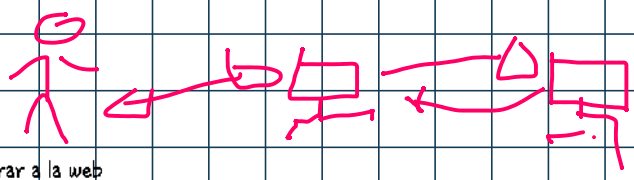
Clase 2- capa de aplicación

miércoles, 28 de agosto de 2024 19:01



Funciones:

- Da servicios a usuarios y a las aplicaciones
- Hay comunicación persona, aplicación(UI) y máquina-máquina(M2M)
- Aplicaciones que usan la red son de esta capa, protocolos que usan las apps también
- Hay apps que no son de red y tienen que trabajar con otras apps-servicios para entrar a la web

Capa de red-----> capa de app, capa de presentación y capa de sesión del OSI

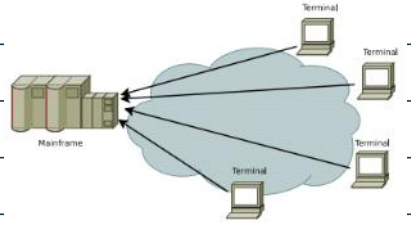
- Maneja DATOS: que van por ejemplo de servidor de usario a servidor web (ej protocolo para mandar mail)

Modelos de comunicación de aplicaciones:

- Modelo mainframe
- Modelo cliente/servidor
- Modelo P2P
- Modelo híbrido

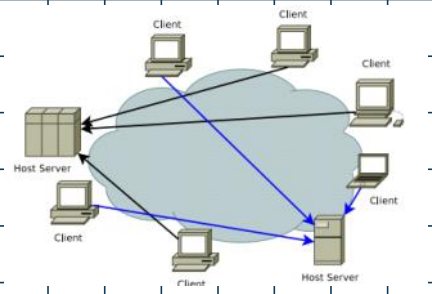
Modelo de mainframe

- Cliente es tonto, solo corre comunicación e interfaz con el usuario
- Servidor es quien procesa



Modelo cliente-servidor

- Carga compartida
- Cliente pone el procesamiento de la interfaz, servidor el resto
- Servidor corre el servicio esperando de forma pasiva la conexión
- Modelo 1 a n, m a n (1 sv, muchas requests)



Peer-to-peer

- Carga completamente compartida y distribuida
- Los participantes pueden ser cliente, sv, o los dos.
- Puede escalar en rendimiento, no en aministracion

HTTP Y WEB CACHE

Elementos web:

- URI:lo que se referencia
- URL:donde está lo que se referencia
- Formato de URL: protocol://[user:pass@]host:[port]/[path].
- Modelo cliente sevidor, request/response sin estados, no se establece una sesión (responde lo que le pidieron y listo).
- Peticiones llegan desde los clientes hacia el sv. Si yo me conecto al 80 o el 443 y el está andando, me va a responder
- Le manda datos com texto asci

Pasos para obtener un documento:

Versíon de HTTP 1.0 estándar [RFC-1945].
Define formato, proceso basado en HTTP 0.9:

- Se debe especificar la versíon en el requerimiento del cliente.

- Para los Request, define diferentes métodos HTTP.
- Define códigos de respuesta
- Admite repertorio de caracteres, además del ASCII, como: ISO-8859-1, UTF-8, etc.
- Admite MIME (No solo sirve para descargar HTML e imágenes).
- Por default NO utiliza conexiones persistentes. (por defecto pide, obtiene y cierra, no queda abierta la conexión)

Servidor responde(a diferencia de http 0.9). Da codigo de error

GET: obtener el documento requerido. Puede enviar información, pero no demasiada. Es enviada en la URL. Formato `?var1=val1&var2=val2....` Limitación de tamaño de URL por parte de las implementaciones. NO espera recibir datos en body.

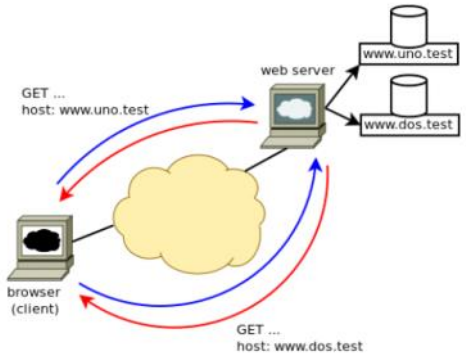
HEAD: idéntico a GET, pero sólo requiere la meta información del documento, por ejemplo, su tamaño. Usado por clientes con caché.

POST: hace un requerimiento de un documento, pero también envía información en el Body. Generalmente, usado en el fill-in de un formulario HTML(FORM). Puede enviar mucha más información que un GET.

PUT: usado para reemplazar un documento en el servidor. En general, deshabilitada. Utilizado, por ejemplo, por protocolos montados sobre HTTP, como WebDAV [WDV].

DELETE: usado para borrar un documento en el servidor. En general, deshabilitada. También, puede ser utilizada por WebDAV.

LINK, UNLINK: establecen/des-establecen relaciones entre documentos.



Como sabe el sv con qué le tiene que contestar? No necesito un servidor por sitio. Viene en la cabecera de host la dirección a la que quiere ir. Si no le digo, va solito al 1ro que tenga. EJ paginas del siu con = ip, cabecera las disingue.

Autentificación http

- Tipo cundo me meto a la config del router, no es un login que hizo alguien bonito y demas sino que es de una de Http
- Requiere user y password y da error 401 si no es

A partir de http1 puedo decirle que sea no persistente la conexión
Connection: keep alive

HTTP 1.1
Pipelling:me permite en conexiones persostentes pedir cosas sin esperar que me responda lo que le pedí antes. Se usa para acelerar la velocidad. En la v1 tiene que tener =orden lo que mando a lo que recibo pero en la 2 no. pROBLEMA
Trace: para tratar de ver si alguien interceptó la conexión y metió mano. Cliente puede ver que lo que recibe es lo que manda. Se usa para testing.

Proxy: para que mi compu en lugar de conectarse de una a un sv se conecte desde otro equipo
Es como un intrmediario. Se usa por ejemplo para filtrar cosas de la web y banearte algunas paginas

Redirect:
Temporal(302) permanente(301). ??????????

Server side scripts:

- Dinamismo para crear apps.
- Ejecuta del lado del servidor
- Lenguajes de scripting más flexibles y seguros: PHP, ASP, JSP

Client side cripts:
- Ejecuta de lado del cliente, en el navegador
- Otros lenguajes JScript, VBScript.

Cookies:
- Para establecer estado, relación entre la persona y la página
- Cuando sitio web quiere setear requerimiento le pide petición

HTTPS(HTTP sobre TLS/SSL)
- Usa port 443
- Canal entre cliente y servidor que nadie en el medio de la comunicación puede interceptar
- Canal es cifrado

Cache
- Busca ahorrar recursos de red:mejore respuestas y demás servidores se revientan si tienen mucha gente preguntando. Netflix por ejemplo guarda contenido en servidores más chiquitos locales para no tener que hacer 10000 request al big sv.
- Cache del navegador(del cliente)

Protocolo de texto --> ascii, utf8

Ssl/tls para proteger la comunicación se establece una clave simétrica

Conexiones persistentes
Pipelining
Navegadores configuran varias conexiones (chrome 6) con el servidor para zafar del problema del pipeling.
Muchos recursos