Práctica 2

miércoles, 4 de septiembre de 2024 11:23

1. ¿Cuál es la función de la capa de aplicación?

Su principal función es proporcionar los servicios más cercanos al usuario final. Permite la comunicación entre aplicaciones en distintos sistemas mediante la red. Http es un protocolo de la capa de aplicación que facilita La transferencia de hipertexto y otros recursos entre un cliente y un servidor. Da servicios de comunicación A los usuarios y a las aplicaciones.

2. Si dos procesos deben comunicarse:
a. ¿Cómo podrían hacerlo si están en diferentes máquinas?

Usando protocolos de res como tcp/ip

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

3. Explique brevemente cómo es el modelo Cliente/Servidor. Dé 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?

En un modelo cliente/servidor La carga es compartida. El cliente se ocupa de procesar la interfaz y el servidor hace el resto del procesamiento. El servidor corre su servicio y espera pasivamente la conexión. Se habla de que es un modelo asimétrico de modo 1 a N o M a n con m<n (un sv, muchos clientes). Un ejemplo de esto es un browser.
Hay otros modelos como por ejemplo el p2p en el que la carga se distribuye entre todos los peers. Esto es tipo torrent.

4. Describa la funcionalidad de la entidad genérica "Agente de usuario" o "User agent".

Un agente de usuario es el componente de la capa de aplicación que actua en nombre del usuario final. Por ejempko, un navegador web seria un user agent

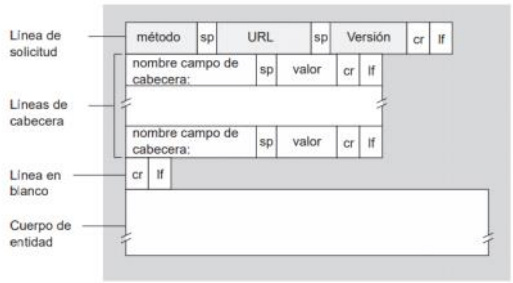
5. ¿Qué son y en qué se diferencian HTML y HTTP

Html es un lenguaje de marcas, se usa para estructurar paginas web. Http es un protocolo de red para transferir6datos. Dice como se piden Y mandan los recursos

6. HTTP tiene definido un formato de mensaje para los requerimientos y las respuestas. (Ayuda: apartado "Formato de mensaje HTTP", Kurose).

a. ¿Qué información de la capa de aplicación nos indica si un mensaje es de requerimiento o de respuesta para HTTP? ¿Cómo está compuesta dicha información?¿Para qué sirven las cabeceras?

La primera linea es de solicitud: tiene el método (get, put, post, head, delete), el url y la versión de http. Get es cuando el navegador solicita un objeto, el cual se especifica en el campo URL..
Las siguientes lineas son de cabecera. Por ejemplo host que dice dónde está el objeto, connection que dice si va a perdurar o no la conexión, user-agent que especifica el tipo de navegador que está haciendo la solicitud y accept-lenguaje dice que el usuario prefiere recibir un lenguaje especifico.



c. Suponga que desea enviar un requerimiento con la versión de HTTP 1.1 desde curl/7.74.0 a un sitio de ejemplo como www.misitio.com para obtener el recurso /index.html. En base a lo indicado, ¿qué información debería enviarse mediante encabezados? Indique cómo quedaría el requerimiento.

GET /index.html HTTP/1.1
Host: www.misitio.com
User-agent: curl/7.74.0
Accept: */* --> esto le dice que recibo lo que venga

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

- **-I (o --head)**: Este parámetro se utiliza para hacer una solicitud HTTP de tipo HEAD. Esto significa que curl solo solicitará los encabezados de la respuesta, sin descargar el cuerpo del contenido. Es útil para verificar la disponibilidad de una URL o para obtener metadatos del recurso, como el tipo de contenido o la longitud del mismo, sin descargar el recurso completo.

- **-H (o --header)**: Este parámetro se usa para agregar un encabezado HTTP personalizado a la solicitud. Por ejemplo, si necesitas enviar un encabezado de autenticación o modificar el encabezado User-Agent, usarías -H "Encabezado: valor". Puedes utilizarlo múltiples veces para agregar varios encabezados a la solicitud.
- **-X (o --request)**: Este parámetro permite especificar el método HTTP que deseas usar en la solicitud, como GET, POST, PUT, DELETE, etc. Por defecto, curl utiliza GET, pero con -X puedes cambiarlo a otro método dependiendo de la operación que deseas realizar en el servidor.
- **-s (o --silent)**: Este parámetro hace que curl funcione en modo silencioso. No muestra el progreso de la descarga ni los errores. Es útil si deseas que curl se ejecute sin mostrar salida en la terminal, especialmente cuando se usa en scripts donde la salida debe ser controlada.

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.

Hizo un solo requerimiento en el que obtuvo el html de la página.

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

El atributo href en un tag <link> especifica la URL de un recurso externo, como una hoja de estilo CSS o un icono de sitio. Este enlace permite que el navegador cargue y aplique el recurso al documento HTML. El img es para poner la dirección de la imagen a mostrar.
El navegador hace solicitudes http para descargar los recursos externos que se especifican y los usa de la forma que corresponda.

c. Para visualizar la página completa con imágenes como en un navegador, ¿alcanza con realizar un único requerimiento?
No, poque deberemos conseguir recursos externos como por ejemplo el css o el archivo javascript

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?

El -v da info del intercambio. > es lo que el cliente manda al sv, < el inverso
El primer comando redirige la salida a /dev/null por lo que lo descarta, el segundo lo muestra

10. Investigue cómo define las cabeceras la RFC.
a. ¿Establece todas las cabeceras posibles?.
No, define muchas de las más comunes y estándar utilizadas en las solicitudes y respuestas HTTP pero no todas.

b. ¿Cuántas cabeceras viajaron en el requerimiento y en la respuesta del ejercicio anterior?
4 en el requerimiento, 7 en la respuesta

c. ¿La cabecera Date es una de las definidas en la RFC? ¿Qué indica?
Si, indica la fecha y hora en que se generó la respuesta

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 linea de la respuesta?
Especifica que versión de http se está usando y que se estableció correctamente la conexión
HTTP/1.1 200 OK
b. ¿Cuántos encabezados muestra la respuesta?
7
c. ¿Qué servidor web está sirviendo la página?
Apache/2.4.56 (Unix)
d¿El acceso a la página solicitada fue exitoso o no?
si
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?
curl -v -H "If-Modified-Since: Mon, 20 Mar 2023 00:00:00 GMT" www.redes.unlp.edu.ar
El comando a usar es ese. Si se ingresa una fecha posterior a la última moficación no devuelve el html, en caso contrario si lo hace

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 hay un encabezado que se llama content-length que muestra el tamaño de la respuesta y permite que el cliente lea hasta que encuentre la cantidad especificada.
En http/1.1 está el Transfer-Encoding: chunked que manda los datos en chunks de tamaño variable. Cada uno arranca indicando el tamaño de su bloque en hexa. El cliente lee hasta encontrar un bloque de tamaño 0 lo que indica el final del cuerpo.

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
curl -I [redacted].ar/restringido/index.php
curl -I [redacted]

Está weno para el cliente porque le permite saber qué pasó con sus solicitudes y cómio manejar las respuestas. Es recontra necesario que estén estandarizados porque permiten la comunicación. Por ejemplo, los que son 1xx informan, 2xx éxito, 3xx redirección, 4xx error del cliente, etc.

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.

curl -u redes:RYC www.redes.unlp.edu.ar/restringido/index.php
-u manda user y psw codificado de una en base 64.

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 línea del final).
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?
Me da el html de la pagina?
c. Repita el proceso anterior, pero copiando la salida del recurso /extras/prueba-http-1-1.txt. Verifique que deberia poder pegar varias veces el mismo contenido sin tener que ejecutar telnet nuevamente.

Interpretación y Observaciones

- **HTTP/1.0 vs. HTTP/1.1**: En HTTP/1.0, cada solicitud abre una nueva conexión. En HTTP/1.1, las conexiones pueden ser persistentes (keep-alive), lo que permite múltiples solicitudes a través de la misma conexión. Esto significa que después de una solicitud, puedes seguir usando la misma conexión para más solicitudes sin tener que abrir una nueva conexión.
- **Contenido Reutilizable**: Cuando usas HTTP/1.1 con Connection: keep-alive, la misma conexión puede ser reutilizada para varias solicitudes. Esto se refleja en tu prueba, donde telnet mantiene la conexión abierta para múltiples solicitudes, y puedes pegar la misma respuesta repetidamente.
- **Solicitud HTTP Manual**: Si copias y pegas la misma solicitud en telnet sin cerrar la conexión, el servidor deberia responder con el mismo contenido. La capacidad de reutilizar la conexión es una caracteristica clave de HTTP/1.1.

????

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 en el puerto 80 abre una conexión directa con el sv. Sigue usando http pero no es tan automatizzado ni pijudo como curl
Observe la definición de método y recurso en la RFC. Luego responda, ¿Qué método HTTP utilizó?
¿Qué recurso solicitó?
Get
¿Qué diferencias notó entre los dos casos? ¿Puede explicar por qué?
Http 1.1 tiene conexiones persistentes mientras que el 1.0 no.
¿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?
Está bueno pq no hay que abrir constantemente la xonección para traerse cositas pero hay que tener cuidado con cuándo se cierran.

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:

POST: los parámetros están en el cuierpo

```
POST /http/metodos-lectura-valores.php HTTP/1.1
Host: www.redes.unlp.edu.ar
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 135
Origin: http://www.redes.unlp.edu.ar
Connection: keep-alive
Referer: http://www.redes.unlp.edu.ar/http/metodo-post.html
Upgrade-Insecure-Requests: 1

form_nombre=jose&form_apellido=martinez&form_mail=josemartinezosti@gmail.com&form_sexo=sexo_masc&form_pass=1234&form_confirma_mail=onHTTP/1.1 200 OK
Date: Tue, 10 Sep 2024 13:20:31 GMT
Server: Apache/2.4.56 (Unix)
X-Powered-By: PHP/7.4.33
Content-Length: 2641
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>M&eacute;todos HTTP: Lectura de valores desde REQUEST</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">

    <!-- Le styles -->
    <link href="../bootstrap/css/bootstrap.css" rel="stylesheet">
    <link href="../css/style.css" rel="stylesheet">
    <link href="../bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

    <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
      <script src="./bootstrap/js/html5shiv.js"></script>
    <![endif]-->
```

GET:los parámetros están en la barra de direcciones

```
GET /http/metodos-lectura-valores.php?form_nombre=jose&form_apellido=martinez&form_mail=joseaaaa&form_sexo=sexo_masc&form_pass=12345&form_confirma_mail=on HTTP/1.1
Host: www.redes.unlp.edu.ar
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.redes.unlp.edu.ar/http/metodo-get.html
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Tue, 10 Sep 2024 13:22:43 GMT
Server: Apache/2.4.56 (Unix)
X-Powered-By: PHP/7.4.33
Content-Length: 2624
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>M&eacute;todos HTTP: Lectura de valores desde REQUEST</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">

    <!-- Le styles -->
    <link href="../bootstrap/css/bootstrap.css" rel="stylesheet">
    <link href="../css/style.css" rel="stylesheet">
    <link href="../bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

    <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
      <script src="./bootstrap/js/html5shiv.js"></script>
    <![endif]-->
  </head>
```

Característica	GET	POST
Ubicación de los Datos	URL	Cuerpo de la solicitud
Longitud de los Datos	Limitada por la URL	Mayor capacidad de datos
Seguridad	Menos seguro (datos visibles en URL)	Más seguro (datos no visibles en URL)
Caché	Puede ser almacenado en caché	Generalmente no se almacena en caché
Idempotencia	Sí (repetir la solicitud no cambia el estado)	No (repetir puede cambiar el estado)
Uso Común	Solicitar datos, búsquedas, enlaces compartibles	Enviar datos, formularios, actualizaciones

18. Investigue cuál es el principal uso que se le da a las cabeceras Set-Cookie y Cookie en HTTP y qué relación tienen con el funcionamiento del protocolo HTTP.

Se usan sobre todo para gestionar ssiónses, autentificación y personalización. Con set-cookie se le da un valior a la cookie, después se pueden usar distintos parámetros para cambiarle cositas. Después cookie la usa el cliente para pedirle al servidor una cookie especifica. Mediante las cookies el sertvidor puede recordar al cliente. Como http tiene solicitudes que son independientes entre si, las cookies sirven para solventar las limitaciones que eso implica.

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

El que es basado en texto es legible mientras que el binario no. HTTP/1.0, HTTP/1.1 son basados en trexto y HTTP/2 no lo es. Que sea binario lo hace más chico y eficiente pero menos legibles por humnaos

20. Responder las siguientes preguntas:
a. ¿Qué función cumple la cabecera Host en HTTP 1.1? ¿Existia 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)
Host indica el nombre del dominio y el puerto al que se hace la solicitud. Se usa porque permite que varias páginas o dominios compartan un mismo ip.
En http/1.1 es obligatoria, en 1.0 no existia pq cada sitio tenia su propia ip. Algunos navegadores si lo implementaron pero no era obligatoria.
En http/2.0 la cabecera sigue existiendo pero se renombró a :authority

Resumen:

- **HTTP/1.0:** No tenía la cabecera `Host` como estándar, por lo que no soportaba bien el hosting virtual.
- **HTTP/1.1:** Introdujo la cabecera `Host` de forma obligatoria para permitir que múltiples dominios compartan una misma dirección IP.
- **HTTP/2:** La cabecera `Host` se sigue usando, pero se denomina `:authority`, y su transmisión es más eficiente gracias a la compresión y el uso de un formato binario.

b. En HTTP/1.1, ¿es correcto el siguiente requerimiento?
GET /index.php HTTP/1.1
User-Agent: curl/7.54.0
No, falta la cabecera host

c. ¿Cómo quedaria 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

:method: GET
:path: /index.php
:authority: www.info.unlp.edu.ar
:scheme: https
Conceptualmente seria eso pero no pq estaria en binario

Ejercicio de Parcial
curl -X ?? www.redes.unlp.edu.ar/??
> HEAD /metodos/ HTTP/??
> Host: www.redes.unlp.edu.ar
> User-Agent: curl/7.54.0

< HTTP/?? 200 OK
< Server: nginx/1.4.6 (Ubuntu)
< Date: Wed, 31 Jan 2018 22:22:22 GMT
< Last-Modified: Sat, 20 Jan 2018 13:02:41 GMT
< Content-Type: text/html; charset=UTF-8
< Connection: close

a. ¿Qué versión de HTTP podria estar utilizando el servidor?
1.1 porque usa el encabezado host y a su vez cierra la conexión (lo que seria default en 1.0)
b. ¿Qué método está utilizando? Dicho método, ¿retorna el recurso completo solicitado?
Head (-i, pide solo los encabezados)
c. ¿Cuál es el recurso solicitado?
Los encabezados?
d. ¿El método funcionó correctamente?
si
e. Si la solicitud hubiera llevado un encabezado que diga:
IF-Modified-Since: Sat, 20 Jan 2018 13:02:41 GMT
¿Cuál habria sido la respuesta del servidor web? ¿Qué habria hecho el
navegador en este caso?