Redes II

Resumen de Cuestionario

Emiliano Salvatori

Agosto 2019

1. Cuestionario clase nº 2

1.1. Definición de Socket

Los procesos envían mensajes desde la capa de aplicación a la capa de transporte mediante una API que se denomina Socket, el cual depende del tipo de protocolo que se vaya a emplear (TCP o UPD), se utiliza un socket u otro. El desarrollador de aplicaciones controla totalmente el proceso, pero a la hora de decidir cómo se enviará el mensaje a la capa de transporte, si será TCP o UDP, esta capa le deja configurar unos pocos parámetros, como el tamaño del buffer o del segmento.

Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un host o terminal en Internet) y un número de puerto (el cual identifica a la aplicación en particular).

1.2. Describa como es el funcionamiento de las Cookies y para qué sirve. ¿Riesgos asociados?

HTTP es un protocolo SIN memoria, por lo que si se quiere tener un control del historial de sus usuarios, hay que pedir que se registren y tener todas sus peticiones y transferencias en una base de datos y leerlas cada vez que ingresen a la página. El uso de COOKIES evita este procedimiento, almacenando en el navegador del lado del cliente, determinados datos.

Una cookie consiste en:

- Un par nombre-valor con datos almacenados.
- Una fecha de expiración a partir de la cual la cookie será inválida y eliminada.
- El dominio y el directorio del servidor a cual se enviará dicho dato.

El uso general de la cookie es:

- Cuando se quiere conectar por primera vez desde un navegador web a una página, por ejemplo Amazon, se envía un típico mensaje HTTP de solicitud.
- El Servidor de Amazon nos contestará el mensaje de petición enviado PERO con una solicitud de cookies que se guardará en nuestro navegador, de la siguiente forma: set-cookie: 1234. Por otro lado, el servidor de Amazon, generará en su base de dato la entrada 1234 para guardar información sobre mis conexiones.
- Cada vez que nos conectemos a Amazon, lo que hará nuestro navegador es enviarle la cookie almacenada con el número 1234 para que el servidor pueda buscar en su base datos toda la información nuestra y así poder tener disponibles todo el historial de navegación.

El problema que tiene el uso de las cookies es la privacidad:

- Se puede conocer información personal: nombre, email, etc.
- Pueden guardar información a lo largo de internet, como por ejemplo los anuncios que se muestran.
- Se puede hacer un seguimiento del historial de la navegación del usuario.
- Muchas empresas venden sus cookies a terceros, eso generó una controversia por lo que en la Unión Europea se decretaron leyes que atentan contra el desconocimiento de los navegantes en páginas que guardan cookies.

1.3. Explique y describa la Arquitectura Cliente-Servidor

En una arquitectura cliente-servidor siempre existe un host activo, denominado servidor, que da servicio a las solicitudes de muchos otros hosts, que son los clientes. Los hosts clientes pueden estar activos siempre o de forma intermitente. Un ejemplo clásico es la Web en la que un servidor web siempre activo sirve las solicitudes de los navegadores que se ejecutan en los hosts clientes. Cuando un servidor web recibe una solicitud de un objeto de un host cliente, responde enviándole el objeto solicitado.

Con la arquitectura cliente-servidor, los Host clientes no se comunican directamente entre sí; por ejemplo, en la aplicación web, dos navegadores no se comunican entre sí. Otra característica de la arquitectura cliente-servidor es que el servidor tiene una dirección fija y conocida, denominada dirección IP. Puesto que el servidor tiene una dirección fija y conocida, y siempre está activo, un cliente siempre puede contactar con él enviando un paquete a su dirección. Entre las aplicaciones más conocidas que utilizan la arquitectura cliente-servidor se encuentran las aplicaciones web, FTP, Telnet y de correo electrónico.

1.4. ¿Qué tipos de protocolos existen y que definen los protocolos de la capa de Aplicación?

Los protocolos de dominio público son los definidos por los RFC's: Por ejemplo HTTP o SMTP. Los protocolos de propietarios son por ejemplo aquellos que utiliza KaZaA o Skype.

Definen lo siguiente:

- Definen los tipos de mensajes intercambiados, por ejemplo: mensajes de requerimiento y respuesta.
- Definen la sintaxis de los tipos de mensajes: los campos en los mensajes que utilizan & son delimitados.
- Definen la semántica de los campos, por ejemplo el significado de la información en los campos.
- Por último definen las reglas para cuándo y cómo los procesos deben enviar y responder a mensajes.

1.5. Diferencias entre Web Cache y Proxy

A medida que las comunicaciones fueron haciéndose más veloces, también se implementaron mejoras en la navegación. La idea es que haya un servidor intermediario, gestionando las peticiones que los clientes hacen hacia los servidores, para que en caso de que haya una segunda solicitud a una misma página ya visitada, no tener que remitir la solicitud al servidor original, sino que sea el servidor Proxy quien conteste primero esa petición, cuya página estará guardad en su memoria caché (no hay que confundir, ya que no es la memoria caché del navegador)

El Servidor Proxy será por el que pasarán todas las peticiones de un cliente, si tiene lo solicitado, este se lo devolverá, en caso contrario, redirigirá la petición al Servidor original y cuando el objeto pase por el servidor Proxy, se guardará en él en caso de que se vuelva a consultar.

Por lo tanto el Servidor Proxy actuará como servidor para mi navegador web, como de cliente para otros servidores webs. Estos son instalados por el ISP para acelerar la navegación y evitar tráfico innecesario.

1.6. Explique las Arquitecturas P2P e Hibridas. Ejemplos de c/u

Arquitectura Peer To Peer

También denominada **arquitectura entre pares o P2P**

1. A diferencia de la Arquitectura Cliente-Servidor, todos los Host pueden actuar como clientes o servidores de forma simultánea.
2. Por ello, los servidores NO siempre estarán activos.
3. Los host clientes se podrán comunicar entre sí, pudiendo también cambiar su dirección IP.
4. Esta arquitectura es altamente escalable pero muy difíciles de gestionar.

Inconvenientes:

- Uno de los inconvenientes que tiene esta arquitectura es que las infraestructuras de red que han desplegado los ISP están orientadas a arquitecturas Cliente-Servidor donde es mayor el tráfico de bajada que el de subida (Aunque el uso de fibra óptica está re-dimensionando las líneas, haciendo un uso simétrico del ancho de banda).

- Otro de los problemas es la seguridad, dada la naturaleza extremadamente distribuida y abierta, es como poner puertas al campo.
- Con estas arquitecturas se han de crear políticas de incentivos para convencer a los usuarios y que ofrezcan voluntariamente ancho de banda, almacenamiento y computación.

Ejemplo de programas que utilizan la Arquitectura P2P:

- BitTorrent.Net
- G3 Torrent
- mlMac
- MXIE
- QTorrent
- Shareaza
- MuTorrent
- Vuze

Arquitecturas Híbridas

Hay arquitecturas p2p donde no todos los host son iguales, por lo que no son p2p puras. Son híbridas.

- Se tienen pares comunes (peers)
- Pero también se tienen super-peers por lo que se genera una jerarquía en la red.
- Se tiene un servidor centralizado de login, pero el índice que asigna los nombres de usuario a las direcciones IP está distribuido entre los super-peers, probablemente con una tabla hash distribuida.
- Tiene un protocolo privado por lo que no se sabe a ciencia cierta cómo trabaja.

Una de las aplicaciones más conocidas y utilizada que emplea la **Arquitectura P2P** es Skype.

1.7. Para que se usa el GET Condicional

Para actualizar el contenido de los servidores proxy se utiliza el comando GET condicional, mediante el cual se puede introducir en la cabecera, ciertas condiciones para que se traiga el objeto del servidor sólo en caso de que se cumplan. Por ejemplo la cláusula: *if-modified-since* es decir, si ha cambiado desde la última fecha que tiene constancia. Esta fecha se obtuvo en el proxy cuando le llegó el objeto por última vez en su campo last-modified. En este caso, si el servidor proxy tiene el objeto actualizado el servidor le devolverá un 304 not-modified evitando enviar un objeto de nuevo. En caso contrario, si el objeto ha cambiado, le devolverá un 200 OK con el objeto solicitado.

1.8. Compare los protocolos HTTP persistente con y sin pipeline. Como serían los tiempos si se quiere bajar con 1 archivo HTML con 5 objetos

Otra mejora es la denominada *pipelining* que consta de NO esperar a recibir el objeto completo para pedir el siguiente, sino hacer varias peticiones, simultáneas al servidor y que este entregue los objetos según los vaya enviando, o recibiendo las solicitudes de objetos. Esto puede llevar a problemas en algunos navegadores, porque algunos objetos pueden bloquear a otros objetos más importantes. Y es difícil implementarlo correctamente para que la mejora sea apreciable, esto hace que todos los navegadores tengan deshabilitado el *pipelining* por defecto. Se emplea un algoritmo mejor que se denomina *multiplexación*.

1.9. Que servicios proveen los Protocolos de transporte TCP y UDP

Servicio TCP:

- Es Orientado a la conexión establecer conexión (setup) requerido entre procesos cliente y servidor antes de transferencia
- Ofrece Transporte confiable entre proceso Transmisor (Tx) y Receptor (Rx)
- Tiene Control de flujo: Tx no sobrecargará al Rx
- Tiene Control de congestión: el Tx se frena cuando la red está sobrecargada
- No provee: garantías de retardo ni ancho de banda mínimos

Servicio UDP:

- Transferencia de datos no confiable entre proceso Tx y Rx.
- No provee: establecimiento conexión, confiabilidad, control de flujo, control de congestión, garantías de retardo o ancho de banda

¿Por qué existe UDP? UDP es más rápido para enviar rápido un paquete, se utiliza por eso. Porque no abre una conexión de forma previa.

1.10. Que tipos de servicios requieren las aplicaciones de la capa de transporte

La capa de Aplicación necesita:

- **Confiabilidad en la entrega (sin pérdida de datos)**: Algunas aplicaciones requieren transferencia 100 % confiable. Algunas otras si pueden tolerar pérdida (video y audio)
- **Retardo**: Algunas aplicaciones (como los video juegos) requieren bajo retardo para ser *efectivas*.
- **Ancho de Banda**: Algunas aplicaciones requieren una cantidad mínima de ancho de banda para ser *efectivas*. Y otras aplicaciones (*Aplicaciones elásticas*) hacen uso de la banda ancha que obtengan.

1.11. Compare los protocolos HTTP persistente y HTTP no persistente. Como serían los tiempos si se quiere bajar con 1 archivo HTML con 5 objetos

Los tipos de conexiones HTTP pueden ser:

- **Las conexiones no persistentes**: son aquellas donde se cierra la conexión TCP luego de haber transferido cada objeto solicitado.
- **Las conexiones persistentes**: son aquellas en las que se pueden enviar varios objetos antes de cerrar la conexión.

Para evitar este problema al cerrar la conexión luego de enviado el archivo (y con esto generar empobrecimiento en la transferencia de múltiples objetos), se utilizan conexiones persistentes.

Lo que se hace es mantener una conexión abierta entre cliente y servidor aún luego de haber transferido el objeto solicitado por el primero, así en caso de que se soliciten más objetos, se envián por la misma conexión abierta que se tiene. Por lo que el cliente emplea un solo RTT (Round Trip Time) para abrir la conexión y luego un RTT por cada objeto que sea transferido.

Round trip time: tiempo que tarda un paquete en ir del cliente al servidor y volver, sin tener en cuenta su tamaño. En una conexión TCP normal, el cliente envia una solicitud de conexión, y cuando el servidor le contesta que si, estableciendo una conexión, ese es el tiempo RTT.

En el caso práctico se tiene una página HTML con 5 objetos dentro. En las conexiones **HTTP NO persistentes** se tendrán 2 RTT por cada objeto solicitado, en este caso se solicitan 6 objetos (la página HTML con sus sucesivos cinco objetos), por lo que se tienen 12 RTT totales.

En cambio, en las conexiones **HTTP persistentes**, se utiliza 1 RTT para establecer la conexión con el Servidor y luego 1 RTT por cada uno de los objetos que se solicitan; por lo que en caso de pedir 6 objetos (la página HTML con sus sucesivos 5 objetos) se tendrán 7 RTT totales.

2. Questionario Clase nº 3