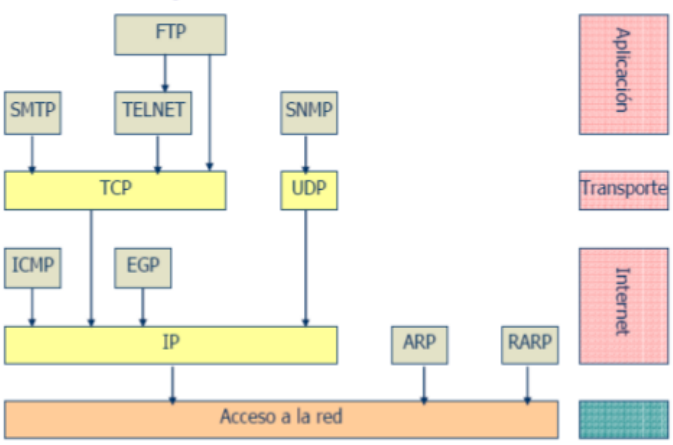
TEMA 2. SERVICIOS Y PROTOCOLOS DE APLICACIÓN EN INTERNET.

1. Introducción a las aplicaciones de red.

- Estructura de protocolos:



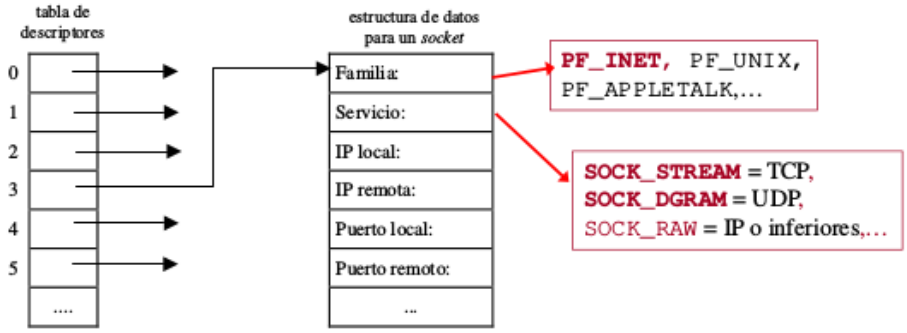
-SMTP: Correo electrónico.
-FTP: Transferir archivos.
-TELNET: Conexión entre una máquina y otra remotamente.
-SNMP: Gestión de red.
-HTTP: Protocolo de web.
-TCP: Protocolo con conexión fiable.
-UDP: Más eficiente y ligero con menos funcionalidad que TCP.
-IP: Protocolo de la capa de internet.
-ICMP: Gestión de redes.
-EGP: Calcula la dirección de destino.
-ARP: Calcula direcciones MAC.

- Arquitectura cliente/servidor:
  Un servidor es una aplicación que permite el acceso remoto a ciertos recursos software o hardware en un host. La aplicación servidora está siempre en escucha hasta que le llega alguna solicitud remota por parte de un cliente.

  El servidor debe estar siempre en funcionamiento para poder estar siempre a la escucha de peticiones, debe tener una IP permanente y pública para que sea fácil de localizar y, sobre todo en grandes “data centers” suelen estar agrupados en granjas.

  Por otro lado, los clientes funcionan intermitentemente, no tienen por qué estar pagando una IP permanente para su router por lo que ésta suele ser dinámica, y al acceder desde un dispositivo distinto al router también suelen tener IP privada.

- La interfaz socket:
  Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red. En la práctica, es un puntero a una estructura.



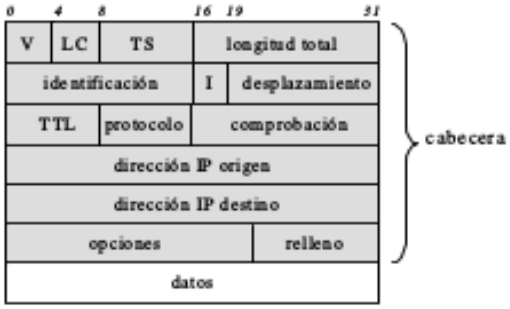
- Retardo en cola:
  El tiempo de servicio es el tiempo medio en dar servicio a un solicitante (desde que éste envía la petición hasta que obtiene una respuesta). Sirve tanto para páginas como para routers, que hacen un servicio de reenvío al solicitante.

  El retardo en cola se calcula mediante la teoría de colas:

  $$R = \frac{\lambda \cdot (T_s)^2}{1 - \lambda \cdot T_s}$$

  Ts: es el tiempo de servicio
  λ: tasa de llegada de solicitudes

- ¿Qué definen los protocolos de aplicación?
  - El tipo de servicio:
    - Orientado o no orientado a conexión.
    - Realimentado o no.
  - El tipo de mensaje:
    - Request (petición), response (respuesta).
  - La sintáxis:
    - Definición y estructura de “campos” en el mensaje.



  - La semántica:
    - Significado de los campos.
  - Las reglas:
    - Cuándo los procesos envían mensajes/responden a mensajes. Para que la comunicación sea efectiva debe hacerse siguiendo unas reglas.

- Tipos de protocolos:
  - *Protocolos de dominio público*, este tipo de protocolos buscan la colaboración/feedback de la gente para encontrarle fallos o mejorarlos}. Normalmente buscan ser estandarizados. Por ejemplo HTTP, SMTP, etc. *VS propietarios*, al contrario que los de dominio público, son creados por una empresa e implantados en un producto. Estas empresas intentan esconderlo lo máximo posible para que nadie sepa cómo funciona el protocolo.

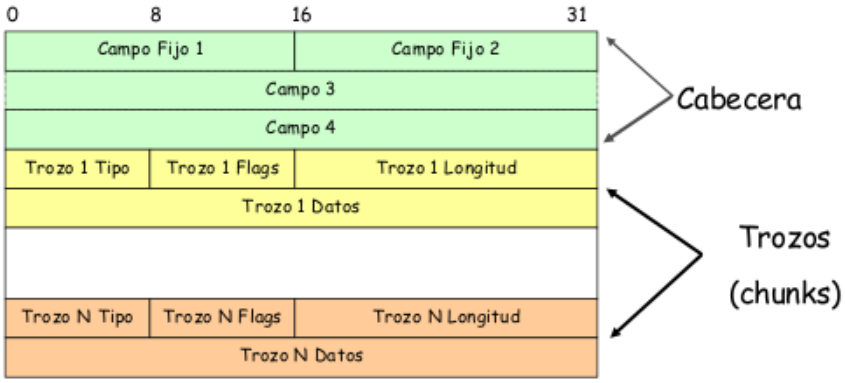
- *Protocolos in-band*, mandan en el mismo paquete la información completa, tanto de datos como de control (HTTP), *VS out-of-band*, usan canales distintos para cada cosa (FTP). Por ejemplo, FTP envía la información de control (usuario, contraseña, comandos get/post, etc) por una conexión y los datos, por otra conexión paralela separada.

- *Protocolos stateless*, no guardan información del cliente, a no ser que tengan una intranet o un fichero como las cookies, *VS state-full*, son servicios que guardan el estado del cliente, por ejemplo, el carrito de la compra, si eres premium o no.

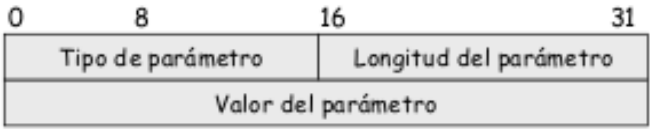
- *Protocolos persistentes*, siempre están conectados, *VS no-persistentes*, crean una conexión cada vez que se manda un trozo de información.

- <u>Tendencia; hacer los protocolos flexibles con:</u>
  - Una cabecera fija
  - Una serie de "trozos" (obligatorios y opcionales)

  La tendencia hoy en día, para no trabajar con cabeceras muy grandes, es hacer cabeceras flexibles, es decir, poner una cabecera fija muy pequeñita y luego añadir trozos según lo que necesitemos. Al principio de cada trozo se indica su longitud y la información que contiene dicho trozo.



  Los trozos pueden incluir una cabecera específica más una serie de datos en forma de parámetros:
  - Fijos: deben seguir un determinado orden.
  - De longitud variable u opcionales.
  - Siguiendo el formato TLV:



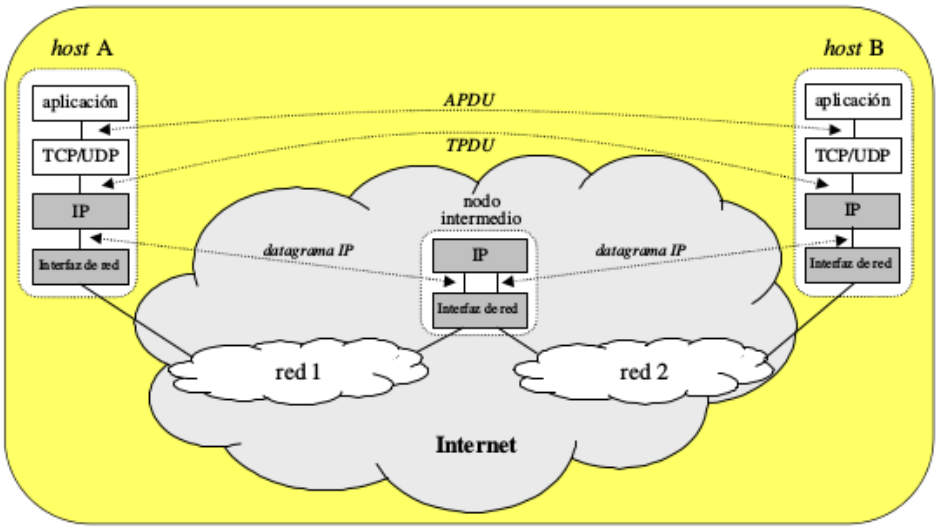
- <u>Características:</u>
  - Pérdidas de datos (errores): algunas aplicaciones (como por ejemplo de audio o de video) pueden tolerar alguna pérdida de datos, ya que el usuario no se da cuenta si pierde un frame del video o un microsegundo de una canción. Sin embargo, otras (como por ejemplo TELNET) requieren una transferencia 100% fiable sin pérdida alguna.

  - Requisitos temporales: Algunas apps denominadas inelásticas (ej., telefonía Internet, juegos interactivos) requieren retardo acotado (delay) para ser efectivas.

  - Ancho de banda: Algunas apps requieren envío de datos a un ritmo determinado.

- Seguridad: Encriptación, autenticación, no repudio (es decir, no poder negar las acciones que has hecho)…

- Protocolos de transporte:



  - Servicio TCP: lo usan los servicios que quieren fiabilidad, que no se pierdan datos. Está orientado a conexión, tiene control de flujo y de congestión implementados.

  - Servicio UDP: lo usan los servicios que quieren rapidez. UDP no tiene nada de lo que TCP tiene implementado, aunque nosotros podemos implementarlo.

  TCP y UDP, que están en la capa de transporte, al ser usuarios del protocolo IP, que está en la capa de red, **no garantizan**:
  - Retardo acotado: no garantizan que la respuesta se obtenga en un intervalo de tiempo máximo.
  - Fluctuaciones acotadas: no garantiza que haya una velocidad fija en la red.
  - Mínimo throughput: no garantiza el envío de paquetes a un ritmo mínimo constante.
  - Seguridad: no garantiza que los envíos lleguen tal cual se han enviado.

2. Servicio de Nombres de Dominio (DNS).
La comunicación en Internet precisa de direcciones IP asociadas a las interfaces finales involucradas en esta comunicación. Estas direcciones IP son las direcciones que identifican a cada máquina y las que se usan para direccionarlas. Como para los usuarios usar esto no es cómodo, se utilizan los nombres de dominios, que están asociados a dichas IP's.

Debido a esto, se necesita un sistema para convertir los nombres de dominio a direcciones IP (servicio de resolución de nombres) y así surge el sistema de nombres de dominio o Domain Name System, DNS.

Goliat.ugr.es ← → 150.214.20.3

*Al nivel 1 se le denomina dominio genérico (.com, .es, …).

Para que DNS funcione, debe existir una gran base de datos que albergue a qué IP's pertenecen los nombres de dominio y viceversa, para poder hacer la traducción. Tener esta base de datos centralizada supondría que todo el tráfico que hay en internet fuera a esta base de datos, ralentizando muchísimo el servicio y suponiendo un mantenimiento de esta base datos casi imposible. Para hacer esta traducción sostenible, DNS hace uso de una gran cantidad de servidores distribuidos por el mundo para mapear los hosts de Internet. Estos servidores son:

- *Servidores raíz "."*: Este tipo de servidores se encargan de resolver una petición para la resolución de un nombre y devuelve una lista de los servidores TLD que son capaces de resolver la petición.

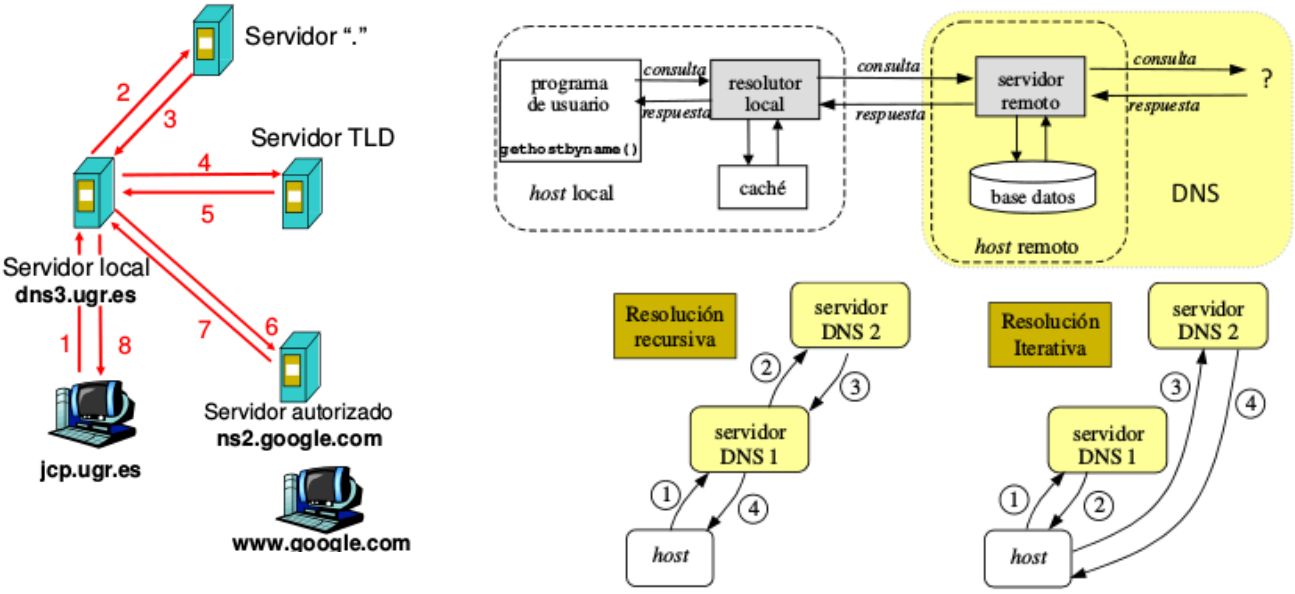
- *Servidores de dominio (TDL)*: Estos servidores se encargan de los dominios de más alto nivel como .com, .net, etc.. Se encargan de resolver la petición DNS que le ha enviado el servidor root, y si no son capaces de hacerlo, darán la dirección del servidor autorizado capaz de resolver esta traducción.

- *Servidores locales:* Un servidor local es aquel que tiene cada proveedor de internet, red de universidad... (ISP) posee un servidor local que resolverá todas las peticiones DNS que pertenecen a su dominio.

- Resolución:
  - Iterativa: en este caso, cuando se realiza una consulta DNS y el servidor no la puede responder, este le pide al servidor que puede resolver la consulta que la resuelva y se la de al host que ha realizado la consulta inicial.

  - Recursiva: el host que realiza la consulta se la manda a un servidor. Si este no la puede resolver, se la realizará a otro, y este a otro si no la puede resolver, y así sucesivamente.

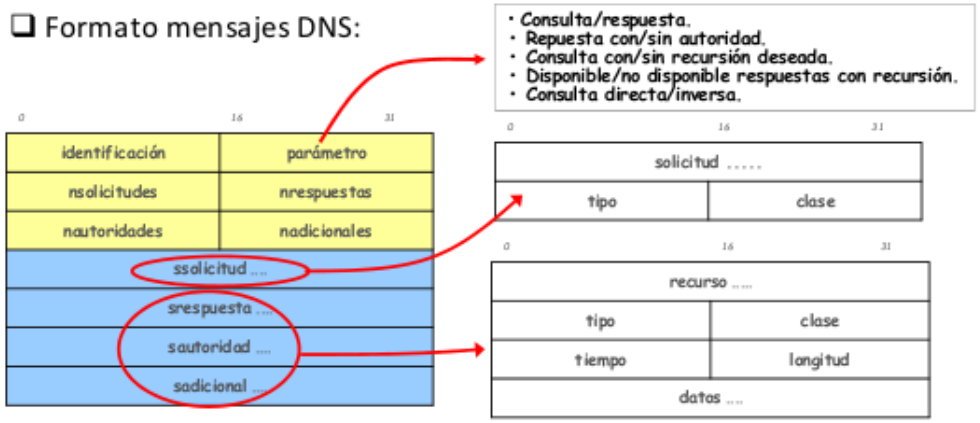
- Uso de caché: el host realiza la petición al servidor local, este mira en caché y si puede responder, responde y si no, le envía la consulta a otro servidor remoto.



- <u>Gestión de la base de datos distribuida y jerárquica:</u>
  - Está formada por un conjunto de servidores cooperativos que almacenan parcialmente la BD que se denomina (BIND).
  - Cada servidor es responsable de lo que se denomina **zona**.
  - Una **zona** es un conjunto de nombres de dominio contiguos de los que el servidor tiene toda la información y es su **autoridad**.
  - Los **servidores autoridad** deben contener **toda** la información de su zona.
  - La autoridad puede **desplegarse** jerárquicamente a otros servidores.
  - Cada zona debe tener **al menos** un servidor de autoridad.

- En cada zona hay servidores **primarios** (copia de la BD) y **secundarios** (obtiene la BD por transferencia).
- Existe un servicio caché para mejorar prestaciones.
- Cuando un cliente solicita una resolución de nombres a su servidor puede ocurrir:
  - *Respuesta CON autoridad*: el servidor tiene autoridad sobre la zona en la que se encuentra el nombre solicitado y devuelve la dirección IP.
  - *Respuesta SIN autoridad*: el servidor no tiene autoridad sobre la zona en la que se encuentra el nombre solicitado, pero lo tiene en la cache.
  - *No conoce la respuesta*: el servidor preguntará a otros servidores de forma recursiva o iterativa. Normalmente se "eleva" la petición a uno de los servidores raíz.

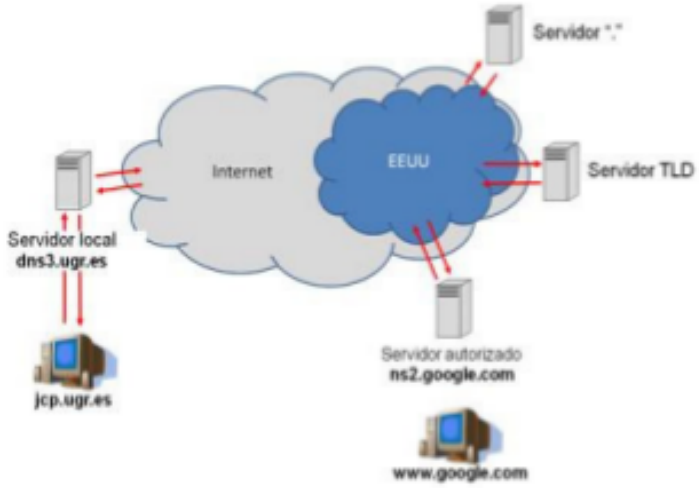
- ¿Cómo es la BD DNS?
  - Todo el dominio está asociado a un registro **R**esourse **R**ecord.
  - Cada **RR** es una tupla con 5 campos:
    - Nombre del dominios
    - Tiempo de vida: tiempo de validez de un registro para la caché.
    - Clase: en internet siempre IN.
    - Tipo: tipo de registro.
      - SOA: Registro con la autoridad de la zona.
      - NS: Registro que contiene un servidor de nombres.
      - A: Registro que define una dirección IP.
      - MX: Registro que define un servidor de correo elect.
      - CNAME: Registro que define el nombre canónico de un nombre de dominio.
      - HINFO: Información del tipo de máquina y sistema operativo.
      - TXT: Información del dominio.
    - Valor: contenido que depende del campo tipo.
  - Existe una BD asociada de **resolución inversa** para traducir direcciones IP en nombres de dominio.
  - Formato de mensajes DNS:



  - DNS se ofrece en el puerto 53 mediante UDP normalmente o TCP (para respuestas grandes > 512 bytes).

Ejercicio:

6. En la siguiente figura se ilustra un ejemplo de acceso DNS por parte de una máquina (jcp.ugr.es) que quiere acceder a los servicios de www.google.com. Para obtener la dirección IP del servidor, es necesario que la consulta pase por todos los servidores del gráfico. Considerando unos retardos promedio de 8 µs dentro de una red LAN, de 12 ms en cada acceso a través de Internet (4 ms si la conexión se restringe a EEUU) y de 1 ms de procesamiento en cada servidor:



1. Calcule el tiempo que se tardaría si la solicitud al servidor local es recursiva, pero el propio servidor
local realiza solicitudes iterativas.

2. Especifique una política (recursiva-iterativa) más rápida de solicitudes y el tiempo que tardaría la
solicitud en ser respondida. ¿Qué desventaja tiene sobre la solución anterior?

De la máquina jcp.ugr.es al servidor local,

seg = 8 * 10^{-6} = 0,000008

Del servidor local a cada uno de los tres servidores de la figura,

seg = 3*(12*10^{-3}) = 0,036

Además, en cada máquina debe dedicarse 1ms de procesamiento,

seg = 1*10^{-3} = 0,001

Así obtenemos la siguiente fórmula,

Tt = (2 * 0,000008) + (2 * 0,036) + (4 * 0,001) = 0,076016 seg.

El tiempo que tardaría con peticiones recursivas sería menor, ya que dos peticiones se realizan dentro de EEUU y por tanto, sólo tardan 4ms en ser respondidas:

Trecursivo = (2 * 0,000008) + (2 * (12*10^{-3})) + (2 * (4 * 0,001) * 2) + (4 * 0,001) = 0,044016 seg.

3. La navegación Web.

La navegación Web consiste en pedir una información a un servidor, y este nos la devuelve de forma que nuestro navegador sea capaz de interpretarla. Esta información puede ser texto (denominado hipertexto y puede contener enlaces a otras páginas, denominándose hiperenlaces), imágenes, audio, vídeo…

Esta información se concentra en un fichero HTML que conforma el esqueleto de la página, siendo esto lo primero que mira el navegador que interpreta el código. A continuación puede mirar las páginas de estilo asociadas, imágenes, audio…

- Protocolo HTTP
  Se implementa en dos programas, utilizando la arquitectura cliente-servidor.
  - Cliente: Es nuestro navegador web, que realiza peticiones de objetos y los muestra.
  - Servidor: Atiende las peticiones que le llegan y envía los objetos que son solicitados.

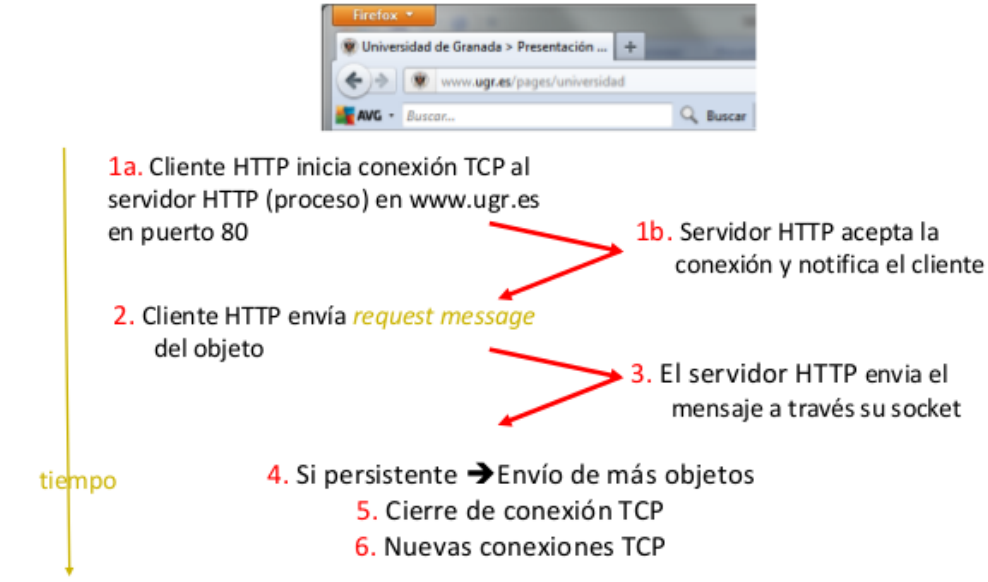
- Características HTTP
  - HTTP se basa en conexiones TCP al puerto 80, iniciando el cliente la conexión TCP para solicitar datos, y el servidor la recibe en la interfaz del socket asociado. Entonces el servidor, envía por ese socket la información que le han solicitado y cierra la conexión TCP, recibiendo el cliente la información pedida.

  - HTTP es un protocolo “stateless", esto significa que el servidor (en principio), no guarda ninguna información del cliente, por lo que no guarda el estado del cliente, ni ningún

tipo de información respecto a él. Esto se debe a que si se guardara la información de los clientes, sería imposible almacenar este tipo de información. Para ello se usan las cookies, que almacenan la información del cliente en el propio cliente.

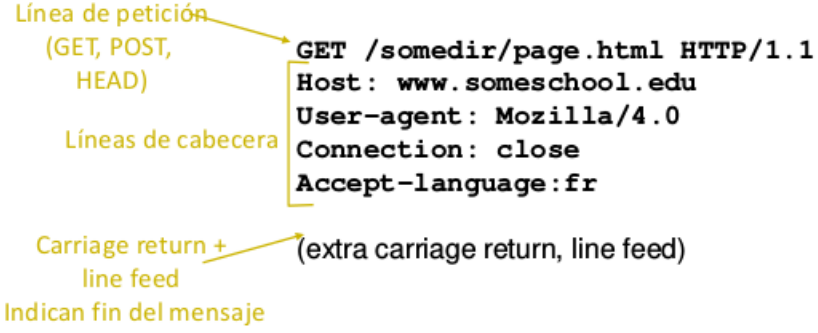
- Existen dos tipos:
  - No persistente → Se envía únicamente un objeto en cada conexión TCP, es decir, cuando ha terminado de enviar el objeto cierra la conexión TCP. Es decir, se abriría una conexión para pedir el html, otra para pedir la hoja de estilos, etc.
  - Persistente → Se pueden enviar varios objetos a través del socket en una misma conexión TCP entre el cliente y el servidor.

    *Lo que realmente hacen los navegadores es lanzar conexiones persistentes, así, cuando pedimos una página web con nuestro navegador, éste no lanza un sólo cliente con un sólo socket al puerto x sino “tres o cuatro": con el primero pide la página web y con el resto, las imágenes, videos, etc. Los navegadores están muy optimizados en este aspecto.

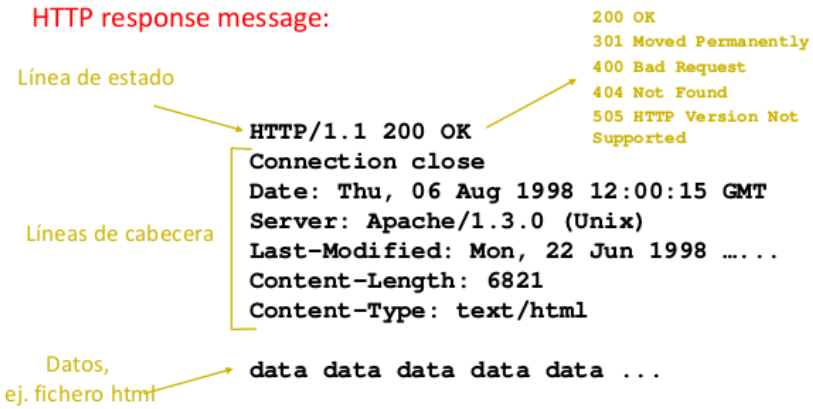


- <u>Tipos de mensajes HTTP:</u>
  - request: El comando más utilizado es el GET, junto a éste, indicamos el path y la versión de HTTP que entiende el navegador, que normalmente suele ser la más avanzada, aunque si el servidor entiende una anterior, tendrá que hacerse la comunicación con esa. Hay alternativas a GET, las más típicas son POST y HEAD.
    - GET manda toda la información en forma de path, cuando por ejemplo pedimos un determinado archivo de la página, lo hacemos mandando información de su path.
    - POST toda la información que mandamos aparece en la sección ``extra carriage return, line feed".
    - HEAD se usa para comprobar la configuración del servidor, es decir, aunque pidamos un determinado objeto del servidor, el servidor contestará como si lo hubiese enviado pero sin enviarlo.

      *Connection: close (el cliente está pidiendo que la conexión se cierre, es decir, una conexión no persistente).



- Response:
  *Content-Length (tamaño de los datos que se van a enviar).



Tras esta cabecera, se procede a enviarse los datos.

Ejercicio:
8. Compare el rendimiento en términos temporales de HTTP persistente y no persistente considerando los siguientes parámetros:

Descarga de una página web con 10 objetos incrustados
Tiempo de Establecimiento de conexión TCP → 5 ms
Tiempo de Cierre de conexión TCP → 5 ms
Tiempo de solicitud HTTP → 2 ms
Tiempo de respuesta HTTP (página web u objeto) → 10 ms

En una conexión persistente en la que queremos descargar una página web con diez objetos primero debemos establecer la conexión TCP, una vez establecida la conexión se solicitan tanto la página web como los objetos incrustados en ella y por último, se cierra la conexión. Así, se obtendría el siguiente tiempo de descarga:

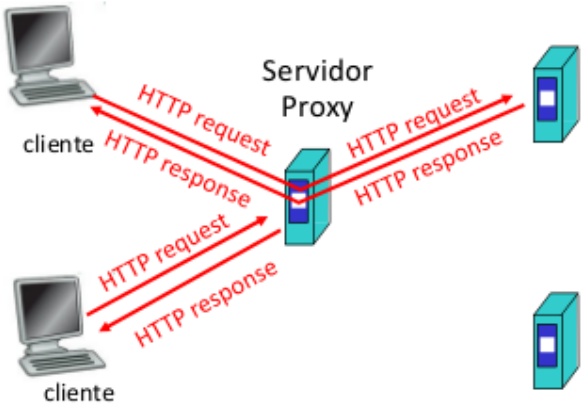
Td = Te + 11 * (Ts + Tr) + Tc = 5 + 11 * (2 + 10) + 5 = 142ms
*11 = pagina web + 10 objetos incrustados.

Por otro lado, en una conexión no persistente tendríamos que abrir y cerrar la conexión cada vez que quisiéramos descargar un objeto, por tanto, tendríamos el siguiente tiempo de descarga:

Td = 11 * (Te + Ts + Tr + Tc) = 11 * (5 + 2 + 10 + 5) = 242ms

La conexión persistente obtiene un mejor rendimiento en este caso.

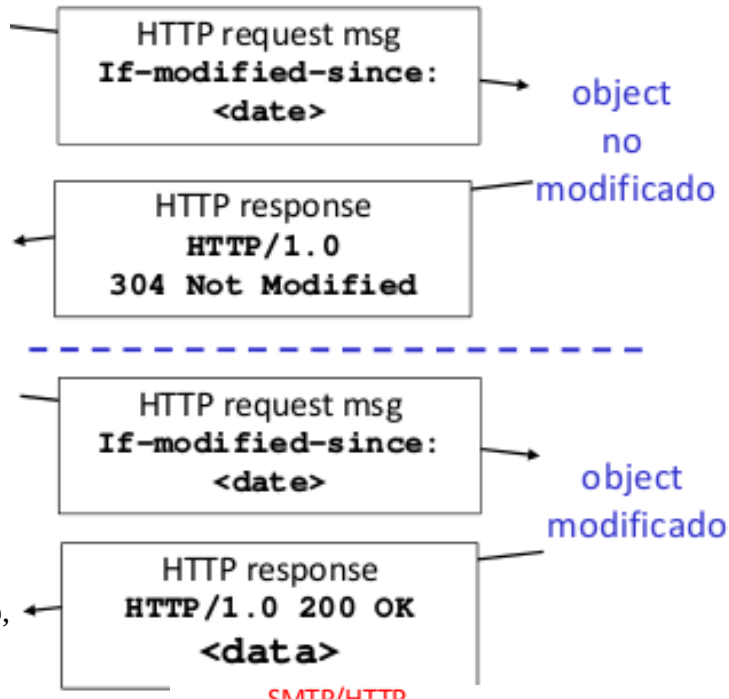
- Web cache
  La cache sirve para no tener que utilizar ancho de banda para resolver una petición que se ha hecho recientemente.



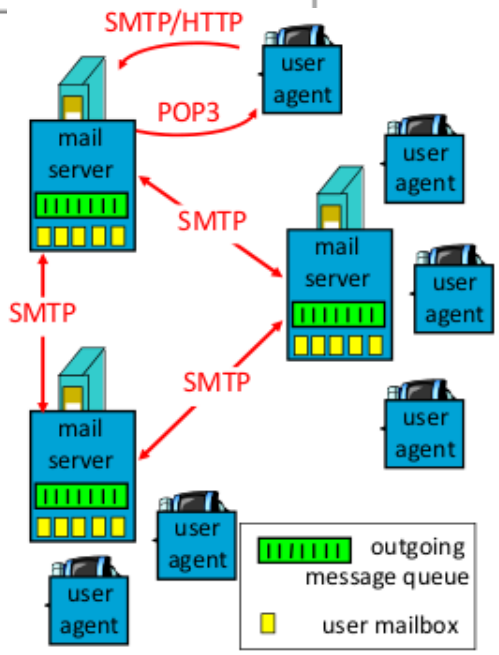
-Un cliente hace una petición al servidor proxy y éste se da cuenta de que no tiene dicha petición en cache, por lo que hace la petición al servidor en concreto.
- Éste le devuelve la respuesta al servidor proxy y éste, se la da al cliente.
-Tras esto, otro cliente distinto vuelve a pedir la misma página web, por lo que directamente el servidor proxy le responde con ésta. Esta respuesta es mucho más rápida y es la que nos ahorra ancho de banda.

Ejemplo de respuesta:

HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html



Esta respuesta lo que nos está indicando es si el objeto está actualizado en la cache o no. Por ejemplo, cuando accedemos a una web que no está en cache, el proxy reenvía la solicitud al servidor y nos reenvía la respuesta a nosotros, pero también guarda la respuesta en su cache. Cuando otro cliente solicita la misma página web, el proxy no sabe si la copia que tiene en cache está actualizada o no, por lo que tiene que asegurarse de que tiene una copia actualizada. Para ello, hace una solicitud al servidor original preguntando por el objeto, pero a dicha solicitud le añade información sobre que el objeto está ya en su cache con una determinada fecha. Si el servidor no ha modificado ese objeto desde entonces, le responde diciendo que lo tiene actualizado, en caso contrario, le responderá con la versión actualizada del objeto.



4. El correo electrónico.

- <u>El correo electrónico:</u>
  - Componentes principales:
    - Cliente de correo (user agent) y Servidor de correo (mail server): son los dos agentes que intervienen (el remitente y el/los destinatario/s del correo) y los servidores. Por cada origen y cada destino, hay un cliente y un servidor.
    - Simple Mail Tranfer Protocol SMTP: es el protocolo utilizado para enviar correo electrónico, los otros protocolos de descarga que intervienen son POP3, IMAP, HTTP.
- <u>Características de SMTP:</u>
  - SMTP se implementa mediante dos programas:
    - Cliente SMTP: el servidor actúa como cliente cuando recibe un correo y tiene que reenviarlo a otro servidor. Para ello activa un puerto dinámico que se conectará con el puerto 25 del destino, y enviará el correo a través de esa conexión.
    - Servidor SMTP: cuando enviamos un correo, el servidor SMTP está escuchando peticiones en el puerto 25.

Se utiliza el protocolo TCP, lo cual tiene sentido porque TCP es el protocolo fiable de la capa de transporte y todos los servicios que no quieren perder ni un sólo carácter de la información que están enviando lo usan.

- SMTP es orientado a conexión, in-band y state-full, consta de 3 fases:
  - Handshaking (“saludo”), parte inicial de la conexión.
  - Transferencia de mensajes.
  - Cierre.

- Los mensajes deben estar codificados en ASCII de 7 bits → Extensiones MIME, que sirven para añadir otro tipo de codificación, sobre todo en los archivos adjuntos.

- <u>Pasos en el envío/recepción de correo:</u>

1) El usuario origen compone mediante su Agente de Usuario un mensaje dirigido a la dirección de correo del usuario destino

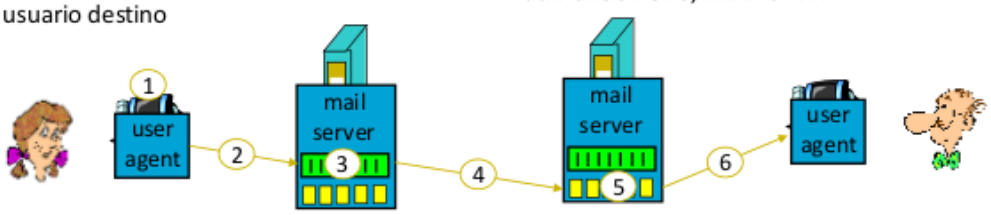
2) Se envía con SMTP o HTTP el mensaje al servidor de correo del usuario origen que lo sitúa en la cola de mensajes salientes

3) El cliente SMTP abre una conexión TCP con el servidor de correo (obtenido por DNS) del usuario destino

4) El cliente SMTP envia el mensaje sobre la conexión TCP

5) El servidor de correo del usuario destino ubica el mensaje en el mailbox del usuario destino

6) El usuario destino invoca su Agente de Usuario para leer el mensaje utilizando POP3, IMAP o HTTP



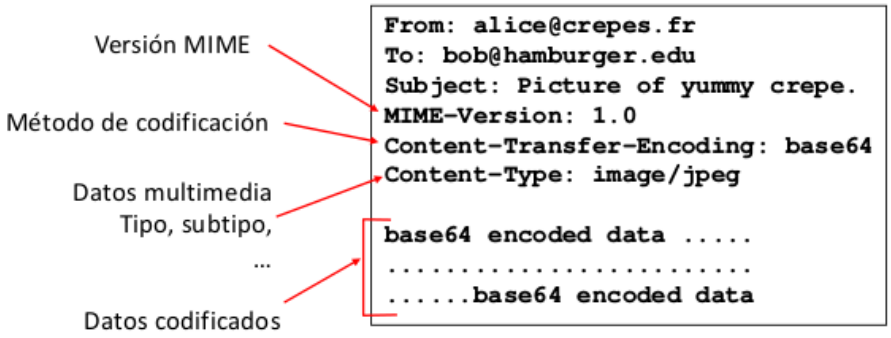
Ejemplo de intercambio de mensajes:

```
S: 220 smtp1.ugr.es
C: HELO ugr.es
S: 250  smtp1.ugr.es
C: MAIL FROM: uno@ugr.es
S: 250 Ok
C: RCPT TO: dos@ugr.es
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: Subject: Correo estúpido
C: Tengo ganas de enviarte un correo…
C: ¿Te importa si lo hago?
C: .
S: 250 Ok: queued as KJSADHFFWDF
C: QUIT
S: 221 Bye
```

*Se indica el remitente, el destinatario del correo. Se introduce “DATA” para indicarle al servidor que vamos a escribir el correo. El servidor nos responde que para terminar usemos un “.” en una linea en blanco. Una vez terminado se envía y se cierra la conexión.

- <u>Extensiones MIME</u>
  - Las extensiones MIME van encaminadas a soportar:
    - Texto en conjuntos de caracteres de US-ASCII.
    - Adjuntos que no son de tipo texto.
    - Cuerpos de mensajes con múltiples partes.
    - Información de encabezados con conjuntos de caracteres distintos de ASCII.

-Ejemplo de un archivo MIME:



En la imagen, se ve un ejemplo de un archivo codificado multimedia con una extensión MIME. Para ello se especifica la versión MIME con la que ha sido codificado, el método de codificación, el tipo de dato multimedia que se está enviando (vídeos, imágenes, etc) y tras algunos datos más, se adjunta la imagen en sí codificada.

- Protocolos de acceso (POP3)
  El protocolo POP3, al igual que SMTP, tiene una fase de conexión, una de servicio y otra de cierre.
  - Fase de autorización: El cliente se identifica con su nombre de usuario y contraseña, y el servidor le responde con +OK o -ERR en función de los datos introducidos.
  - Fase de transacción: Fase en la cual se listan los correos que hay, podemos operar con ellos mediante un índice… Cuando se usa POP3 tras leer un correo se elimina, ya que POP3 no está preparado para guardar correos a largo plazo, para eso se usa IMAP.
  - Fase de actualización: Se pasa a esta fase cuando se cierra la conexión, en esta fase es cuando el servidor realmente borra los correos que le hemos indicado, leído…

- Ventajas de IMAP4 frente a POP3
  - Permite organización en carpetas en el lado del servidor (MTA)
  - Mantiene información entre sesiones. Guarda los correos en el servidor.
  - Permite la descarga de partes de los mensajes.
  - Se puede acceder con varios clientes (POP también, pero en modo descargar y guardar).

- Ventajas de Web MAIL frente a POP3
  - Organización total en el servidor, accesible desde cualquier cliente con HTTP. El agente de correo está integrado en el servidor web.
  - Seguridad: uso extendido de HTTPS. Los emails se envían encriptados.

5. Seguridad y protocolos seguros.

- Primitivas de seguridad:
  - Confidencialidad: es lo primero que pensamos cuando hablamos de seguridad, que sólo pueda acceder a la información la persona que queremos que acceda. Para ello, la medida más típica es la criptografía.
  - Responsabilidad:
    - Autenticación: debemos poder identificar a los agentes que participan en la comunicación.

        - No repudio: No se puede negar el autor de una determinada acción.
        - Control de accesos: cuando se accede a una red, para tener cierta seguridad, debemos identificar quién se ha conectado, desde qué dispositivo, etc.

    - Integridad: Detección de alteraciones (intencionalidad o no) de la información.

    - Disponibilidad: Mantener las prestaciones de los servicios con independencia de la demanda.

- <u>Mecanismos de seguridad:</u>
    - Cifrado simétrico: la idea básica del cifrado simétrico es que se usa la misma clave tanto para encriptar como para desencriptar. Dicha clave sólo puede ser sabida por el destinatario y el remitente del mensaje. Los más usados son 3DES y AES. El problema de este sistema es cómo transmitir la clave al destinatario del mensaje sin que ésta sea capturada por una tercera persona.
      C = K(P) & P = K(C)

    - Cifrado asimétrico: Este tipo de cifrado usa una clave distinta para encriptar como para desencriptar. Éste par de claves se generan a partir de un algoritmo y tienen una característica especial: a partir de la clave pública es imposible inferir la privada y viceversa, así, si alguien dispone del texto cifrado y nuestra clave pública no podrá hacer nada. Para poder transmitir un mensaje usando cifrado asimétrico, el destinatario del mensaje debe, en primer lugar, decirnos su clave pública. Así, para hacer una comunicación segura a través de internet, en primer lugar le enviamos al destinatario (usando cifrado asimétrico) la clave simétrica usada para cifrar y a partir de ahí nos comunicamos usando cifrado simétrico que es mucho más rápido que el asimétrico. El secreto de la clave pública y privada es que con un ordenador actual nos llevaría años poder obtener una a partir de la otra.
      $C = K^{+} (P)$ & $P = K^{-} (C)$

    - Funciones Hash: es una función computable mediante un algoritmo tal que: H: U → M. x → h(x) Tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte en un rango de salida finito, normalmente cadenas de longitud fija.

    - Message authentication code (MAC): se usa para evitar que alguien pueda modificar los mensajes que mandamos. No se debe confundir con la dirección MAC de una tarjeta de red. Es una porción de información utilizada para autenticar un mensaje. Los valores MAC se calculan mediante la aplicación de una función hash criptográfica con clave secreta K, que sólo conocen el remitente y destinatario, pero no los atacantes.
      M | H(K | H(K | M)) integridad + autenticación

    - Firma digital: sirve para autenticar la autoría de un documento. Es igual que firmar un documento en papel con nuestra firma, pero digitalmente. El procedimiento para firmar el documento consiste en hacer un hash del documento usando la clave privada, ya que la clave privada es algo que sólo tiene la persona que firma el documento. Usando la clave pública de dicha persona se podría comprobar que efectivamente fue ella la que firmó el documento desencriptando el hash. Sin embargo, no podemos asegurar que la firma digital esté asociada a la identidad de una persona ya que cualquiera puede afirmar ser una persona y en realidad, ser otra muy distinta. Para solucionar esto necesitamos un tercer agente en la comunicación que verifique la identidad de dicha clave pública, este tercer agente se llama entidad certificadora.

- Certificado: que no son más que la asociación entre una identidad y una clave pública. Para poder tener un certificado tenemos que ir a la entidad certificadora con nuestro DNI y nuestra clave pública. La autoridad certificadora hace una firma digital con su clave privada y se comprueba dicho certificado con la clave púbica de la entidad certificadora. Esto se hace automáticamente cuando hacemos accesos por HTTPS, se usa para comprobar que la página es quien dice ser y no estamos siendo víctimas del phishing.
  (ID,K^{+ID}) + K^{-CA} (ID,K^{+ID}) CA: Autoridad de Certificación.

- Seguridad:
  - Seguridad perimetral: regular acceso de usuarios.
    - Física: son las vallas, cámara y todo dispositivo físico que proporciona seguridad.
    - Digital:
      - Firewalls, sistemas de detección de instrucciones (IDS) y de respuesta (IRS) *IDS, no bloquean el tráfico, sino que lo analizan en busca de alguna anomalía basándose en una serie de reglas que tienen definidas y alertan de las anomalías que encuentren. Ej: Snort.
  - Seguridad (criptográfica) en protocolos:
    - Capa de aplicación:
      - Pretty Good Privacy (PGP): se diseñó orientado al correo electrónico pero es casi un mecanismo genérico. Incorpora criptografía simétrica para confidencialidad, asimétrica para intercambio de claves, firma electrónica y certificado.
      - Secure Shell (SSH): es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente.
    - Capa de sesión (entre aplicación y transporte):
      - Transport Secure Layer (TSL): se dice que están en capa de Sesión porque en realidad se encuentran en capa de Aplicación, pero siempre se ponen debajo de otro protocolo de capa de aplicación, por ejemplo HTTPS se basa en poner debajo SSL o TLS y también porque creamos sesiones de comunicación seguras entre dos finales sobre las cuales se puede crear un capa de aplicación encriptada. *Confidencialidad (Ksecreta negocidad) + Autenticación (para el server con Kpública) + Integridad (con MAC).
    - Capa de red:
      - IPSec (VPN): al trabajar en capa de Red, encripta toda la comunicación incluida la capa de red del dispositivo porque lo que hace es un túnel donde encripta toda esta parte (de capa de red hacia arriba) y lo mete en otro paquete de capa de red.

6. Aplicaciones multimedia.

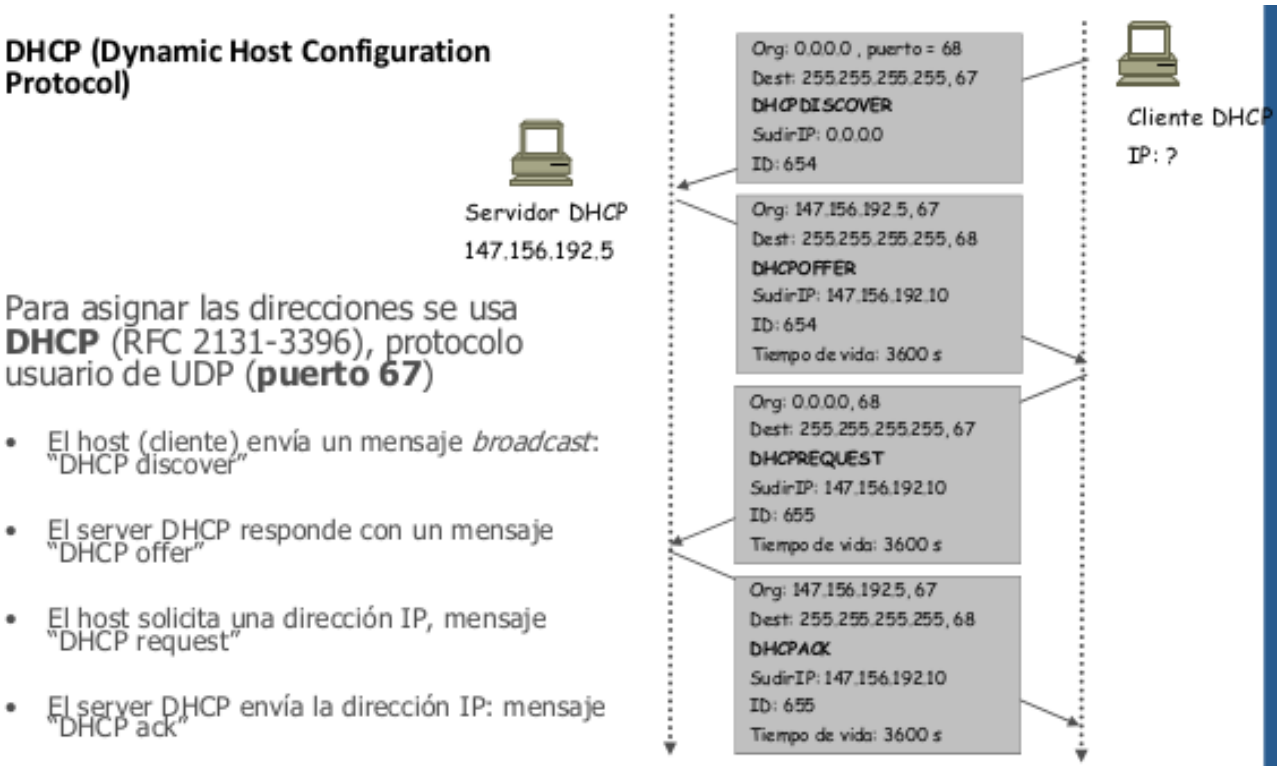
- Conceptos:
  - Aplicaciones multimedia: son aplicaciones que tienen que ver con el audio y el video.
  - Calidad de servicio (QoS): capacidad de ofrecer el rendimiento requerido para una aplicación. Por ejemplo, si se ha podido ver bien un video o no.
- Tipos de aplicaciones:
  - Flujo de audio y vídeo (streaming) almacenado → YouTube. (el delay debe ser mínimo).
  - Flujo de audio y vídeo en vivo → emisoras de radio o IPTV. (se puede permitir un poco de delay).
  - Audio y vídeo interactivo → Skype. (nada de delay ya que podemos perder información).

- Características fundamentales:
  - Elevado ancho de banda.
  - Tolerantes relativamente a la pérdida de datos (que el usuario no se dé cuenta pero que no se le corte el vídeo).
  - Exigen Delay acotado, no hay problema con que tengamos un buffer intermedio en el que ir guardando información, pero si hay tanto delay que en un momento dado el buffer está lleno empieza a haber desconexiones y el video va entrecortado.
  - Exigen Jitter acotado, variación del Delay.
  - Uso de multicast, es una difusión en internet, imitando el comportamiendo de las emisiones de radio o televisión, donde desde una antena emisora se manda la misma señal a todas las antenas, no mandan una señal a cada casa.

7. Aplicaciones para interconectividad de redes locales.

- DHCP (Dynamic Host Configuration Protocol): DHCP se usa para la asignación dinámica de direcciones IP, para así evitar el tener que configurarlas manualmente. Con esto conseguimos que la conexión a una red inalámbrica sea mucho más sencilla: seleccionar la red inalámbrica deseada, introducir la contraseña y ya está. Esto es importante porque la mayoría de los usuarios no tienen suficiente conocimiento técnico como para configurar ellos mismos sus direcciones IP, el servidor DNS, etc.

  DHCP nos da una dirección IP dentro del rango que tengamos configurado, también nos da la dirección IP del DNS primario y secundario y el punto de acceso (normalmente el router) que vamos a utilizar.



El primer paquete emitido por el cliente es un paquete del tipo DHCPDISCOVER.
El servidor responde con un paquete DHCPOFFER, fundamentalmente para enviarle una dirección IP al cliente.
El cliente establece su configuración y luego realiza un DHCPREQUEST para validar su dirección IP (una solicitud de transmisión ya que DHCPOFFER no contiene la dirección IP) .

El servidor simplemente responde con un DHCPACK con la dirección IP para confirmar la asignación. Normalmente, esto es suficiente para que el cliente obtenga una configuración de red efectiva, pero puede tardar más o menos en función de que el cliente acepte o no la dirección IP.

8. Cuestiones y ejercicios.