

## ICMP y Ping

Empecemos con un primer ejercicio para contextualizar.

***Ejercicio 1 [0,25p]:*** *¿Cu´ales son las direcciones MAC e IPv4 de tu ordenador? Realiza una captura de pantalla del s´ımbolo de sistema (terminal), donde se puedan leer todas las direcciones IPv4 y MAC de tu equipo. ¿Cu´al es el comando que has utlizado? Recuerda que para acceder al s´ımbolo de sistema, si est´as 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 direccion IPv4 10.0.0.17 y la direccion MAC o direccion fisica A6-97-F1-64-5D-AB

*Captura de pantalla de computadora

Descripción generada automáticamente*

Sigamos con una primera captura de los paquetes generados al realizar un ping. En concreto, vamos a realizar un ping desde el s´ımbolo de sistema (o terminal). Una vez abierta la consola, ejecuta el comando ping [www.uoc.edu](http://www.uoc.edu/) (o la p´agina web que prefieras). Si est´as utilizando Linux o MacOS, debes abrir el terminal y ejecutar el comando ping -c 4 [www.uoc.edu](http://www.uoc.edu/) (donde el par´ametro c del comando ping permite especificar el nu´mero de paquetes ping a enviar). Despu´es de haber ejecutado el ping correctamente, se puede detener la captura en Wireshark pulsando el bot´on 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 continuaci´on, haz clic en *Apply* a la derecha del filtro. Wireshark pasa a mostrar solo los mensajes relativos al protocolo ICMP.

***Ejercicio 2 [0,25p]:*** *¿Cu´antos paquetes ICMP se observan? Adjunta una captura de pantalla al respecto. ¿C´omo se identifica un mensaje ICMP en el datagrama IP? ¿Cu´al es el taman˜o de las peticiones que manda por defecto el comando ping? Indica cu´al ser´ıa 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

*Interfaz de usuario gráfica, Aplicación, Tabla, Excel

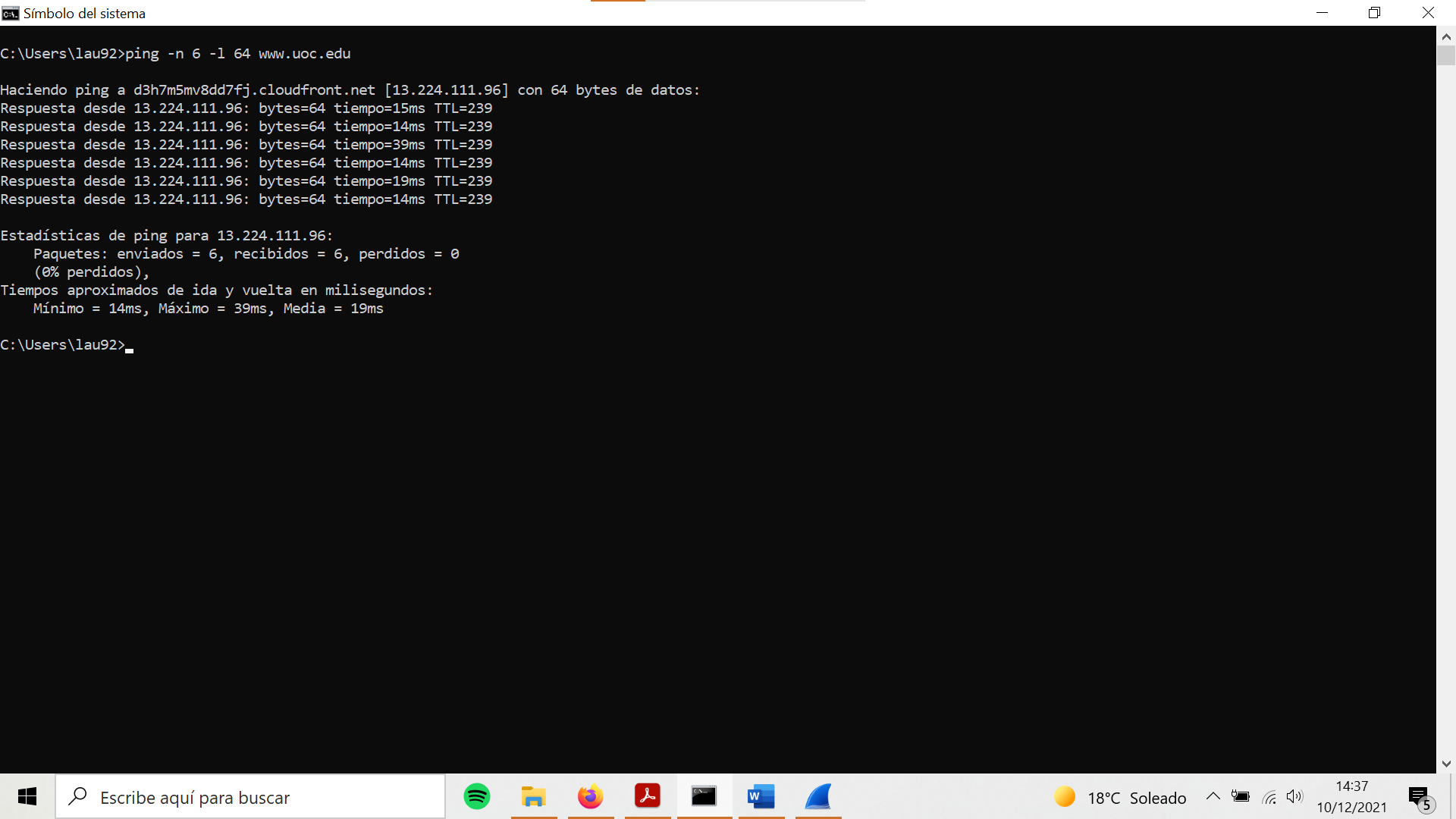
Descripción generada automáticamente*

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

*Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente*

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 .



# 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 est´andares y abiertos - cualquiera puede acceder a la especificaci´on (se encuentran en Internet) e implementarla. La principal orga- nizaci´on que se encarga de definir estos est´andares 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 est´andares y los publican en las llamadas RFC (Request for Comments). Como ejemplo, el protocolo TCP inicialmente se defini´o en la RFC nu´mero 793, publicado en 1981, y posteriormente se fueron an˜adiendo funcionalidades adicionales en nuevas RFCs.

***Ejercicio 3 [0,25p]:*** *Indica cu´ales son los nu´meros de RFC para los siguientes protocolos y, para cada uno de ellos, indica el an˜o de publicaci´on:*

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 caracter´ıstica del protocolo SCTP que lo diferencie de TCP.*

*¿Qu´e 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 t**olerancia 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 p´agina web. Es momento de introducir el protocolo HTTP.

## Comportamiento del protocolo HTTP

El protocolo HTTP (Hypertext Transfer Protocol) es un protocolo de aplicaci´on para transmitir informaci´on en la web a trav´es 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 m´as) en combinaci´on con otros protocolos, como por ejemplo el SSL (Secure Sockets Layer) y m´as recientemente el TLS (Transport Layer Security). A nivel de la arquitectura de red, el protocolo TLS se situ´a entre la capa TCP/IP y el HTTP. Esta combinaci´on da lugar al protocolo HTTPS, tambi´en conocido como HTTP over TLS.

Para generar tr´afico HTTP tan solo tenemos que introducir en nuestro navegador de Internet habitual la URL de una p´agina web HTTP (y no HTTPS). Un ejemplo de web HTTP es [http:](http://www.washington.edu/)

[//www.washington.edu/](http://www.washington.edu/) .

Cuando un navegador intenta acceder a una p´agina web realiza una petici´on a un servidor HTTP. El protocolo HTTP tiene dos m´etodos principales para que el cliente (navegador) realice esta petici´on al servidor: el m´etodo POST y el m´etodo GET. Si todo ha ido bien, tras recibir esta petici´on, el servidor web donde se encuentra alojada la p´agina web contesta satisfactoriamente a trav´es del mensaje 200 OK. Para ello, encapsula en un mensaje HTTP el c´odigo HTML que contiene la informaci´on 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* ). Ver´as que al navegar por alguna de estas p´aginas el nu´mero de paquetes HTTP aumenta sensiblemente.[2](#_bookmark2) Despu´es de haber capturado paquetes durante aproximadamente un par de minutos det´en 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 petici´on POST o GET y su respuesta correspondiente. Contesta a las siguientes preguntas: 1) ¿Qu´e mensaje HTTP es el mensaje de respuesta a la petici´on GET o POST? 2)*

*¿Cu´anto tiempo ha pasado entre la petici´on del cliente y la respuesta del servidor? 3) ¿Cu´al es el nu´mero asignado por Wireshark al mensaje de petici´on y al de respuesta? 4) ¿Puedes identificar la direcci´on IP del cliente? ¿Y la direcci´on IP del servidor? 5) Por u´ltimo, indica qu´e versi´on del protocolo HTTP estamos utilizando.*

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

2Si no consigues capturar ningu´n paquete HTTP al entrar en estas p´aginas, borra el historial de cookies de tu navegador e int´entalo de nuevo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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 numero signado por wireshark al mensaje de petición es 392 y el numero agisnado almensaje 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. Ayud´andote de los documentos de la capa de red que hemos visto en el reto 2, identifica los elementos de la cabecera IP y descr´ıbelos.*

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

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
5. Identificación 0xba2d es el numero de secuencia del datagrama
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 s´olo los que tengan tu direcci´on IP como origen e incluye una captura de pantalla.*

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

***Ejercicio 8 [0,75p]:*** *Manteniendo el filtro de tu direcci´on IP, identifica la direcci´on MAC de tu ordenador. ¿Cu´al es? En los varios paquetes filtrados, salientes de tu ordenador, identifica tambi´en la direcci´on MAC de destino. ¿A qu´e dispositivo corresponde? Incluye una captura de pantalla donde se pueda ver la direcci´on MAC de origen y destino.*

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

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. Ayud´andote con los documentos de la capa de transporte que hemos visto en el reto 3, identifica los elementos de la cabecera TCP y desr´ıbelos brevemente.*

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

**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
6. Suma de comprobación (16 bits): 0x8784 el campo de suma de comprobación de 16 bits se utiliza para la comprobación de errores del encabezado y los datos
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

***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 conexi´on HTTP que has iniciado. ¿Hay alguna raz´on por la que se use ese puerto? ¿Cambiaría ese puerto si en vez de conectarnos contra un servidor HTTP lo hiciéramos contra un servidor HTTPS?*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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 menu´ Statistics-Protocol Hierarchy de Wireshark podr´as obtener una lista porcentual de los diferentes paquetes capturados. En particular podr´as analizar qu´e 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, ¿qu´e porcentaje de paquetes usa TCP como transporte, y qu´e porcentaje usa UDP? Incluye una captura de pantalla que muestre esta estad´ıstica. 2) Teniendo en cuenta esta estad´ıstica, ¿qu´e 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 UPD. 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.

*Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media*

Vamos a analizar ahora algunos aspectos b´asicos del protocolo TCP. Cuando dos nodos se co- munican mediante TCP, se establece una conexi´on 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 bu´squeda a trav´es de tu navegador web. Filtra los paquetes del protocolo TCP y, para mayor claridad, f´ıltralos con tcp.port==80. Se visualizar´an 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) ¿Cu´al es el nu´mero de secuencia del segmento TCP SYN? (Identifica el sequence number raw, y no el relativo.) 3) ¿Cu´al es el nu´mero de secuencia (raw) del SYN ACK y el nu´mero de acknowledgement enviado por el servidor? ¿C´omo se han determinado estos valores? 4) ¿Cu´al es el nu´mero de acknowledgement del siguiente ACK? ¿C´omo se ha determinado este valor? Acompan˜a tus respuestas de una captura de pantalla donde se pueda observar lo descrito.*

El cliente envía un pequeño segmento TCP al servidor , cuyo sgmento SYN sien el valor de 1 añadido de un initial sequence number (client\_isn) escogido de forma aleatoria como numero 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 el campo de reconocimiento del encabezado (ACK#) del segmento TCP se establece en client\_isn + 1 y añade su propio initial seguence number (server\_isn)

Finalmente el cliente recibe estos datos conocidos en su conjunto como SYNACK segment. Y envían un ultimo segmento al servidor con el SYN con valor de 0 el numero de reconocimiento de encabezado con el valor server\_isn+1. La conexión esta establecida

2) TCP SYN

Obetenemos el numero de forma aleatoria 2852913009

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

4) Siguiente ACK

Calculado de sequence number del servidor +1 = 1677759826

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Comportamiento del protocolo DNS (Domain Name System)

Volvamos a Wireshark y prepar´emoslo para una nueva captura. Antes de proceder, es recomen- dable vaciar la cach´e del DNS para asegurarnos de que la petici´on 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 p´aginas web como pueden ser [https://webrtchacks.](https://webrtchacks.com/) [com/](https://webrtchacks.com/), <https://www.ietf.org/>, <http://phdcomics.com/>u otras. Una vez tengas varias p´aginas abiertas, det´en la captura de Wireshark.

***Ejercicio 13 [0,25p]:*** *Describe el principal objetivo del protocolo DNS. Con la ayuda del menu´ Sta- tistics de Wireshark, explica cu´antos paquetes DNS has capturado en total y su taman˜o m´ınimo, medio y m´aximo. Incluye una captura de pantalla de la informaci´on estad´ıstica 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 caturado 488 paquetes con un tamao medio de 88.34 minimo de 21 y máximo de 353

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Filtra ahora la captura por el protocolo DNS.

***Ejercicio 14 [0,5p]:*** *Identifica las peticiones DNS y sus mensajes de respuesta. 1) ¿Se env´ıan a trav´es de TCP o de UDP? 2) ¿Cu´al es el puerto de destino de las peticiones DNS? ¿Cu´al 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 peticione 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

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

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

***Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente***

***Ejercicio 15 [0,25p]:*** *¿Cu´al es la direcci´on IP de tu servidor DNS? Justifica la respuesta acom- pan ´andola de una captura de pantalla de Wireshark.*

*Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente*

## Comportamiento del protocolo SIP (Session Initiation Protocol)

SIP (Session Initiation Protocol) es un protocolo del nivel de aplicaci´on que permite inicializar, modificar y finalizar sesiones interactivas que impliquen elementos multimedia como v´ıdeo, voz, mensajer´ıa instant´anea, juegos en l´ınea, realidad virtual, etc.

Las operadoras de telecomunicaci´on usan el protocolo SIP para sen˜alizar el tr´afico en sus redes m´as modernas de telefon´ıa (ya sean fijas o m´oviles) y lo acompan˜an 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, cu´al ha sido la calidad del audio, etc) se reporta en los llamados CDR (Call Detailed Records) generando un alto volumen de datos. Esta informaci´on la utilizan las operadoras para tomar decisiones sobre cuestiones operativas (dimensionamientos y capacidades de red, identificar problemas, cambiar l´ogicas 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 menu´ ’Telefon´ıa’ y selecciona la opci´on ’Flujos SIP’. Selecciona el primer flujo (lo puedes identificar, ya que su estado est´a marcado como ’COMPLETED’) y clica en la opci´on ’Flow Sequence’.

***Ejercicio 16 [0,25p]:*** *¿Cu´antos mensajes SIP puedes ver en el diagrama mostrado? Indica cu´ales son.*

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

*Tabla

Descripción generada automáticamente*

***Ejercicio 17 [0,25p]:*** *Identifica cu´ales son las direcciones IP de los dos tel´efonos que participan en esta llamada.*

Proviene de 10.0.2.20 a 10.0.2.15

***Ejercicio 18 [0,25p]:*** *¿Cu´anto 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 aplicaci´on basada en SIP, queremos iniciar un stream de video desde nuestra Webcam para conectarnos a la plataforma lanzada re- cientemente por Facebook, Meta. La Webcam genera una resoluci´on de 640x480 p´ıxeles por frame, donde cada p´ıxel se representa con 3 colores, y cada color con 8 bits. El video resultante tiene una frame rate de 20 frames por segundo.*

* + 1. ¿Cu´al es el ancho de banda necesario para transmitir el video sin compresi´on? ¿Cu´al es si aplicamos un algoritmo de compresi´on con un ratio 40:1

640 x 480 x 3 x 8 bit (no comprimido) =7.372.800 bit /frame x 20 frames/sec= 147456000 bit/sec = 144 Mbps

A la compresión 40:1

144/40 Mbps =3.6 Mbps

* + 1. ¿Qu´e ratio de compresi´on ser´ıa necesario si quisi´eramos transmitir el video sobre un enlace de 512 kbps?

144Mbps / 512 kbps (0.5 Mbps) = 288 :1

# Extracci´on de datos desde una API

API es el acr´onimo en ingl´es de *application programming interface*. La API es b´asicamente una interfaz que habla por nosotros con un programa. Cuando un programador decide poner a dispo- sici´on pu´blica algu´n conjunto de datos generados por sus programas, lo puede hacer a trav´es de una interfaz de este tipo. Otros programadores extraen estos datos de la aplicaci´on construyendo URLs o a trav´es de clientes HTTP. Una API est´a caracterizada por tres elementos:

el **usuario**, que es la persona realizando la petici´on;

el **cliente**, que es el ordenador que env´ıa la petici´on al servidor; el **servidor**, que es el ordenador que responde a la petici´on.

Los datos se almacenan en un servidor y la documentaci´on correspondiente se pone a disposici´on 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 disposici´on para que los desarrolladores puedan acceder a ellos y manipularlos.

A trav´es de una API conocida como RESTful API, podemos realizar peticiones (*queries*) utilizando el protocolo HTTP. ¿C´omo? Lo haremos realizando *HTTP-requests* y *HTTP-responses*. Concreta- mente, las APIs pueden realizar un nu´mero limitado de acciones entre las que encontramos:

**GET**: solicita datos a un servidor;

**POST**: env´ıa cambios desde el cliente al servidor;

**PUT**: revisa o an˜ade a la informaci´on existente;

**DELETE**: destruye informaci´on existente.

¿A qu´e tipos de datos podemos acceder? Por ejemplo, con Facebook podr´ıamos obtener la lista de amigos entre un amigo nuestro y nosotros mismos, o el nu´mero de likes obtenidos en ciertas p´aginas. A Twitter le podr´ıamos pedir la lista de temas y argumentos m´as tratados en una cierta

´area geogr´afica, por ejemplo en Espan˜a. A trav´es de la Google Direction API podemos obtener rutas para llegar de una direcci´on a otra. Por u´ltimo, desde la Google Book API, podr´ıamos obtener la lista de libros de un autor, etc.

La mayor´ıa de aplicaciones mencionadas requieren gestionar una autorizaci´on que se tiene que pedir, concret´andose en una contrasen˜a o clave de la API. Las API pu´blicas tambi´en tienen limita- ciones en cuanto a cantidad de datos disponibles y nu´mero de peticiones por segundo posibles. Los datos que obtenemos se representan a trav´es de un formato espec´ıfico conocido como JSON (Java Script Object Notation). JSON es un formato est´andar que se usa para transmitir datos entre un servidor y una aplicaci´on 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 cach´e del DNS (ver Secci´on 2.2).

Realicemos ahora una petici´on a la API que nos devuelva informaci´on 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 direcci´on IP del servidor de la API FruityVice. Filtramos los paquetes capturados por DNS y buscamos la petici´on al servidor DNS local de la direcci´on IP del servidor [www.fruityvice.com.](http://www.fruityvice.com/)

***Ejercicio 20 [0,25p]:*** *¿Cu´al es la direcci´on IP que resuelve el dominio* [*www.fruityvice.com?*](http://www.fruityvice.com/) *Adjunta una captura de pantalla indicando el paquete donde se encuentra esta respuesta.*

***Ejercicio 21 [0,5p]:*** *Teniendo en cuenta la informaci´on que has derivado en el ejercicio anterior (sobre la direcci´on IP que resuelve el dominio de inter´es), estima cu´anto tiempo ha durado la descarga (en milisegundos) de los datos sobre las frutas desde esa direcci´on IP. Adjunta una captura de pantalla de Wireshark para justificar tu respuesta.*

***Ejercicio 22 [0,25p]:*** *Explorando la captura del ejercicio anterior, indica si, teniendo visibili- dad de lo que est´a sucediendo en la red (es decir, lo que vemos en Wireshark), es posible leer el contenido de la informaci´on que se est´a transmitiendo: en este caso la lista de las frutas con sus caracter´ısticas. Justifica tu respuesta.*

Imaginemos ahora que hemos fundado nuestra propia startup para ofrecer, a trav´es de Internet, una API REST que expone datos sobre la evoluci´on, cotizaci´on y transacciones de distintos tipos de criptomoneda. Esta API est´a disen˜ada para que la consuman aplicaciones de trading que, con la informaci´on que extraigan de la API, van a generar recomendaciones de inversi´on personalizadas para cada uno de sus clientes. Llevamos poco tiempo con este proyecto y, aunque tenemos a muchos clientes que est´an 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 cu´ales son nuestros costes de operaci´on. Por este motivo, como se indica en la Figura [3,](#_bookmark5) vamos a exponer nuestra API a trav´es de un u´nico servidor que va a disponer de una conexi´on a Internet limitada a un ancho de banda

de subida de *rS* = 100 Mb/s[3](#_bookmark3)(sentido servidor-clientes). El contenido completo de la informaci´on que expondremos v´ıa nuestra API tiene un taman˜o de *S* = 200 MB[4](#_bookmark4). Imaginemos que Elon Musk, fundador de varias compan˜´ıas entre las que se encuentra Tesla, publica un Tweet anunciando que a partir del pr´oximo an˜o Tesla va a dejar de aceptar moneda tradicional y los pagos deber´an hacerse u´nicamente con Bitcoin.



Figura 2: Tweet imaginario de Elon Musk

Esto, autom´aticamente, genera un alto volumen de transacciones de gente intentando comprar Bit- coin en un espacio de tiempo muy corto. Todos nuestros clientes necesitan actualizar sus directrices de inversi´on inmediatamente, por lo que acceden de forma concurrente a nuestro servidor para des- cargar la informaci´on actualizada. Las computadoras de nuestros clientes se encuentran distribuidas a trav´es de *NA* = 150 redes de acceso, de forma que cada red tiene *NC* = 10 computadoras.

***Ejercicio 23 [0,5p]:*** *Considera que cada computadora de cliente descarga la informaci´on actuali- zada una u´nica vez y que cada petici´on de actualizaci´on devuelve el contenido completo de la API. Calcula:*

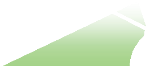
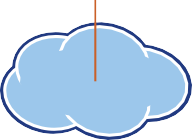
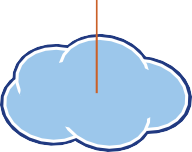
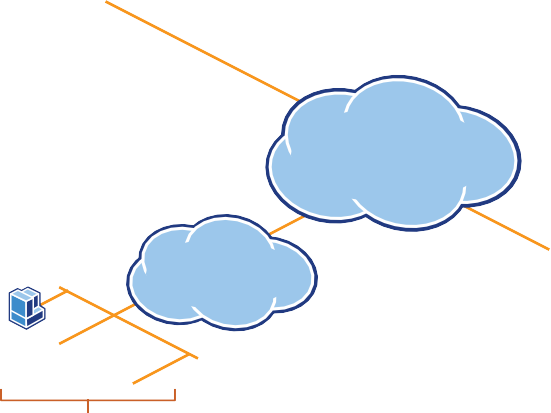
* + - 1. El volumen total de tr´afico enviado desde nuestro servidor, *Tcs,ul*
      2. El volumen de tr´afico recibido por cada una de las redes de acceso, *Tcs,a*
      3. La velocidad de descarga para cada cliente, *rcs,dl*
      4. El tiempo total de descarga para cada cliente, *tdl*

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 inversi´on en tiempo real. Para ello, hablamos con nuestro equipo de ingenier´ıa 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. Despu´es de con- siderar distintas alternativas tecnol´ogicas y sus costes asociados, decidimos llegar a un acuerdo con

31 Mb*/*s = 106 bits*/*s.

41 MB = 106 bytes = 8 *·* 106 bits.

Client



Server

Access

network

Core

network

Access

network

Access

network

Bottleneck 100 Mbps

*NC* = 10 clients *NC* = 10 clients

*NA* = 150 access networks

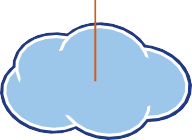
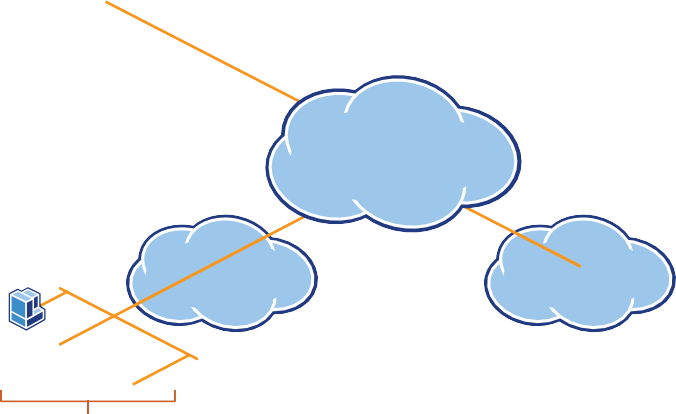
Figura 3: Arquitectura Cliente/Servidor.

una empresa de distribuci´on de contenidos, tambi´en conocidas como CDN (Content Distribution Networks), tal y como se puede ver en la Figura [4.](#_bookmark6) Esta empresa dispone de un servidor CDN en cada red de acceso, por lo que estos, en todo momento, van a mantener una copia actualiza- da 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 informaci´on. La velocidad m´axima de subida para estos servidores de la CND es de *rS* = 100 Mb/s (sentido servidorCDN-clientes).

***Ejercicio 24 [0,5p]:*** *calcula las mismas variables que en el ejercicio anterior, ahora denotadas como Tcdn,ul, Tcdn,a, rcdn,dl, tcdn,dl.*

***Ejercicio 25 [0,5p]:*** *Considera el caso de uso de los dos ejercicios anteriores. ¿Qu´e formato de datos seleccionar´ıas para entregar los datos v´ıa nuestra API REST? Justifica tu respuesta y proporciona un ejemplo de mensaje para este caso de uso.*

Client



Server

Access

network

Core

network

CDN

Server

CDN

Server

*NC* = 10 clients

*NC* = 10 clients

Bottleneck 100 Mbps

Bottleneck 100 Mbps

File size 200 MB

*NA* = 150 access networks

Figura 4: Arquitectura CDN.