

Redes II

Parciales Resueltos

Emiliano Salvatori

Agosto 2019

1. Primer Parcial 2016 - Primera Fecha

1.1. Describa cómo es el funcionamiento de las Cookies y para qué sirven. ¿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:

1. Un par nombre-valor con datos almacenados.
2. Una fecha de expiración a partir de la cual la cookie será inválida y eliminada.
3. 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 uso de las cookies en los servidores es muy variada:

- Para gestionar sesiones y facilitar el login para que el usuario no necesite ingresar usuario y contraseña cada vez que se conecte.
- Mantener carrito de la compra.
- Ayudan a la personalización del sitio, almacenando como el color, idioma o tamaño de letra favorito.

Con respecto a la **Privacidad** los cookies permiten:

- 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.

1.2. Compare los protocolos HTTP persistente y HTTP no persistentes. ¿Cómo serían los tiempos si se quiere bajar con un archivo THML con cinco objetos?

La web se basa por tanto en el protocolo HTTP que sigue un modelo cliente-servidor. Donde el cliente es el navegador y el servidor es el servidor web (como por ejemplo Apache). El protocolo que se utiliza es el **TCP** por lo que el cliente antes de realizar la transferencia de datos, debe establecer una conexión TCP, creando el socket con el servidor web, que tiene su socket esperando en el puerto 80. Acepta la conexión, y una vez establecida la conexión se intercambian los mensajes http y finalmente se cierra la conexión.

HTTP es un **protocolo sin memoria**, por lo que el servidor no mantiene un historial de las peticiones realizadas del mismo cliente.

Tipos de conexiones HTTP:

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

Por otro lado se tiene lo que se denomina **Round trip time** o (RTT): 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 envía una solicitud de conexión, y cuando el servidor le contesta que sí, estableciendo una conexión, ese es el tiempo RTT.

Se necesitarán 2 RTT por cada objeto o página HTTP que se requiera del servidor:

1. Un tiempo entre que se envía una petición de conexión y el servidor devuelve que si
2. Otro tiempo en enviar una petición del objeto que se requiere y el servidor lo envía. Luego de ello como es una **conexión NO persistente** se cierra la conexión, por lo que si se quiere pedir OTRO objeto, se deberá de hacer todos los pasos de nuevo.

CUIDADO: Otra forma de encausar esto, sería pedir por ejemplo 3 conexiones con el servidor, pero esto multiplicaría su rendimiento, porque si por cada objeto solicitado se tienen 2 RTT, por 3 conexiones se tendrían 6 RTT.

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 envían por la misma conexión abierta que se tiene. Por lo que el cliente emplea un solo RTT para abrir la conexión y luego un RTT por cada objeto que sea transferido.

Lo que se emplea en éste último tiempo es lo que se denomina **HTTP persistente concurrente**: donde el cliente abre de forma concurrente (al mismo tiempo) varias conexiones con el MISMO servidor para realizar la transferencia de objetos de forma simultánea y de forma más ágil.

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**.

1.3. Explique la función de al menos 4 tipos de registros del servidor de nombres (DNS)

Los registros DNS no son más que pues cuádruplas que se denominan **Resource Records** (RR), del nombre, el valor, luego el tipo de registro que es y un tiempo de vida.

Formato RR(Nombre, valor, tipo de registro y TTL (tiempo de vida))

Los registros más típicos son el tipo A que es el que hace una correspondencia entre el nombre de la máquina y la IP. En el nombre tendrá el nombre de la máquina y y en el valor tendrá la ip de la máquina. Estos son los registros que más se consultan.

También se puede solicitar un registro tipo **Name Server** (NS) que el nombre se tendrá el dominio, por ejemplo: **google.com** y el valor se tendrá el nombre de un DNS autoritativo para ese dominio, es decir, devolverá un nombre de un servidor DNS que es autoritativo para ese dominio, lo que significa que devolverá un nombre de un servidor DNS, que es autoritativo para ese dominio, y que tiene los registros que corresponden a las máquinas de ese dominio. Por lo tanto, si se quiere saber qué servidores DNS tienen el dominio google.com, se puede realizar una consulta tipo NS.

También hay consultas tipo **Canonical Name** (CNAME), que son consultas o registros (se realiza una consulta y se devuelve un registro de este tipo) en el que este registro en el nombre va a figurar un alias para un nombre canónico que se le indica en el valor por ejemplo, el nombre canónico es la máquina real, digamos `www.ibm.com`, el cual es un servidor que se llama con ese nombre, este es `serveeast.backup2.ibm.com`, por lo que cabe resalta que NO es `www`, ya que eso es un alias realmente la máquina que está sirviendo será la otra (`serveeast.backup2.ibm.com`), es por ello que si se hace la consulta CNAME en otro sitio, a lo mejor hay otro servidor que está más cercano y le va a contestar con otro nombre, porque es otra máquina.

Por último, tenemos registros tipo **Mail Exchange** (MX) en los cuales se va a tener en el valor el nombre canónico del servidor de correo que se corresponde a ese dominio que se ingresará en el nombre. Por lo tanto me va a dar información sobre el servidor de correo que se corresponde al dominio que le estoy pidiendo.

A: hostname a una IP. Se utiliza para darle un nombre a una máquina, nombre canónico se denomina. Si el servidor tiene 30 páginas hospedadas, se le hacen un CNAME, que apunten al primero.

Se utiliza la A para no tener que cambiar IP, cada vez que hay que cambiar la IP o cambiar otra cosa, hay que hacer 30 cambios de registros, en caso con CNAME, lo que permite es que haya sólo un cambio.

MX: Nos dicen qué nombre de servidor manejará determinado correo.

1.4. ¿Cómo y por qué DNS tiene una base jerárquica y Distribuida? Explique e identifique sus partes

En internet las máquinas tienen direcciones IP, pero para evitar recordar estos números se emplean nombres de dominio. El DNS o el *Domain Name System*, es el sistema que permite almacenar las correspondencias entre direcciones IP y nombres de dominio. También se emplea el nombre DNS para referirse al protocolo que se emplea para consultar esa base de datos distribuida. El servidor DNS suele ser máquinas de tipo UNIX que ejecutan BIND sobre UDP utilizando el puerto nº 53.

Todo esto lo hace de una forma transparente al usuario, además se hace de forma descentralizada. Esto es así ya que:

- Un único DNS sería un único punto de fallo.
- Con un gran volumen de tráfico
- Gran distancia a la Base de datos centralizada, al único nodo.
- Provocaría grandes problemas de mantenimiento.
- También contribuiría a tener problemas de escalado .

Por lo tanto se creó una base de datos, distribuida y jerárquica en la que los servidores se distribuyen principalmente en tres niveles:

- Raíz
- Top Level Domain
- Autorizativos.

1.4.1. Ejemplo

Por ejemplo si queremos solicitar la dirección `www.unaj.edu.ar` lo que se hace es solicitarla al servidor Raíz, este devuelve la dirección del servidor Top Level Domain “.ar”, a este le pedimos la IP y nos devuelve la del servidor autorizativo `unaj.edu.ar` y a este servidor se la pedimos nuevamente, y nos devuelve la IP de la máquina `www.unaj.edu.ar`

1.4.2. Jerarquía

- Los DNS Raíz son 13 y están replicados y configurados en clusters por todo el mundo.
- Los DNS TLD : Son los responsables de los dominios de nivel superior y de países: .com, .org, .ar, etc.
- Los DNS autorizativos son los que tendrán los registros DNS de los host que están accesibles al público, como los servidores webs o el correo.

- Por último encontramos a los servidores DNS locales o servidores de nombres predeterminados, que son implementados por cada ISP y NO pertenecen a la jerarquía antes descripta, son a los que primero pedimos a una traducción IP y son los encargados de buscarla, por la jerarquía DNS. Tienen una caché por lo que de cada búsqueda que hacen, se quedan una copia, por lo que si se vuelve a pedir, no tienen que volver a buscarla.

Los registros DNS almacenados en estas caches expiran al cabo de un tiempo y hay mecanismos de actualización y notificación para mantener la información al día.

1.5. ¿Cuáles son las fuentes de retardo de un paquete? Explíquelas (Si corresponde proporcione las fórmulas asociadas a cada retardo)

Existen cuatro motivos dentro de la conmutación de paquetes por lo que se pueden retrasar los paquetes:

1. **Retardo de Procesamiento en el Nodo:** La principal fuente de retardo es la que se produce cuando llega un paquete desde un ordenador a un nodo. El nodo debe procesar ese paquete, es decir, comprobar si tiene errores y determinar por cuál enlace debe salir ese paquete. Es el denominado Retardo de procesamiento en el nodo". Este retardo suele ser del orden de los microsegundos.
2. **Retardo de Cola:** El paquete se dirige al enlace de salida (la puerta de salida del nodo) donde se encuentra que ya hay un paquete del host A esperando a salir por él, por lo que tendrá que esperar, para ser transmitido. Ese tiempo de espera en la cola de salida del nodo se conoce como retardo de cola". Es el tiempo de espera para transmitir el paquete por el enlace de salida. Y dependerá del nivel de congestión que tenga el nodo en el momento de transmitir, variando el tiempo de unos microsegundos o milisegundos si hay mucho tráfico en espera.
3. **Retardo de transmisión:** Es el tiempo que tarda el nodo en meter el paquete por el enlace. Este tiempo dependerá de la longitud del paquete (L) y de la capacidad del ancho de banda del enlace (R). El cual se puede calcular fácilmente realizando la división L/R . Suele ser del orden de micro o milisegundos.
4. **Retardo de propagación:** Es el retardo que se produce al viajar un paquete del nodo de salida al nodo de llegada a través del enlace. Dependerá de este retardo de la longitud del enlace (S) y de la velocidad del medio de propagación (V), que suele ser del orden de los microsegundos, aunque a grandes distancias puede llegar a los milisegundos.

El retardo más interesante para su estudio es el **retardo de cola**: Porque depende de la congestión y de la naturaleza del tráfico que haya a la hora de transmitir el paquete, si es en ráfagas o periódicamente. Variando así de un paquete a otro. En el nodo se puede calcular al tráfico que existe entre haciendo la longitud del paquete * por la tasa de paquetes de llegada, dividido el tráfico de salida. Es la relación que existe entre el tráfico que llega y el que sale.

Los paquetes se pierden porque la cola de salida en el enlace tienen un límite, que de sobrepasarse los paquetes comienzan a perderse. Cuando esto sucede, el protocolo puede indicar al origen que el paquete se perdió y está en el origen enviarlo de nuevo o no (capa que intercede aquí es la de transporte).

1.6. Describa el protocolo POP e IMAP (aplicación que lo utiliza, para qué lo utiliza, cuál es el número de puerto asociado, problemas de seguridad y cómo se solucionan)

Lo anteriormente hablado es para el envío de correo entre Servidores. ¿Cómo hace para acceder desde un Agente a un correo? Para ello existe los protocolos de acceso al correo. Entre ellos se encuentran:

- **POP3:** Post Office Protocol versión nº 3 [RFC 1939] el cual determina una autorización entre Agente y Servidor, la descarga y actualización.
- **IMAP:** Internet Mail Access Protocol [RFC 3501] el cual tiene muchísima más funcionalidad pero es más complejo; también permite la manipulación de mensajes almacenados en el servidor.
- **HTTP:** También puede ser utilizado como los que usan los ya conocidos: Gmail, Hotmail, Yahoo! Mail, etc.

IMAP: Es mucho más potente que POP3, ya que permite gestionar el buzón de correo creando carpetas para poder organizar los correos en el propio servidor por lo que tiene memoria de estado entre sesiones.

POP3: Nos permitirá acreditarnos a nuestro Servidor para poder acceder a nuestro buzón, y descargar así los correos que tengamos almacenados en él. Por defecto, POP3 descarga los correos al Agente de Mail y los borra del Servidor (aunque esto se puede configurar para que los deje almacenados).

- Al conectarse, lo que hace primero es acreditarse para poder acceder al buzón.
- Luego, pide un listado de los correos y los va descargando y marcando para ser borrados.
- Al final, se envía la orden para que el Servidor borre los correos que han sido marcados para borrar.

1.7. ¿Qué tipo de servicios de la capa de transporte necesita una aplicación? Explique cada una de las ventajas y desventajas

Así como hay dos tipos de servicio de red, orientado y no orientado a la conexión, hay dos tipos de servicio de transporte. El servicio de transporte orientado a la conexión es parecido en muchos sentidos al servicio de red orientado a la conexión. En ambos casos, las conexiones tienen tres fases: establecimiento, transferencia de datos y liberación (o terminación). El direccionamiento y el control de flujo también son semejantes en ambas capas. Además, el servicio de transporte no orientado a la conexión es muy parecido al servicio de red no orientado a la conexión. Por lo tanto el tipo de servicio en la capa de transporte son dos:

- Servicio de transporte orientado a la conexión (cuyo protocolo puede ser por ejemplo TCP).
- Servicio de transporte NO orientado a la conexión (cuyo protocolo puede ser por ejemplo UDP).

La pregunta obvia es: ¿si el servicio de la capa de transporte es tan parecido al de la capa de red, por qué hay dos capas diferentes? ¿Por qué no es suficiente una sola capa? La respuesta es sutil, pero crucial. El código de transporte se ejecuta por completo en las máquinas de los usuarios, pero la capa de red, por lo general, se ejecuta en los enrutadores, los cuales son operados por la empresa portadora (por lo menos en el caso de una red de área amplia). ¿Qué sucede si la capa de red ofrece un servicio poco confiable? ¿Qué tal si esa capa pierde paquetes con frecuencia? ¿Qué ocurre si los enrutadores se caen de cuando en cuando? Problemas.

Los usuarios no tienen control sobre la capa de red, por lo que no pueden resolver los problemas de un mal servicio usando mejores enrutadores o incrementando el manejo de errores en la capa de enlace de datos. La única posibilidad es poner encima de la capa de red otra capa que mejore la calidad del servicio.

1.8. ¿Qué tipo de registro se solicitó y cuál fue la respuesta?

Por lo que se puede visualizar, se hizo una consulta DNS para resolver la dirección `www.ole.clarin.com`.

Se envía una consulta y se obtienen dos respuestas. Por una parte se devuelve una consulta de tipo **Canonical Name** (CNAME), donde el alias es `www.ole.clarin.com` y en la segunda respuesta, se obtiene la dirección asociada a ese alias que termina siendo: `200.42.93.137`

2. Primer Parcial 2016 - Segunda fecha

2.1. Explique DHT Circular y DHT Circular con atajos. Abandono de peers

Las tablas de hash distribuidas, conocidas por las siglas DHT, (del inglés, Distributed Hash Tables) son un tipo de tablas de hash, almacenan pares de clave-valor y permiten consultar el valor asociado a una clave, en las que los datos se almacenan de forma distribuida en una serie de nodos (sistemas distribuidos) y proveen un servicio eficiente de búsqueda que permite encontrar el valor asociado a una clave. Para esto último usan un sistema de enrutado que permite encontrar de forma eficiente el nodo en el cual está almacenada la información que se necesita.

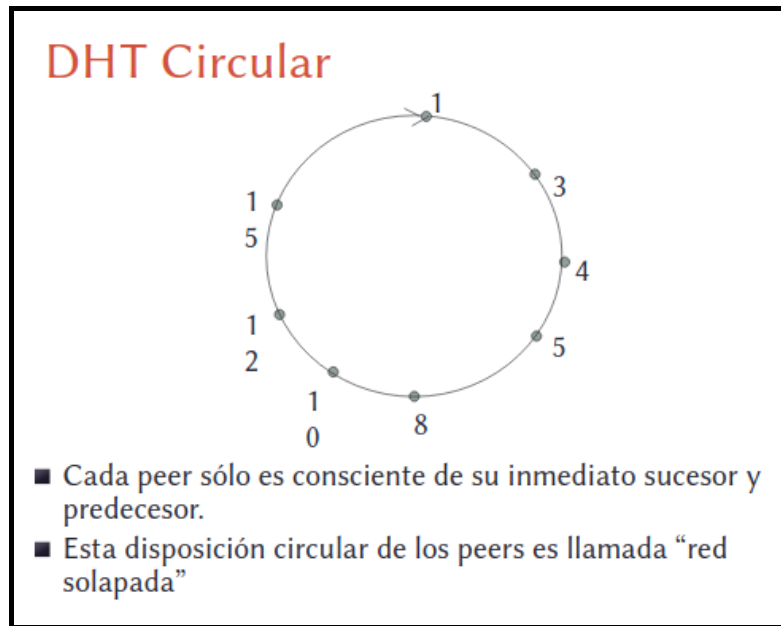
La responsabilidad de mantener el mapeo de las claves a los valores está distribuida entre los nodos, de forma que un cambio en el conjunto de participantes causa una cantidad mínima de interrupción. Esto permite que las DHTs puedan escalar a cantidades de nodos extremadamente grandes, y que puedan manejar constantes errores, llegadas y caídas de nodos.

2.1.1. DHT Circular

Cada par sólo es consciente de la existencia de su inmediato sucesor; por ejemplo, el par 5 conoce la dirección IP y el identificador del par 8, pero no tiene por qué saber nada acerca de los restantes pares que pueda haber en la DHT. Esta disposición circular de los pares es un caso especial de **red solapada**. En una red solapada, los pares forman una red lógica abstracta que reside por encima de la red de computadoras ¿subyacente? formada por los enlaces físicos, los routers y los hosts. Los enlaces de una red solapada no son enlaces físicos, simplemente son uniones virtuales entre parejas de pares. Normalmente, un enlace solapado utiliza muchos enlaces físicos y routers de la red subyacente.

La DHT circular proporciona una solución muy elegante para reducir la cantidad de información solapada que debe gestionar cada par. En resumen, cada par sólo es consciente de la existencia de otros dos pares, su

inmediato sucesor y su inmediato predecesor. Pero esta solución introduce un nuevo problema. Aunque cada par únicamente es consciente de sus dos pares vecinos, para encontrar al nodo responsable de una clave (en el caso peor), todos los N nodos de la DHT tendrán que reenviar un mensaje siguiendo el contorno del círculo, luego, como media, se enviarán $N/2$ mensajes

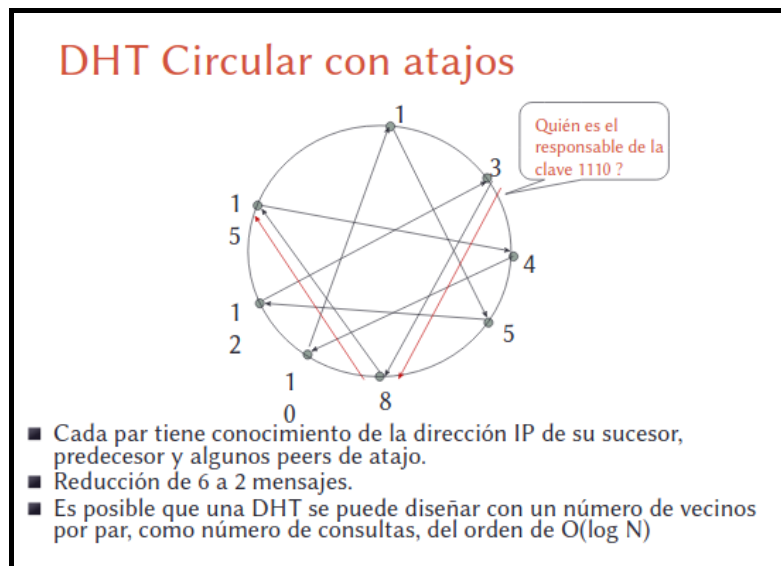


2.1.2. DHT Circular con Atajos

Por tanto, al diseñar una DHT hay que llegar a un compromiso entre el número de vecinos del que cada par puede llevar la cuenta y el número de mensajes que la DHT necesita enviar para resolver una única consulta. Por un lado, si cada par lleva la cuenta de todos los demás pares (solapamiento en malla), entonces sólo habrá que enviar un mensaje por consulta, pero cada par tiene que controlar a N pares. Por otro lado, con una DHT circular, cada par sólo es consciente de la existencia de dos pares, pero se envían $N/2$ mensajes como media por cada consulta. Afortunadamente, podemos ajustar nuestro diseño de la DHT de manera que el número de vecinos por par, así como el número de mensajes por consulta se mantenga en un tamaño aceptable. Una posible mejora consistiría en utilizar la red solapada circular como base y añadir “atajos” de modo que cada par no sólo tenga controlado a su inmediato sucesor, sino también a un número relativamente pequeño de pares de “atajo” dispersos alrededor del círculo.

Estos atajos se emplean para agilizar el enrutamiento de los mensajes de consulta. Específicamente, cuando un par recibe un mensaje para consultar una clave, reenvía el mensaje al vecino (al sucesor o a uno de los vecinos atajo) más próximo a la clave. Evidentemente, los atajos pueden reducir de forma significativa el número de mensajes utilizado para procesar una consulta.

En consecuencia, ahora la pregunta que debemos planteamos es: “Cuántos vecinos atajo debe tener cada par y qué pares deberían ser esos vecinos atajo?”. Se ha demostrado que una DHT se puede diseñar de modo que tanto el número de vecinos por par, así como el número de mensajes por consulta, sea igual a $O(\log N)$, donde N es el número de pares. Este tipo de diseño proporciona un compromiso satisfactorio entre las soluciones extremas de emplear topologías solapadas de red en malla y redes circulares.



2.2. Explique las diferencias entre servidor web cache y proxy. Explique cómo funciona cada uno.

2.2.1. Servidor Web Proxy

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 guardada en su memoria caché, (¡No es la 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.

Ventajas:

- Navegación más rápida.
- Reducción del tráfico.
- Control de tráfico.
- Navegación anónima.
- Navegación segura.
- Navegación como si se estuviera navegando desde un país determinado.

¿Cómo actúa si se quiere consultar la página www.unaj.edu.ar ? De la siguiente forma:

1. El navegador establece una conexión TCP con la caché web y envía una solicitud HTTP para el objeto a la caché web.
2. La caché web comprueba si tiene una copia del objeto almacenada localmente. Si la tiene, la caché web disuelve el objeto dentro de un mensaje de respuesta HTTP al navegador del cliente.
3. Si la caché web no tiene el objeto, abre una conexión TCP con el servidor de origen, es decir, con www.unaj.edu.ar. La caché web envía entonces una solicitud HTTP para obtener el objeto a través de la conexión TCP caché-servidor. Después de recibir esta solicitud, el servidor de origen envía el objeto dentro de un mensaje de respuesta HTTP a la caché web.
4. Cuando la caché web recibe el objeto, almacena una copia en su dispositivo de almacenamiento local y envía una copia, dentro de un mensaje de respuesta HTTP, al navegador del cliente (a través de la conexión TCP existente entre el navegador del cliente y la caché web).

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 casode 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.

2.3. Tipos de Arquitectura de Aplicación: breve explicación de cuántas y cuáles son, ¿cómo funcionan? Citar cuatro aplicaciones que utilicen arquitecturas Híbridas

Existen dos tipos de paradigmas arquitectónicos predominantes en las aplicaciones de red moderna:

1. La arquitectura **Cliente-Servidor**.
2. La arquitectura **P2P**.
3. También existen las **Híbridas** cuando se emplean rasgos de una y de la otra.
4. **Cloud**: es como el cliente servidor, pero con el servidor en la nube.

2.3.1. 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 de tipo:

- Web.
- FTP.
- Telnet.
- Correo Electrónico.

Las principales características de un **servidor** en una arquitectura Cliente-Servidor, son:

- Estará siempre activo, normalmente esperando conexiones de los clientes.
- Tendrá una IP fija, permanente y conocida a la que se podrán dirigir los clientes para solicitar los servicios que ofrezca ese servidor.
- El principal problema que tienen los servidores es el escalado, según la demanda que se tenga de sus servicios.

Las principales características de un **cliente** en una arquitectura Cliente-Servidor, son:

- Se conectan de forma intermitente con el servidor.
- Pueden cambiar su dirección IP usando direccionamiento dinámico.
- Los clientes NO se comunican directamente entre si.

2.3.2. Arquitectura P2P

En una arquitectura P2P existe una mínima (o ninguna) dependencia de una infraestructura de servidores siempre activos. En su lugar, la aplicación explota la comunicación directa entre parejas de hosts conectados de forma intermitente, conocidos como peers (pares). Los pares no son propiedad del proveedor del servicio, sino que son las computadoras de escritorio y portátiles controlados por usuarios, encontrándose la mayoría de los pares en domicilios, universidades y oficinas. Puesto que los pares se comunican sin pasar por un servidor dedicado, la arquitectura se denomina arquitectura peer-to-peer (P2P).

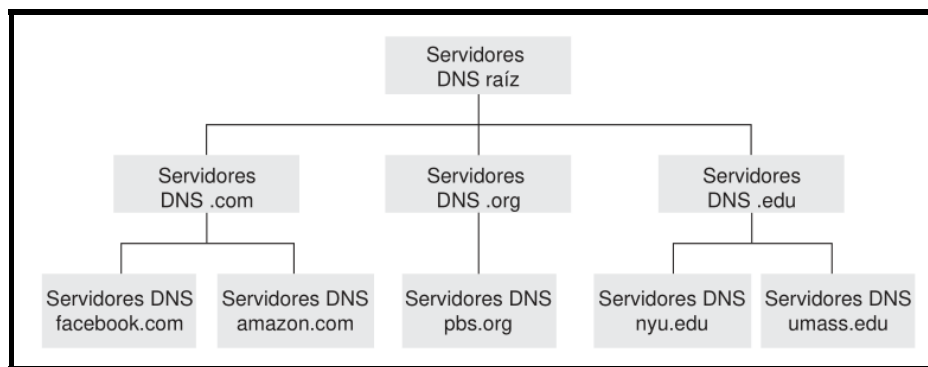
Muchas de las aplicaciones actuales más populares y con un elevado nivel de tráfico están basadas en arquitecturas P2P. Entre estas aplicaciones se incluyen la distribución de archivos (por ejemplo, BitTorrent), la compartición de archivos (como eMule y LimeWire), la telefonía por Internet (como Skype) e IPTV (como PPLive).

Las principales características de la arquitectura **P2P** son:

- Los Host pueden actuar como clientes o servidores de forma simultánea.
- Por lo anterior, los servidores NO siempre estarán activos.
- Los host clientes se podrán comunicar entre sí, pudiendo también cambiar su dirección IP.
- Esta arquitectura es altamente escalable pero muy difíciles de gestionar.
- 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.

2.4. ¿Qué diferencia una consulta DNS recursiva de una iterativa? Dé ejemplos

Parte de la jerarquía de los servidores DNS:



2.4.1. Servidores DNS raíz

Existen unos 400 servidores de nombres raíz distribuidos por todo el mundo. Trece organizaciones diferentes gestionan estos servidores de nombres raíz. Los servidores de nombres raíz proporcionan las direcciones IP de los servidores TLD

2.4.2. Servidores de Dominio de Nivel Superior (TLD)

Para todos los dominios de nivel superior, como son **com, org, net, edu, gov** y todos los dominios de nivel superior correspondientes a los distintos países, como por ejemplo, **uk, fr, ca, jp** existe un servidor TLD (o agrupación de servidores). La empresa Verisign Global Registry Services mantiene los servidores TLD para el dominio de nivel superior **com** y la empresa Educause mantiene los servidores TLD para el dominio de nivel superior **edu**. La infraestructura de red que da soporte a un TLD puede ser muy grande y compleja. Los servidores TLD proporcionan las direcciones IP para los servidores DNS autoritativos.

2.4.3. Servidores DNS autoritativos

Todas las organizaciones que tienen hosts accesibles públicamente (como son los servidores web y los servidores de correo) a través de Internet deben proporcionar registros DNS accesibles públicamente que establezcan la correspondencia entre los nombres de dichos hosts y sus direcciones IP. Un servidor DNS autoritativo de una organización alberga estos registros DNS. Una organización puede elegir implementar su propio servidor DNS autoritativo para almacenar estos registros; alternatively, la organización puede pagar por tener esos registros almacenados en un servidor DNS autoritativo de algún proveedor de servicios. La mayoría de las universidades y de las empresas de gran tamaño implementan y mantienen sus propios servidores DNS autoritativos principal y secundario (backup).

2.4.4. Consulta DNS Recursiva e Iterativa

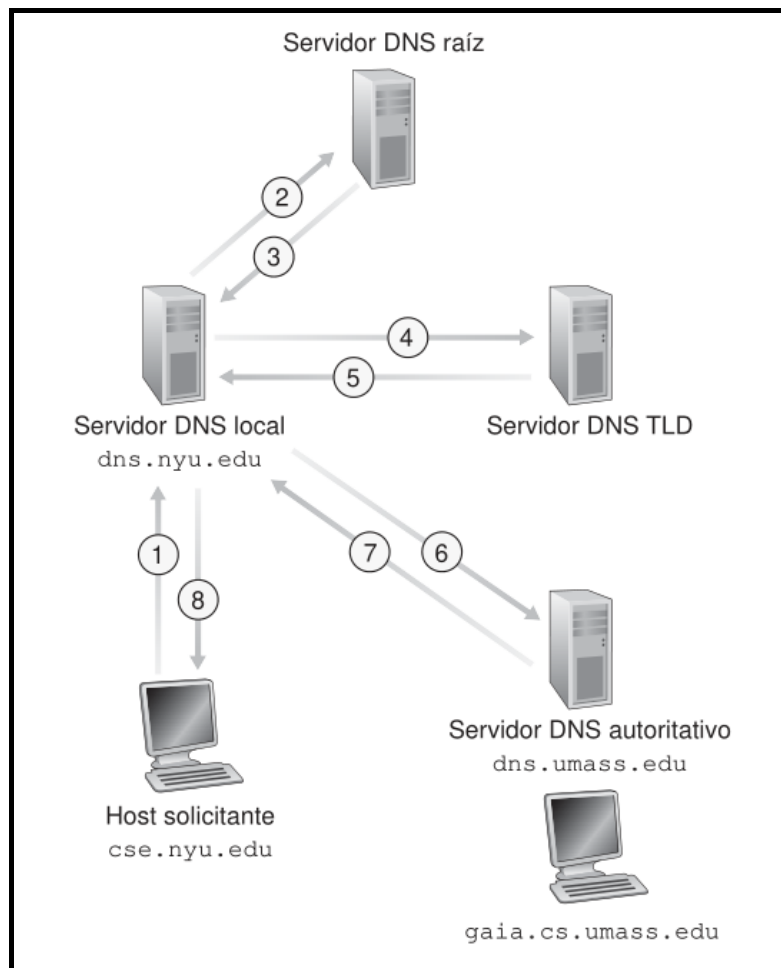
Una **consulta recursiva** obliga a un servidor DNS para responder a una solicitud con un error o una respuesta de éxito. Los clientes DNS (resoluciones) normalmente realizan consultas recursivas. Con una consulta recursiva, el servidor DNS debe ponerse en contacto con otros servidores DNS que necesita para resolver la solicitud. Cuando reciba una respuesta correcta de DNS (los otros servidores), a continuación, envía una respuesta al cliente DNS. La consulta recursiva es el tipo de consulta típica usado por una resolución de consultas a un servidor DNS y por un servidor DNS consulta su reenviador, que es otro servidor DNS configurado para procesar solicitudes reenviadas a él.

Cuando un servidor DNS procesa una consulta recursiva y la consulta no se puede resolver desde datos locales (archivos de zona local o caché de consultas anteriores), la consulta recursiva debe trasladarse a un servidor DNS raíz. Cada aplicación basada en estándares de DNS incluye un archivo de caché o sugerencias del servidor raíz que contiene entradas para los servidores DNS de raíz de los dominios de Internet. (Si el servidor DNS está configurado con un reenviador, el reenviador se utiliza antes de que se utilice un servidor raíz.)

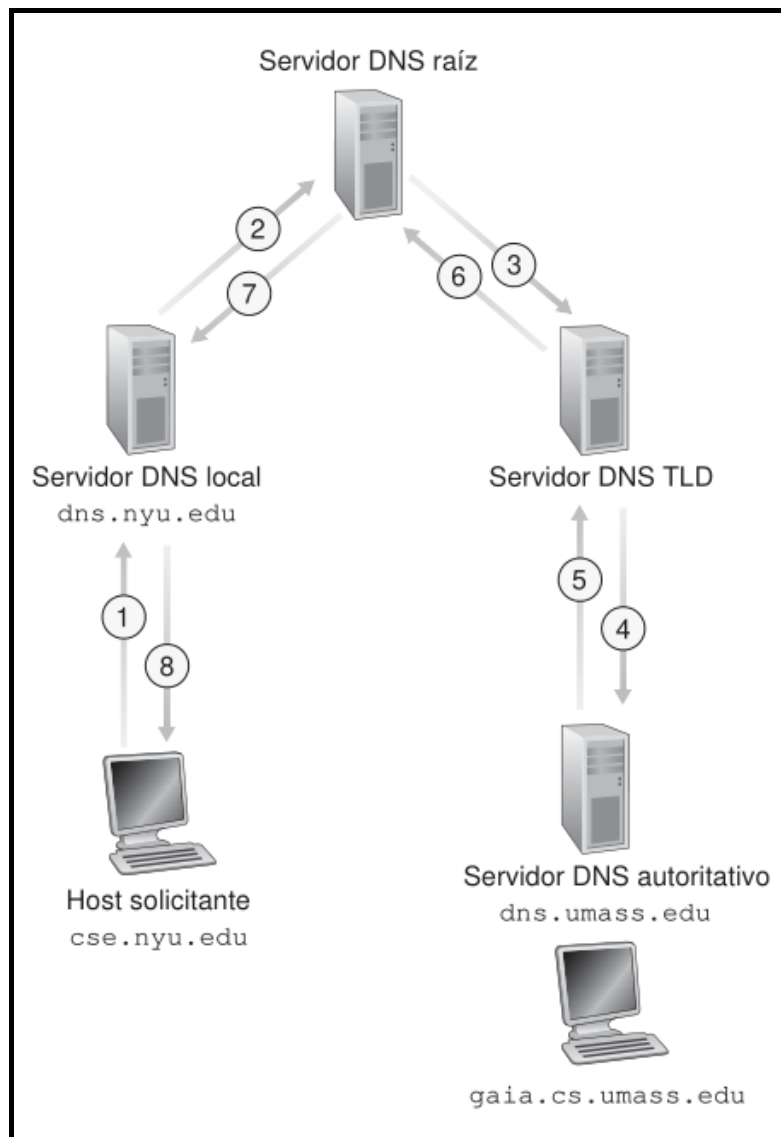
Una **consulta iterativa** es uno en el que se espera que el servidor DNS responda con la mejor información local que tiene, basado en lo que sabe el servidor DNS de los archivos de zona local o de la caché. Esta respuesta es también conocida como una remisión, si el servidor DNS no está autorizado para el nombre. Si un servidor DNS no tiene ninguna información local que puede responder la consulta, simplemente envía una respuesta negativa. Un servidor DNS realiza este tipo de consulta al intentar encontrar nombres fuera de su dominio local (o dominios) (al no está configurado con un reenviador). Podría tener que consultar un número de servidores DNS externos en un intento de resolver el nombre.

2.4.5. Ejemplo

Suponga que el host `cse.nyu.edu` desea conocer la dirección de `gaia.cs.umass.edu`. Suponga también que el servidor DNS local de la Universidad de Nueva York para `cse.nyu.edu` se denomina `dns.nyu.edu` y que un servidor DNS autoritativo para `gaia.cs.umass.edu` se llama `dns.umass.edu`. Como se muestra en la Figura, el host `cse.nyu.edu` envía en primer lugar un mensaje de consulta DNS a su servidor DNS local, `dns.nyu.edu`. El mensaje de consulta contiene el nombre de host que debe ser traducido, `gaia.cs.umass.edu`. El servidor DNS local reenvía la consulta a un servidor DNS raíz, el cual toma el sufijo `edu` y devuelve al servidor DNS local una lista de las direcciones IP de los servidores TLD responsables del dominio `edu`. El servidor DNS local reenvía a continuación el mensaje de consulta a uno de estos servidores TLD. El servidor TLD toma nota del sufijo `umass.edu` y responde con la dirección IP del servidor DNS autoritativo correspondiente a la Universidad de Massachusetts, es decir, `dns.umass.edu`. Por último, el servidor DNS local reenvía la consulta directamente a `dns.umass.edu`, que responde con la dirección IP de `gaia.cs.umass.edu`. Observe que en este ejemplo, para obtener la dirección correspondiente a un nombre de host, se han enviado ocho mensajes DNS: ¡cuatro mensajes de consulta y cuatro mensajes de respuesta! Pronto veremos cómo el almacenamiento en caché DNS reduce este tráfico de consultas.



El ejemplo mostrado en la Figura utiliza tanto consultas recursivas como consultas iterativas. La consulta enviada desde `cse.nyu.edu` a `dns.nyu.edu` es una consulta recursiva, ya que la consulta solicita a `dns.nyu.edu` que obtenga por sí mismo la correspondencia. Pero las tres consultas siguientes son iterativas puesto que todas las respuestas son devueltas directamente a `dns.nyu.edu`. En teoría, cualquier consulta DNS puede ser iterativa o recursiva. Por ejemplo, la Figura 2 muestra una cadena de consultas DNS para las que todas las consultas son recursivas. En la práctica, las consultas normalmente siguen el patrón mostrado en la Figura 2: la consulta procedente del host que hace la solicitud al servidor DNS local es recursiva y las restantes consultas son iterativas.



2.5. ¿Cuál es el objetivo del protocolo FTP? Explique cómo funciona, y sus modos de operación (Activo, Pasivo)

Su nombre significa **File Transfer Protocol** sirve para transferir ficheros entre un cliente y un servidor, el cual permite navegar por los directorios de ambos.

Una de las cosas curiosas que tiene FTP es que crea dos conexiones, canales; por un canal se envían las señales de control y por el otro canal, los datos. Primero el cliente establece la conexión de control del puerto 21 del servidor, proporcionando clave y contraseña en caso de ser necesario. Una vez establecida, el cliente puede enviar comandos para navegar por los directorios del servidor, buscando la ubicación que le interesa.

A la hora de conectarse con un servidor FTP, el cliente podría utilizar

- El modo **ACTIVO** (o modo PORT): Eligiendo un puerto aleatorio, por encima del 1024 y conectándose con el puerto 20 del servidor para enviar esos datos. Por ejemplo: el cliente elige el puerto 2000, envía un comando PORT al servidor indicándole el puerto que ha elegido, de manera que el servidor pueda abrirle una conexión de datos donde se transferirán los archivos y los listados en ese puerto. Esto tiene un **PROBLEMA DE SEGURIDAD** y es que la máquina cliente debe estar dispuesta a aceptar cualquier conexión por sobre el puerto 1023, con los problemas que ello implica en el firewall.
- Para solucionar el problema de seguridad que acarrea el Modo Activo, se desarrolló el Modo **PASIVO** (o modo PASV): cuando el cliente envía un comando PASV por el canal de control, el servidor FTP le indica su puerto (el puerto del servidor, el cual debe ser mayor que el 1023), por ejemplo 2040, al que debe conectarse el cliente. Por lo que el cliente es el que inicia la conexión desde el puerto siguiente al puerto que tenga el control (recordar que son dos puertos los que abre el cliente, uno para control (por ejemplo 1034) y otro para datos (por ejemplo 1035)). Y como es el cliente el que inicia la conexión, no hay problemas de seguridad. En caso de tener que abrir otro fichero, se debe abrir otra conexión.

Al final el protocolo FTP, sirve para lo mismo que el protocolo HTTP, traer ficheros de un servidor y ambos se ejecutan sobre TCP; de hecho los navegadores también interpretan el protocolo FTP, por lo que también sirven como clientes de este protocolo.

2.5.1. Diferencia entre FTP y HTTP

- La principal diferencia que existe entre FTP y HTTP es el uso que hace FTP de las dos conexiones (una parte para control, donde envía los comandos y otra para los datos). Mientras que en HTTP tanto comandos como datos, viajan sobre la misma conexión, mediante los mensajes HTTP.
- HTTP envía la información de cabecera de solicitud y respuesta por la misma conexión TCP que transfiere el archivo (en banda), mientras que FTP envía la información de control *fuera de banda* (lo mismo que realiza el protocolo RTSP).
- El servidor FTP tiene que mantener el estado, ya que debe recordar la carpeta en la que se encuentra el usuario, el modo de transferencia de ficheros que le han sido solicitados, etc.

2.6. Si se conecta a un servidor de correo por medio de un navegador web, ¿Qué protocolo uso para enviar y recibir correo? ¿Y si se conecta mediante un agente de correo (por ejemplo Outlook)? ¿Qué diferencias existen?

Los servidores de correo sólo se comunican entre sí, utilizando SMTP. No utilizan otros ya que, ellos mismos no se preguntan si tienen ellos mismos correo, entre ellos sólo se envían correos, por ende utilizan el protocolo SMTP. Los agentes usan SMTP y otros para descargar los correos.

Tres fases de la transferencias:

1. Saludo (handshaking)
2. Transferencia de correo
3. Cierre (creo)

Cuando se quiere ver el mail desde web se **utiliza el protocolo HTTP**, pero sólo entre los servidores utilizan en SMTP, ojo con esto!. Hay que recordar que cuando se envía un correo desde el agente hasta otro cliente, se utiliza el protocolo SMTP.

Protocolos de acceso de correo siempre lo usan los agentes de correo, solo ellos (outlook, thunderbird, etc). IMAP o POP3 son los protocolos para poder descargar los correos:

- **POP3:** las configuraciones no se guardan (modificaciones tales como crear carpetas, poner etiquetas, etc). Fue el primero que se implementó, era bastante básico.
- **IMAP:** las configuraciones se guardan, todo lo que se hace en el agente de correo, se guarda dentro de los servidores de correo, las configuraciones son permanentes en el servidor. Por lo que la configuración sobre los distintos correos que se realicen en el Cliente de Correo, será también modificado en el servidor.

2.6.1. Funcionamiento de los protocolos de correo y ejemplo

Como elementos fundamentales para un servicio de Correo Electrónico se encuentran:

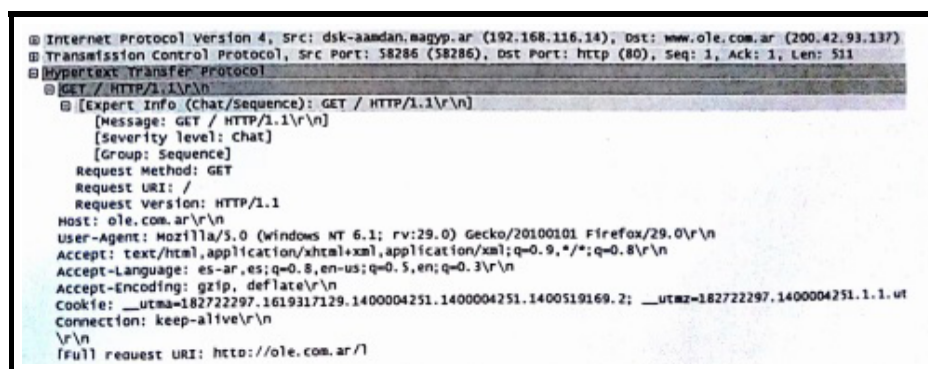
1. Los clientes de Correo: que pueden ser aplicaciones específicas como Thunderbird u Outlook, o directamente desde el navegador.
 - Estos permiten: componer, edita, lee y envía mensajes de correo.
 - Los correos entrantes y salientes se almacenan en el servidor.
2. Servidores de correo que tendrán:
 - Buzón (o mail box) que contiene los mensajes entrantes de un usuario.
 - Cola de mensajes de los correos salientes que tienen que ser enviados.
3. Protocolo SMTP (Simple Mail Transfer Protocol), el cual comunicará los servidores de correo con otros.
 - Utiliza TCP para enviar mails entre servidores de correo.
 - Lado cliente en el servidor de correo del emisor y lado servidor en el servidor de correo del destinatario.

Si Alicia quiere enviar un correo a Bob, entonces lo que sucede es lo siguiente:

1. Alicia utiliza un Cliente de Correo (A), en el cual escribe o edita el correo deseado, y luego lo envía.
2. El Agente/Cliente de Correo de Alicia (B) envía el mensaje a su servidor de correo (un servidor aparte del Cliente que utiliza Alicia) que lo pone en su cola de mails.
3. El Servidor de Correo utiliza SMTP para contactarse con el otro Servidor de Correo (C) que utilizará el Cliente/Agente de Bob (D), y abre una conexión TCP.
4. El cliente SMTP envía el mensaje de Alicia por la conexión TCP (de B a C).
5. EL Servidor de Correo de Bob (C) almacena el mansaje en su buzón (mailbox).
6. Bob invoca a su Agente de Correo (D) para leerlo cuando quiera.

2.7. En base a las siguientes figuras: ¿Qué tipos de mensajes son? Describa las características y cabeceras presentes en ellas

2.7.1. Figura nº 1



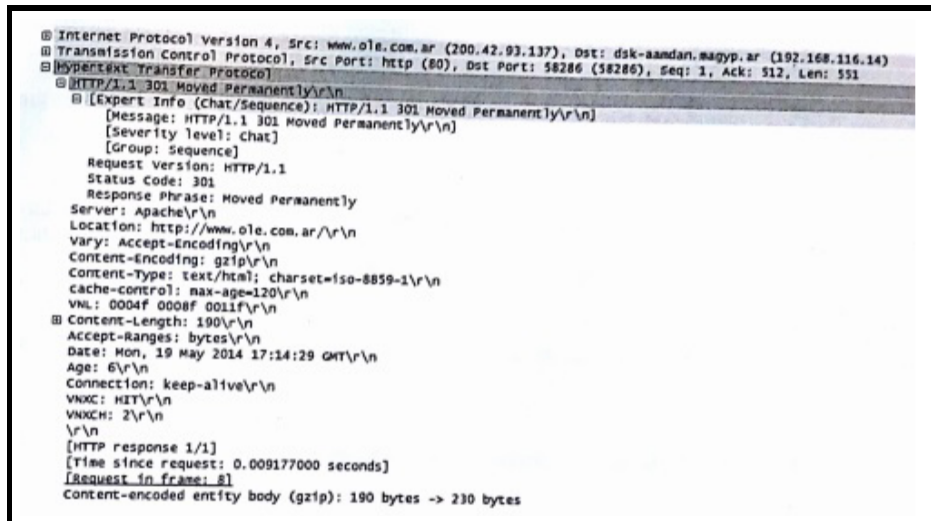
```

@ Internet Protocol Version 4, Src: dsk-aandan.magyp.ar (192.168.116.14), Dst: www.ole.com.ar (200.42.93.137)
@ Transmission Control Protocol, Src Port: 58286 (58286), Dst Port: http (80), Seq: 1, Ack: 1, Len: 511
@ Hypertext Transfer Protocol
  @ GET / HTTP/1.1\r\n
    @ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
      [Message: GET / HTTP/1.1\r\n]
      [Severity level: Chat]
      [Group: Sequence]
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
      Host: ole.com.ar\r\n
      User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:29.0) Gecko/20100101 Firefox/29.0\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
      Accept-Language: es-ar;q=0.8,en-us;q=0.5,en;q=0.3\r\n
      Accept-Encoding: gzip, deflate\r\n
      Cookie: __utma=182722297.1619317129.1400004251.1400004251.1400519169.2; __utmz=182722297.1400004251.1.1.ut
      Connection: keep-alive\r\n
      \r\n
      [Full] request URI: http://ole.com.ar/
  
```

Según la figura nº 1, es un tipo de mensaje Request (utilizando el método GET) de tipo HTTP (que se puede visualizar en el campo **Message**) cuyas características son las siguientes:

- Según el campo **Host** se solicita la página web `ole.com.ar`.
- El agente que se utilizó para lanzar este tipo de petición es el navegador Mozilla, con un Sistema operativo Windows NT 6.1, como lo indica el campo **User-Agent**.
- El agente también en el request, especifica que acepta los lenguajes en el siguiente orden: `es-ar`, `en-us`, `en` como se puede visualizar en el campo **Accept-Language**.
- Utiliza un cookie, por lo que se entiende que el usuario ha consultado previamente la página solicitada, enviándole esta información previa, mediante el nombre de la cookie que está descrita en el campo **Cookie**.
- Se puede visualizar que el tipo de conexión solicitada es persistente, ya que se indica *Keep-alive* en el campo **Connection**, lo que significa que se podrán enviar varios objetos antes de cerrar la conexión.

2.7.2. Figura nº 2



Según la Figura nº 2, se puede visualizar la respuesta del servidor a la petición de la Figura nº 1, en los campos de cabecera, donde:

- En el campo **Message** se indica que hubo un error 301, el cual es un código de estado de HTTP que indica que el host ha sido capaz de comunicarse con el servidor pero que el recurso solicitado ha sido movido a otra dirección permanentemente.
- El campo de cabecera **Server** permite saber que el tipo de servidor que contesta al Request es un Apache.
- **Content-Encoding** : La codificación aceptada, es decir, el algoritmo de compresión utilizado para transferir los datos es de tipo gzip.
- El campo **Content-Type** permite saber que el contenido es de tipo texto y HTML, el cual está codificado mediante el set de caracteres ISO-8859.
- El campo **Content-Lenght** permite saber el tamaño del cuerpo del mensaje en bytes que es de 190.
- El campo **Date** permite saber que el mensaje fue enviado el día Lunes 19 de Mayo del año 2014 a las 17:14:29 horas, para la zona horaria GMT
- El campo **Age** indica el tiempo en segundos que el objeto ha estado en el servidor caché proxy, es decir 6 segundos.
- Se puede visualizar que el tipo de conexión sigue siendo persistente, como lo indica el tipo *Keep-alive*, en el campo **Connection**.