**2. ¿Cuál es la función de la capa de aplicación?**

Provee servicios de comunicación a los usuarios.

Interfaz con el usuario (UI) u otras aplicaciones/servicios

Ofrece a las aplicaciones (de usuario o no) la posibilidad de acceder a los servicios de las demás capas y define los protocolos que utilizan las aplicaciones para intercambiar datos, como correo electrónico (POP y SMTP), gestores de bases de datos y protocolos de transferencia de archivos (FTP).

**3. Si dos procesos deben comunicarse:**

a. ¿Cómo podrían hacerlo si están en diferentes máquinas?

Se pueden comunicar mediante el intercambio de mensajes a través de una red de computadoras.

Un proceso emisor crea y envia mensajes a la red, mientras que el receptor recibe los mensajes y responde con otros mensajes.

b. Y si están en la misma máquina, ¿qué alternativas existen?

Si dos procesos están en la misma maquina entonces pueden comunicarse por medio del sistema de comunicación del sistema operativo.

**4. Explique brevemente cómo es el modelo Cliente/Servidor. De un ejemplo de un sistema Cliente/Servidor en la “vida cotidiana” y un ejemplo de un sistema informático que siga el modelo Cliente/Servidor.**

¿Conoce algún otro modelo de comunicación?

**5. Describa la funcionalidad de la entidad genérica “Agente de usuario” o “User agent”.**

**HTTP**

**6. Observe el indice de la RFC2616, busque el apartado donde se describe el requerimiento y la respuesta.**

¿Qué son y en qué se diferencian HTML y HTTP? ¿En que entidad ubicaría a HTML?

<https://www.w3.org/Protocols/rfc2616/rfc2616.html>

La principal diferencia entre HTML y HTTP es que HTML es un formato de texto mientras que HTTP es un protocolo de la capa de aplicación. En otras palabras, las paginas web pueden contener archivos HTMl, HTTP va a definir cómo los clientes web solicitan estas paginas a los servidores y cómo los servidores transfieren esas páginas a los clientes.

Cuando un usuario solicita una página web (por ejemplo, haciendo clic en un hipervínculo), el navegador envía al servidor mensajes de solicitud HTTP, pidiendo los objetos contenidos en la página. El servidor recibe las solicitudes y responde con mensajes de respuesta HTTP que contienen los objetos.

**7. Utilizando la VM, abra una terminal e investigue sobre el comando curl. Analice para qué sirven los siguientes parámetros (-I, -H, -X, -s).**

El comando "curl" nos permite transferir datos desde o hacia un servidor utilizando protocolos. Funciona sin interaccion del usuario y permite acceder a información de envio de mensajes de terminales; por ejemplo, los headers.

Los parametros que se pueden utilizar son:

-I: muestra el header HTTP del response. --head

-H: sirve para agregar/modificar información al header del request que se va a enviar. –header

*curl -H "X-First-Name: Joe" https://example.com*

-X: especifica el método a utilizar en el request

-s: Silent or quiet mode. Do not show progress meter or error messages.

-v : Makes curl verbose during the operation. Useful for debugging and seeing what's going on "under the hood". A line starting with '>' means "header data" sent by curl, '<' means "header data" received by curl that is hidden in normal cases, and a line starting with '\*' means additional info provided by curl.

<https://curl.se/docs/manpage.html>

**8. Ejecute el comando curl sin ningún parámetro adicional y acceda a www.redes.unlp.edu.ar. Luego responda:**

a. ¿Cuántos requerimientos realizó y qué recibió? Pruebe redirigiendo la salida(>) del comando curl a un archivo con extensión html y abrirlo con un navegador.

Se realizó un único request (con el método GET) y se recibió texto plano con formato HTML. Si se abre con un navegador se obtiene una página web sin diseño.

b. ¿Cómo funcionan los atributos href de los tags link e img en html?

Los atributos href contienen URL de un atributo, los mismos son vinculos a objetos dentro (img) o fuera (link) de la pagina. Cuando los navegadores encuentran estos elementos en el archivo HTML hacen una solicitud al servidor para obtenerlos.

c. Para visualizar la página completa con imágenes como en un navegador, ¿alcanza con realizar un único requerimiento? ¿Cuántos requerimientos serían necesarios para obtener una página que tiene dos CSS, dos Javascript y tres imágenes? Diferencie como funcionaría un navegador respecto al comando curl ejecutado previamente.

No, no es necesario ya que debo hacer un request por cada elemento. Para obtener una página así debo realizar 9 requerimientos:

2 por cada CSS

2 por cada JS

3 por cada imagen

1 por el HTML

1 por el favicon

Cuando ejecutamos el comando curl www.redes.unlp.edu.ar la herramienta curl hará una unica peticion al recurso, mientras que el navegador hará varios request para cargar los distintos recursos de la pagina.

**9. Ejecute a continuación los siguientes comandos:**

**curl -v -s www.redes.unlp.edu.ar > /dev/null**

**curl -I -v -s www.redes.unlp.edu.ar**

**Observe la salida y luego repita la prueba, pero previamente inicie una nueva captura en wireshark.**

Utilice la opción Follow Stream. ¿Qué se transmitió en cada caso?

¿A que se debió esta diferencia entre lo que se transmitió y lo que se mostró en pantalla?

En el primer comando, se transmitió todo el html de la página, ya que se usó el método GET; y no se mostró el cuerpo de la respuesta debido a que se redirige a /dev/null, que lo descarta. En el segundo comando, se usó el método HEAD, por lo que únicamente se transmitieron los headers.

**10. Investigue cómo define las cabeceras la RFC.**

Each header field consists of a name followed by a colon (":") and the field value. Field names are case-insensitive. The field value MAY be preceded by any amount of LWS, though a single SP is preferred. Header fields can be extended over multiple lines by preceding each extra line with at least one SP or HT. Applications ought to follow "common form", where one is known

a. ¿Establece todas las cabeceras posibles?.

Headers? Establece los más normales. Se pueden usar headers personalizados.

b. ¿Cuántas cabeceras viajaron en el requerimiento y en la respuesta del ejercicio anterior?

en el requerimiento y

c. ¿La cabecera Date es una de las definidas en la RFC? ¿Qué indica?

Si. La cabecera Date indica la fecha y hora en la que se realizó el request.

**11. Utilizando curl, realice un requerimiento con el método HEAD al sitio www.redes.unlp.edu.ar e indique:**

a. ¿Qué información brinda la primer línea de la respuesta?

La versión de http, el status code y el reason phrase/status text.

b. ¿Cuántos encabezados muestra la respuesta?

7

c. ¿Qué servidor web está sirviendo la página?

Server: Apache/2.4.56 (Unix)

d. ¿El acceso a la página solicitada fue exitoso o no?

Si, porque el código de respuesta fue 200.

e. ¿Cuándo fue la última vez que se modificó la página?

Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT

f. Solicite la página nuevamente con curl usando GET, pero esta vez indique que quiere obtenerla sólo si la misma fue modificada en una fecha posterior a la que efectivamente fue modificada. ¿Cómo lo hace? ¿Qué resultado obtuvo? ¿Puede explicar para qué sirve?

GET direccional:

The If-Modified-Since request HTTP header makes the request conditional: the server sends back the requested resource, with a 200 status, only if it has been last modified after the given date.

If-Modified-Since: <day-name>, <day> <month> <year> <hour>:<minute>:<second> GMT

Si no fue modificada, se devuelve un 304 Not Modified.

*curl -I -H "If-Modified-Since:Sun, 20 Mar 2023 19:04:46 GMT" www.redes.unlp.edu.ar*

**12. En HTTP/1.0, ¿cómo sabe el cliente que ya recibió todo el objeto solicitado completamente? ¿Y en HTTP/1.1?**

En HTTP/1.0 el cliente puede saber si recibió todo el objeto si el servidor cierra la conexión TCP.

HTTP/1.1 Con el encabezado *content-length*, que indica en bytes el largo de la respuesta.

**13. Investigue los distintos tipos de códigos de retorno de un servidor web y su significado en la RFC. ¿Qué parte se ve principalmente interesada de esta información, cliente o servidor? ¿Es útil que esté detallado y clasificado en la RFC?. Dentro de la VM, ejecute los siguientes comandos y evalue el estado que recibe.**

curl -I <http://unlp.edu.ar>

*HTTP/1.1 301 Moved Permanently*

*Location: https://unlp.edu.ar/*

curl -I [www.redes.unlp.edu.ar/restringido/index.php](http://www.redes.unlp.edu.ar/restringido/index.php)

*HTTP/1.1 401 Unauthorized*

*WWW-Authenticate: Basic realm="Authentication Required"*

The HTTP WWW-Authenticate response header defines the HTTP authentication methods ("challenges") that might be used to gain access to a specific resource.

curl -I [www.redes.unlp.edu.ar/noexiste](http://www.redes.unlp.edu.ar/noexiste)

Los códigos de retorno son:

200: Peticiones correctas

300: Redirecciones

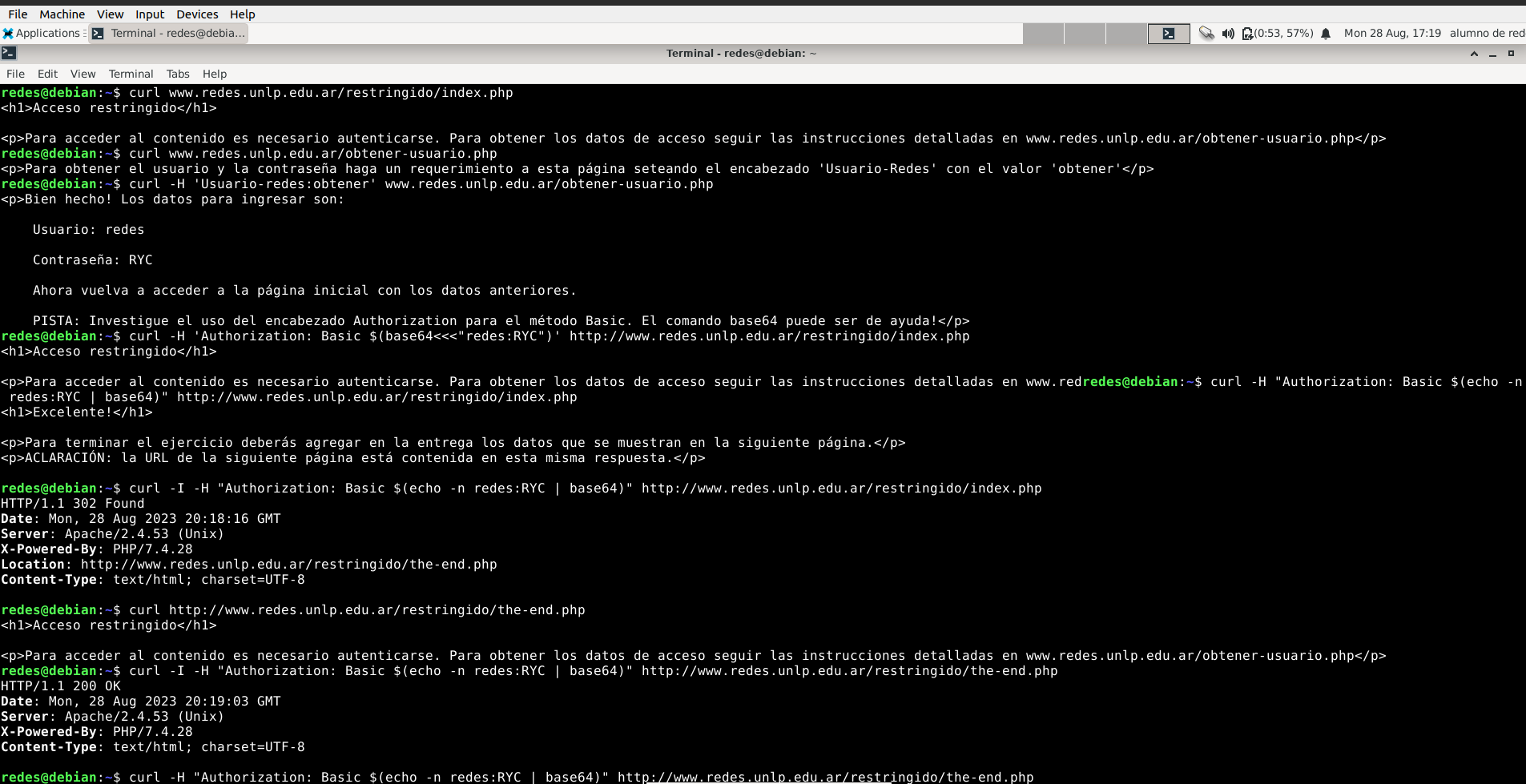
400: Errores del cliente

500: Errores del servidor

El mas interesado es el cliente. Es útil que esté detallado en el RFC para tener un estándar en común para la comunicación.

**14. Utilizando curl, acceda al sitio www.redes.unlp.edu.ar/restringido/index.php y siga las instrucciones y las pistas que vaya recibiendo hasta obtener la respuesta final. Será de utilidad para resolver este ejercicio poder analizar tanto el contenido de cada página como los encabezados.**

Done.



**15. Utilizando la VM, realice las siguientes pruebas:**

a. Ejecute el comando ’curl www.redes.unlp.edu.ar/extras/prueba-http-1-0.txt’ y copie la salida completa (incluyendo los dos saltos de linea del final).

Llamada normal get. Devuelve una petición GET “literal”.

b. Desde la consola ejecute el comando telnet www.redes.unlp.edu.ar 80 y luego pegue el contenido que tiene almacenado en el portapapeles. ¿Qué ocurre luego de hacerlo?

Recibo una respuesta por parte del servidor.

c. Repita el proceso anterior, pero copiando la salida del recurso /extras/prueba-http-1-1.txt. Verifique que debería poder pegar varias veces el mismo contenido sin tener que ejecutar telnet nuevamente.

Con http1.0 el servidor cierra la conexión instantaneamente cuando envía la respuesta, en cambio con http1.1 el servidor espera 5 segundos para cerrar la conexión.

**16. En base a lo obtenido en el ejercicio anterior, responda:**

¿Qué está haciendo al ejecutar el comando telnet? ¿Qué lo diferencia con curl?

Telnet permite el control remoto de los ordenadores por medio de entradas y salidas basadas en texto, conectándose de forma predeterminada al puerto 23 mediante TCP. La diferencia con curl es que con telnet tenemos que escribir a mano el texto respetando el protocolo correspondiente (ej HTTP)

Observe la definición de método y recurso en la RFC. Luego responda, ¿Qué método HTTP utilizó?

Metodo get

¿Qué recurso solicitó?

/http/HTTP-1.1/

¿Qué diferencias notó entre los dos casos? ¿Puede explicar por qué?

Con http1.0 el servidor cierra la conexión instantaneamente cuando envía la respuesta, en cambio con http1.1 el servidor espera 5 segundos para cerrar la conexión.

¿Cuál de los dos casos le parece más eficiente? Piense en el ejercicio donde analizó la cantidad de requerimientos necesarios para obtener una página con estilos, javascripts e imágenes. El caso elegido, ¿puede traer asociado algún problema?

Es mas eficiente Http1.1 porque se puede mantener una conexión para solicitar varios recursos. En cambio con http1.0 se establecería una conexión por cada recurso solicitado.

Si, un problema con http1.1 es si un recurso es muy grande, podría bloquear la conexión (nombre del problema: Head Of Line blocking HOL). Una solución es el uso de multiplexación y pipeling para transmitir varios recursos al mismo tiempo.

**17. En el siguiente ejercicio veremos la diferencia entre los métodos POST y GET. Para ello, será necesario utilizar la VM y la herramienta Wireshark. Antes de iniciar considere: MUCHO\_TEXTO**

a. Abra un navegador e ingrese a la URL: www.redes.unlp.edu.ar e ingrese al link en la sección “Capa de Aplicación” llamado “Métodos HTTP”. En la página mostrada se visualizan dos nuevos links llamados: Método GET y Método POST. Ambos muestran un formulario como el siguiente:

b. Analice el código HTML.

c. Utilizando el analizador de paquetes Wireshark capture los paquetes enviados y recibidos al presionar el botón Enviar.

d. ¿Qué diferencias detectó en los mensajes enviados por el cliente?

Con el post, la

e. ¿Observó alguna diferencia en el browser si se utiliza un mensaje u otro?

**18. HTTP es un protocolo stateless, para sortear esta carencia muchos servicios se apoyan en el uso de Cookies. ¿En qué RFC se definió dicha funcionalidad? Investigue cuál es el principal uso que se le da la Set-Cookie y Cookie en HTTP. ¿Qué atributo de la RFC original fue en parte aprovechado para la implementación?**

Se definió originalmente en la RFC 2109. Se aprovecharon las cabeceras de Set-Cookie y Cookie.

El principal uso que se le da es el de mantener un estado persistente entre el cliente y servidor.

Ejemplo: *Set-Cookie: qwerty=219ffwef9w0f; Domain=somecompany.co.uk*

**19. ¿Cuál es la diferencia entre un protocolo binario y uno basado en texto? ¿de que tipo de protocolo se trata HTTP/1.0, HTTP/1.1 y HTTP/2?**

Un protocolo binario transmite datos en forma de patrones de bits que las máquinas pueden entender directamente. Suele ser más eficiente en términos de velocidad y uso de ancho de banda, ya que no se desperdicia espacio en caracteres de formato o en la interpretación de texto legible por humanos.

Un protocolo basado en texto envía datos como cadenas legibles utilizando caracteres como letras, números y símbolos. Estos caracteres suelen estar codificados en ASCII o UTF-8. Aunque esto hace que la comunicación entre sistemas sea más comprensible para los humanos, puede ser menos eficiente en términos de velocidad y uso de ancho de banda, ya que se requiere más información para representar los mismos datos que en formato binario.

HTTP/1.0, HTTP/1.1 son protocolos basados en textos mientras que HTTP/2 es un protocolo binario.

**20. Analice de que se tratan las siguientes características de HTTP/2: stream, frame, server-push**

Un **stream** es un canal virtual bidireccional que opera dentro de una única conexión TCP. Cada solicitud y respuesta se intercambia a través de un stream separado. Cada stream posee un ID y una prioridad asignada. Los mensajes HTTP/2 (solicitudes y respuestas) se transmiten utilizando estos streams.

Los mensajes HTTP/2 se dividen en **frames** que se envían a través del mismo stream. Básicamente, los streams son identificados y desglosados en frames, que constituyen la unidad de comunicación más pequeña. Estos frames se transmiten y se vuelven a ensamblar en el destino. Existen varios tipos de frames, como los de encabezado, de datos y de prioridad. Estos frames se combinan para formar las solicitudes y respuestas que se intercambian en los strefams. Todos los streams coexisten en una misma conexión.

El **server-push** es una característica que permite al servidor enviar recursos adicionales al cliente antes de que este los solicite de manera explícita. Por ejemplo, pueden ser recursos como hojas de estilo CSS o scripts JavaScript. No siempre es lo que necesita el cliente, depende de que funcionalidad este ofrece.

**21. Responder las siguientes preguntas:**

**a. ¿Qué función cumple la cabecera Host en HTTP 1.1? ¿Existía en HTTP 1.0? ¿Qué sucede en HTTP/2? (Ayuda:** [**https://undertow.io/blog/2015/04/27/An-in-depth-overview-of-HTTP2.html para HTTP/2**](https://undertow.io/blog/2015/04/27/An-in-depth-overview-of-HTTP2.html%20para%20HTTP/2)**)**

La cabecera “Host” en HTTP 1.1 tiene la función de indicar el nombre de dominio (se puede indicar opcionalmente el puerto) al que se está haciendo la solicitud dentro del encabezado de la petición. La inclusión de la cabecera "Host" en una solicitud es obligatoria. Esta es sumamente importante en servidores que alojan múltiples sitios web en una misma dirección IP, ya que permite al servidor saber a qué sitio se está dirigiendo la solicitud. También es importante mencionar que las cachés proxy web necesitan la información proporcionada por la línea de cabecera del host.

**b. ¿Cómo quedaría en HTTP/2 el siguiente pedido realizado en HTTP/1.1 si se está usando https?**

**GET /index.php HTTP/1.1**

**Host:** [**www.info.unlp.edu.ar**](http://www.info.unlp.edu.ar)

Quedaría:

*:method: get*

*:path: /index.php*

*:scheme: https*

*:authority: www.info.unlp.edu.ar*