Tema 2: Servicios y Protocolos de aplicación en Internet

Aclaración

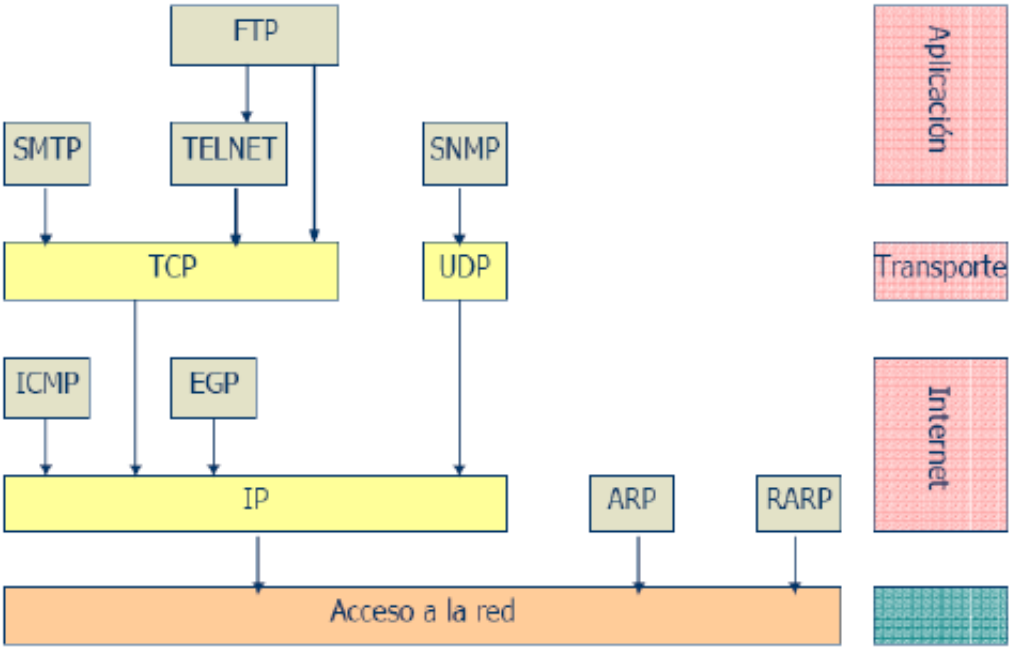
Todo el contenido de estos apuntes viene de las diapositivas de la asignatura ISE del profesor José Camacho Páez http://wdb.ugr.es/~josecamacho/ más las notas que he ido tomando en clase.

Contenido

- Introducción a las aplicaciones de red
- Servicio de Nombres de Dominio (DNS)
- La navegación web
- El Correo Electrónico
- Protocolos seguros
- Aplicaciones multimedia
- Aplicaciones para interconectividad de redes locales

1.Estructura de Protocolos

Protocolos TCP/IP

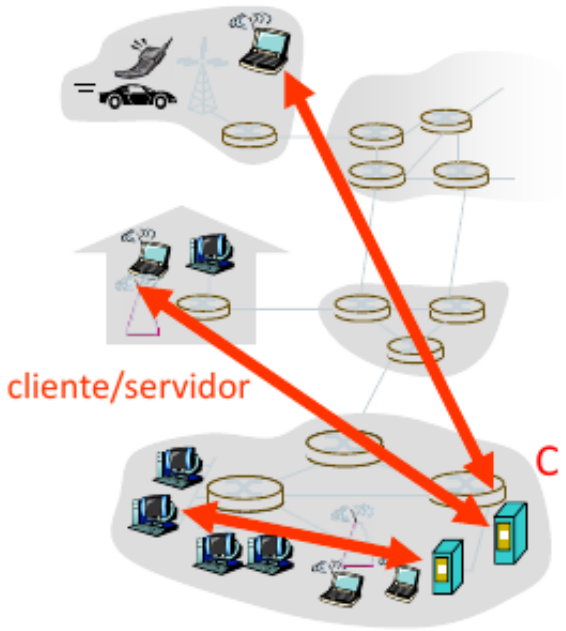


Tenemos protocolos que han sido muy importantes:

- *SMTP*: Correo electrónico
- *FTP*: Archivos
- *TELNET*: terminal a distancia. Es la base de una conexión entre una máquina y otra. Servía para acceso a escritorio remoto.
- *SNMP*: su objetivo es hacer gestión de red
- *TCP*: intenta hacer una conexión fiable
- *UDP*: prácticamente no hace nada
- *IP*: todo se envía sobre este protocolo
- *EGP*: protocolo de encaminamiento
- *ARP y RARP*: para transformar direcciones IP en MAC

Arquitectura Cliente/Servidor

Servidor -> activo todo el tiempo esperando una petición de un cliente. IP en es permanente y pública para que se pueda llegar directamente a él, y no cambie. A veces se agrupan en granjas. *Cliente* -> intermitentes. Pueden tener una IP dinámica o privada. Inicia la comunicación.



Procesos cliente y servidor

Si queremos hacer que un proceso sea accedido por un ordenador necesitamos un **socket** y le tenemos que asignar un puerto. Se suelen asignar el mismo puerto ya que así sería más fácil acceder. Es el puerto 80. Se les denomina puertos bien conocidos. Puedes cambiarlo si quieres pero luego tienes que especificarselo a los clientes.

Retardo en Cola

Tiempo de servicio -> tiempo promedio en darle servicio a un solicitante. También sirve para un router.

El retardo en cola es:

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

¿Qué definen los protocolos de aplicación?

Un protocolo define una serie de tipos de servicios. Dentro de un mismo protocolo hay varios tipos de mensajes: mensaje de solicitud, mensaje de respuesta. El mensaje más gordo es el de respuesta. Las cabeceras tienen que tener una sintaxis, es decir, un formato. Lo que nosotros recibimos en la tarjeta de red es un flujo de bits. Finalmente hay unas reglas, por ejemplo en P2P las reglas son mucho más sofisticadas. Hay distintos tipos:

- *Protocolos dominio público*: nos importan los que están estadarizados. *RFC*

(Request for comments): se crea un documento abierto para que todo Internet pueda comentar sobre él.
- *Protocolos propietario*: de compañías. Ej: Skype, que esconden su documentación para que no se sepa.
- *In-band vs out-of-band*: todos los servicios tienen dos tipos de información asociados: de control y datos. Hay protocolos que en el mismo paquete envían estos dos, y otros que utilizan puertos distintos para cada uno.
- *stateless vs state-full*: los servicios que guardan el estado de un cliente state-full, los que no stateless.
- *persistentes vs no persistentes*: cada cez qye se crea un servicio se envía información, o no.

La tendencia de las nuevas versiones es tener cabeceras fexibles. Tenerlas estáticas era muy cómodo. Las flexibles se basan en una cadena estática y otros trozos(chunks) que se fijan en la cabecera estática. Dentro de los chunks se sigue la estrategia TLU, poníendole un número, luego la longitud del parámetro, y el valor del tamaño que es variable pero ya se sabe su tamaño porque se ha leído anteriormente.

Características

- *Pérdida de datos*: todo lo que tiene que ver con el envío de información importante no debe tener pérdida de datos.Ej: correo electrónico, envío de ficheros…
- *Requisitos temporales*: no debe haber delay. Todo lo interactivo sufre delay.
- *Rendimiento*: tiene que ver con velocidad, no con tiempo.El problema es cuando no hay un rate(velocidad mínima) garantizado.
- *Seguridad*: hay que mantener la seguridad en protocolos. Ej: aquellos que tienen que ver con los pagos en internet.

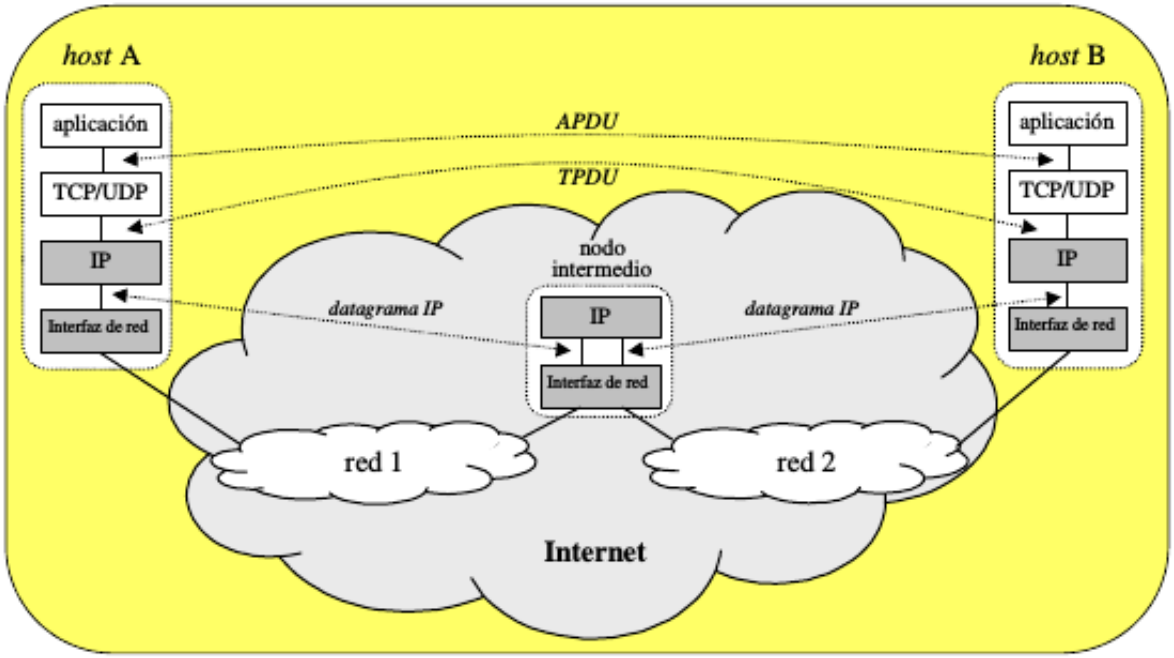
Requerimientos de algunas aplicaciones

Application	Data loss	Throughput	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's ms
stored audio/video	loss-tolerant	same as above	yes, few s
interactive games	loss-tolerant	few kbps up	yes, 100's ms
instant messaging	no loss	elastic	yes and no

Protocolos de transporte

Dependiendo de cuál de las dimensiones queremos optimizar tenemos que usar TCP o UDP. *TCP*-> Los paquetes van a llegar siempre. Aún no se puede garantizar todas las cosas. No se puede decir cuánto va a tardar en responder. Orientado a conexión; Transporte fiable; Control de flujo; Control de congestión. *UDP*-> No orientado a conexión;Transporte no fiable;Sin control de flujo; Sin control de congestión; TCP y UDP(capa de transporte) al ser usuarios del protocolo IP(capa de red) no garantizan:

- Retardo acortado
- Fluctuaciones acotadas
- Mínimo throughput
- Seguridad.En ambos los paquetes van en texto plano



Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP

2.Servicio de Nombres de Dominio (DNS)

DNS-> para traducir nombres a IPs y viceversa. La estructura de nombres de dominio es jerárquica, y las nuevas cosas se van añadiendo a la izquierda.

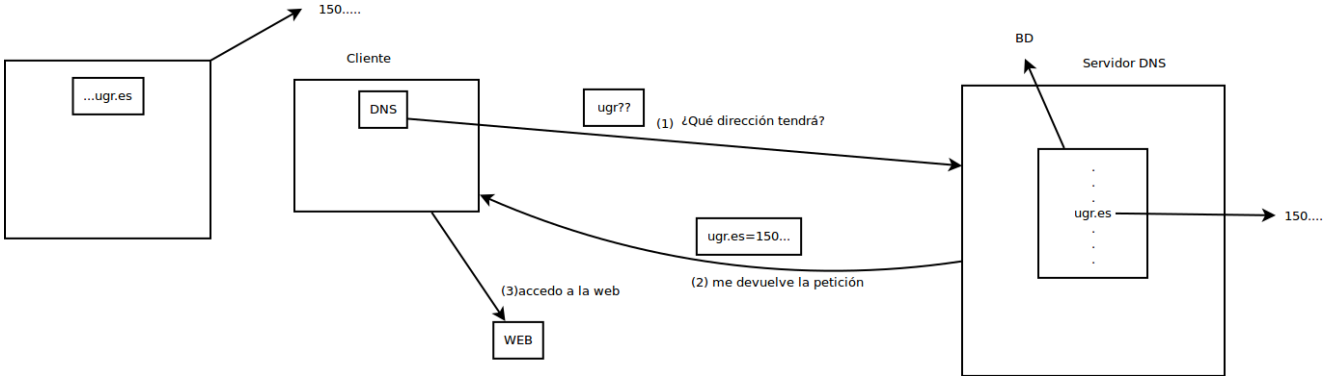
Parte_local.dominio_niveln. … .dominio_nivel2.dominio_nivel1.

ICANN-> se encarga de la solicitud de nombres de dominio. Hay 9 dominios genéricos originales:

- .com -> organizaciones comerciales
- .edu -> instituciones educativas, como universidades, de EEUU
- .gov -> instituciones gubernamentales estadounidenses

- .mil -> grupos militares de estados unidos
- .net -> proveedores de Internet
- .org -> organizaciones diversas diferentes de las anteriores
- .arpa -> propósitos exclusivos de infraestructuras de red
- -int -> organizaciones establecidas por tratados internacionales entre gobiernos
- .xy -> indicativos de la zona geográfica (ej. es(España); pt (portugal)…)

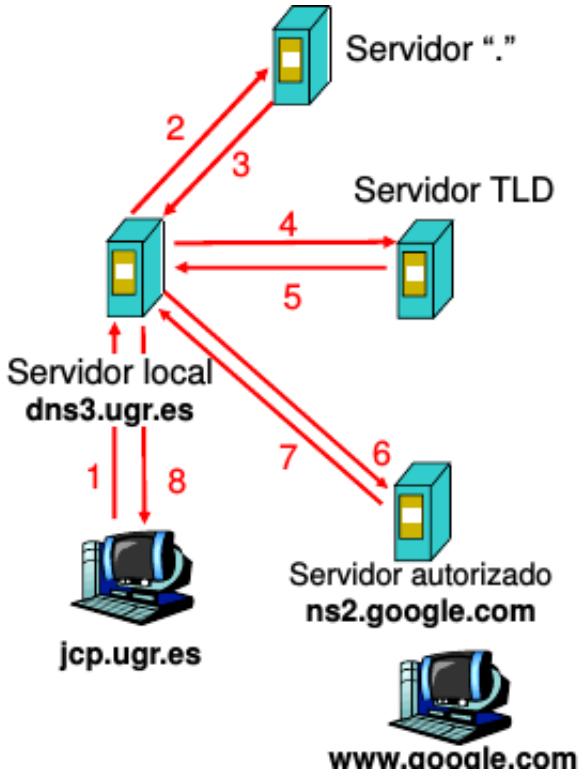
DNS es un servicio cliente-servidor.



Sin el DNS **NO PODEMOS NAVEGAR**. La base de datos del servidor DNS es inmensa, pero nos conviene que sea lo más completa posible y que te responda muy rápidamente. Por lo tanto, necesitamos que haya más de un servidor DNS. Se distribuye también la copia de la base de datos. Y se ponen cerca de donde se van a utilizar, con sentido geográfico.

Tenemos jerarquía en los servidores:

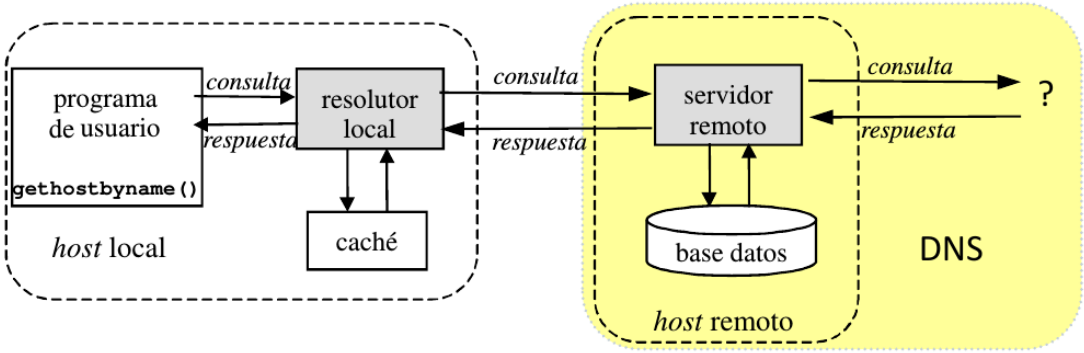
- Servidores "."
- Servidores TLD
- Servidores locales -> para tener la base de datos cerca. Donde más se va a utilizar y actualizar.
- Servidores autorizados -> es el servidor que más manda.

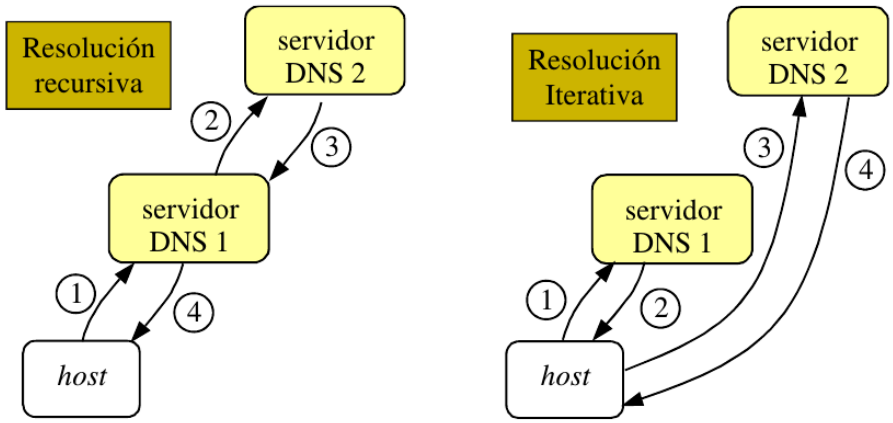


Con respecto a la figura:

- 1. Pedir dirección de Google a un servidor local.
- 1. Una vez hecha la solicitud, el servidor local buscará en su BD o en su caché.
- 1. Si no se encuentra, buscará el servidor "." (el mínimo requisito es que el servidor local tenga en su base de datos las direcciones de los servidores ".")
- 1. Una vez que se le da la dirección del TLD, le pregunta a este servidor
- 1. El servidor TLD le dice la dirección IP del servidor DNS de google.
- 1. Se llama al servidor autorizado que si estará actualizado y este le da la dirección IP correcta.
- 1. Se la envía al servidor local
- 1. Nos la da y accedemos.

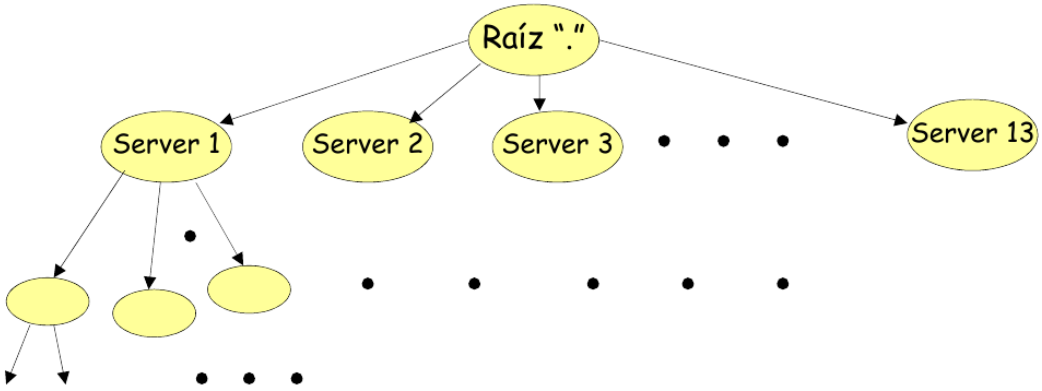
La diferencia entre la resolución recursiva e iterativa no se sabe realmente. Es difícil estimar cuál de las dos va a tardar más.





Gestión de base de datos distribuida y jerárquica

- Está formada por un conjunto de servidores cooperativos que almacenan parcialmente la base de datos.
- 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** (no "cacheada") la información de su zona.
- La autoridad puede **delegarse** jerárquicamente a otros servidores.



Gestión de la base de datos DNS

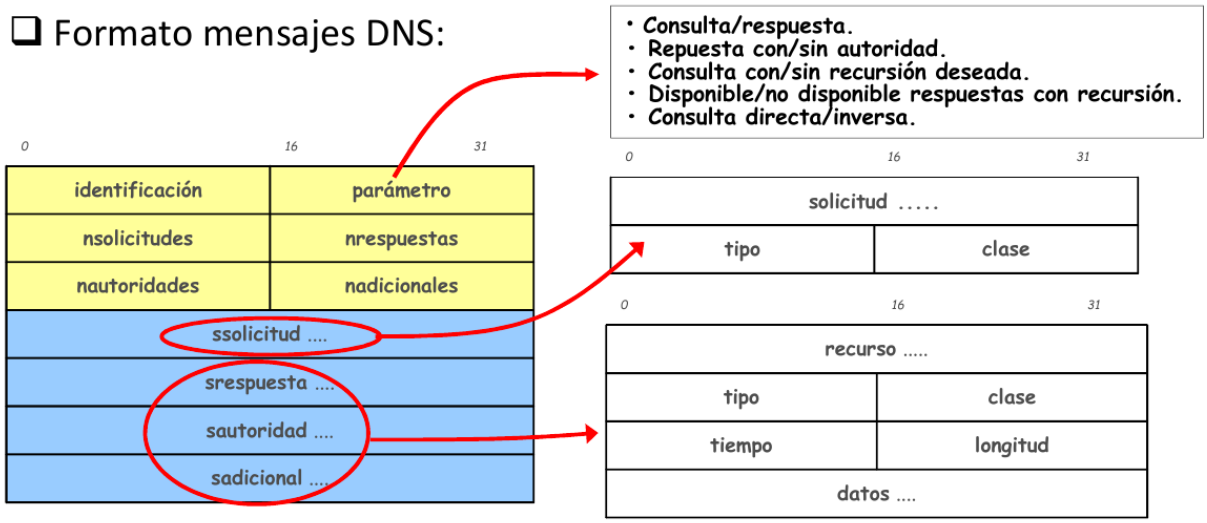
- Cada zona debe tener **al menos** un servidor de autoridad.
- En cada zona hay servidores **primarios** (copia master de la db) y **secundarios** (obtiene la db por transferencia).Los servidores primarios son

los principales y los secundarios están por si estos se caen.
- Además, existe un servicio de **caché** para mejorar las prestaciones.
- La **topología real** de servidores es complicada: existen **13 servidores** raiz.
- Cuando un cliente(resolver) 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 caché.
    - **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 raiz.

¿Cómo es la base de datos DNS?

- Todo dominio está asociado al menos a un registro Resource Record.
- Cada **RR** es una tupla de 5 campos:
    - **Nombre del dominio**: nombre dle dominio al que se refiere el RR.
    - **Tiempo de vida**: tiempo de validez de un registro (para la caché).
    - **Clase**: en Internet simpre IN.
    - **Tipo**: Tipo de registro
        - SOA: Registro de 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 electrónico.
        - 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 **resolucion inversa** para traducir direcciones IP en nombres de dominio.

❑ Formato mensajes DNS:



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

❑ Más información:

- RFC 1034 y RFC 1035 (actualizados 3597 y 3658)
- /usr/doc/HOWTO/trans/es/DNS-COMO
- man named, nslookup, resolver, host.conf, dig
- DNSSEC

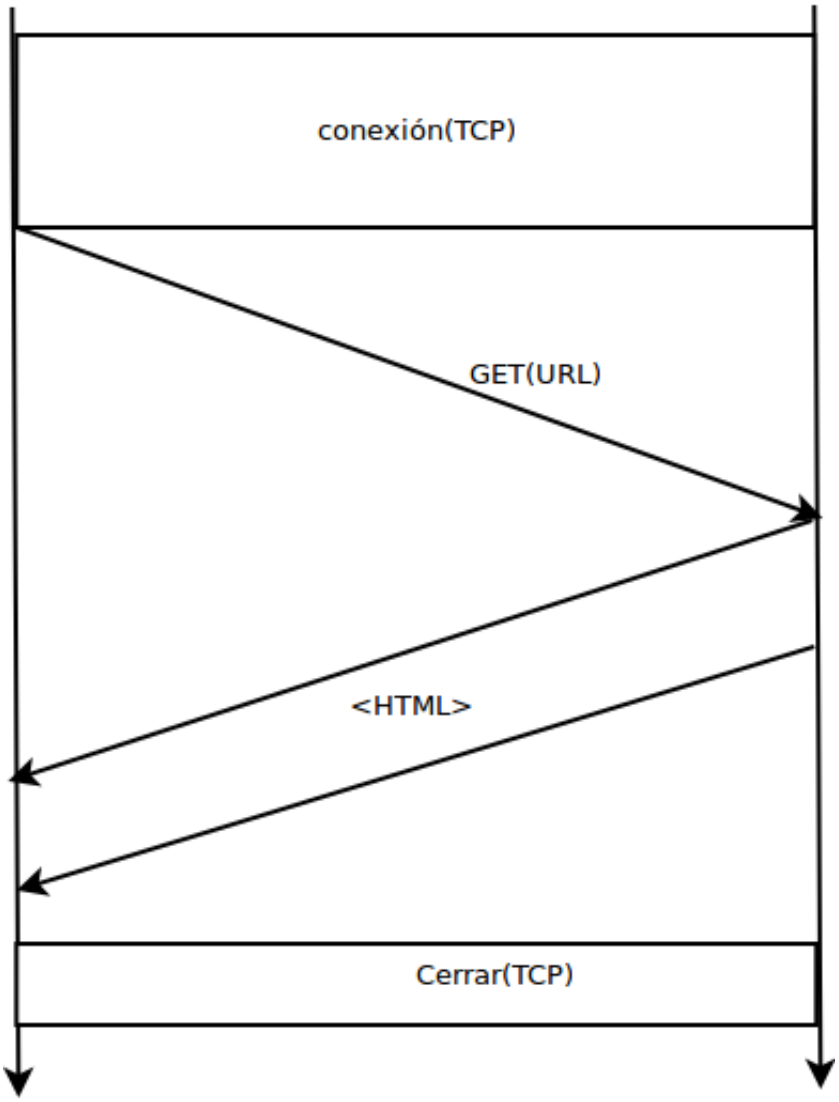
3.Navegación Web

El servicio web es donde pedimos una información y él nos la devuelve de una forma que nuestro navegador lo pueda interpretar. Para acceder a cualquier objeto utilizamos la URL. Lo normal es que el protocolo sea http o https. Primero se pone el servidor,luego el puerto (por defecto el P0) y por último se añade el path.

Características HTTP

Los servidores web no son con estado. Si queremos, tenemos que usar las cookies, que se envían junto a la solicitud. Una conexión no persistente es que abre la conexión, envía un objeto y cierra la conexión.

EJEMPLO: NO persistente

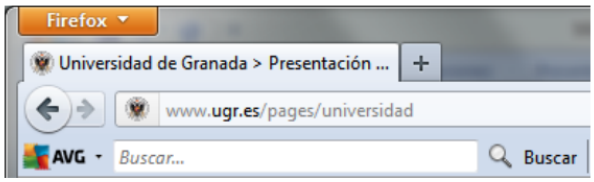


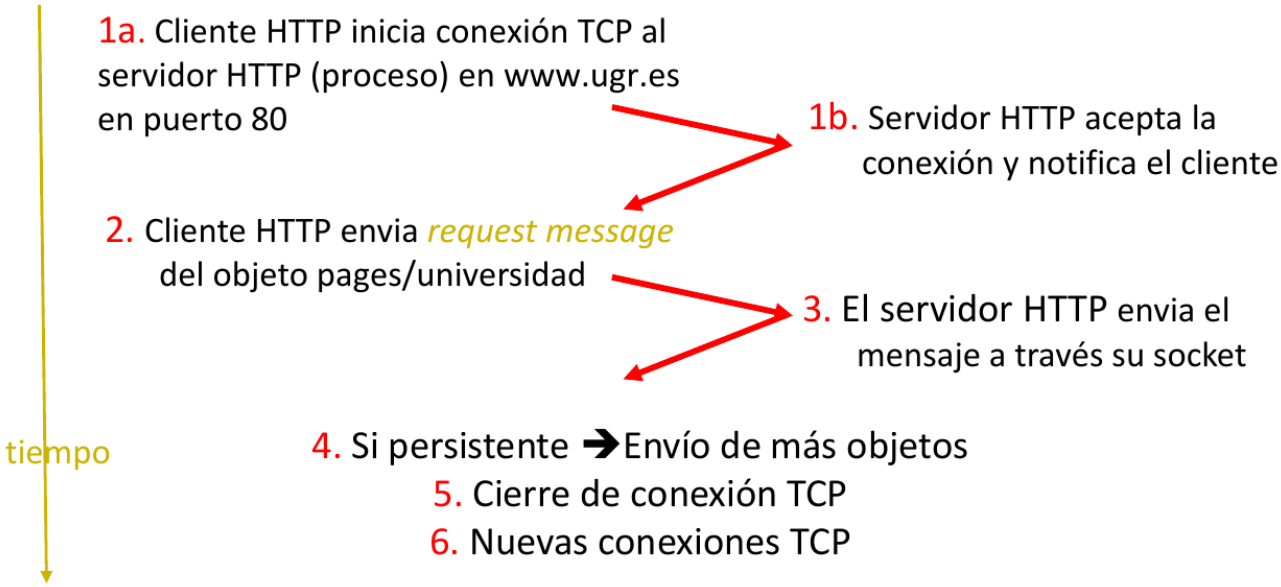
Hay tres rangos de puertos:

- 1-1023 son los puertos **bien conocidos**. Son puertos superusuario, sólo con permisos de administrador.
- 1024-49151 servidicios que no necesitan permisos de superusuario.
- 49152-65535 son **puertos dinámicos** que se dan para el cliente. El SO los libera y se los va dando a los clientes.

Los navegadores hoy en día crean varios sockets en paralelo , lo que puede hacer que se más rápido con conexiones **no persistentes**.

Mensajes HTTP

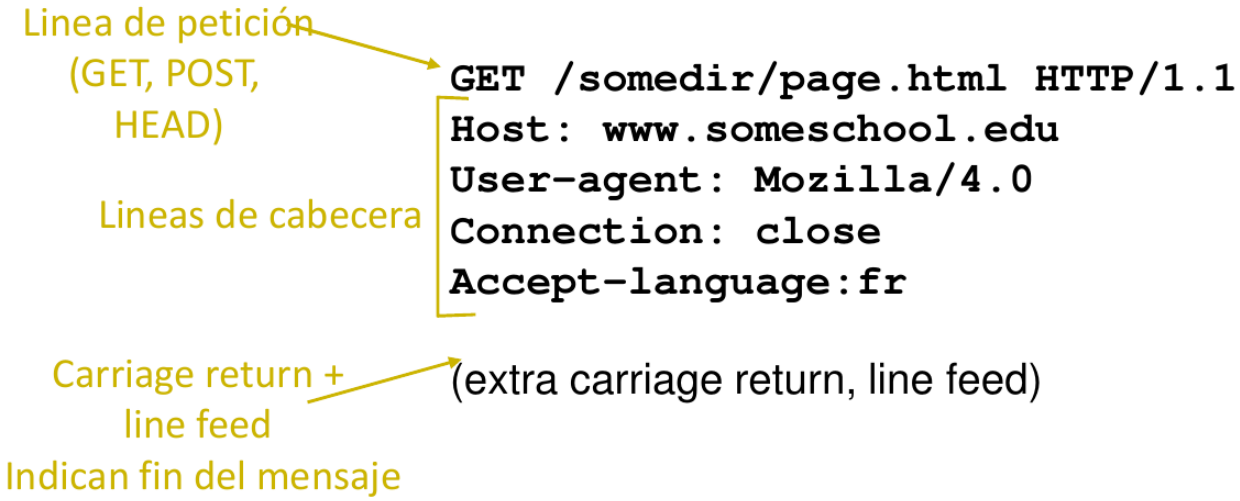




El cliente inicia la conexión. Siempre accede al puerto 80. El servidor acepta o no y responde y ya el cliente envía la solicitud. El servidor contesta con la información y ya se cierra o no dependiendo de si es persistente o no.

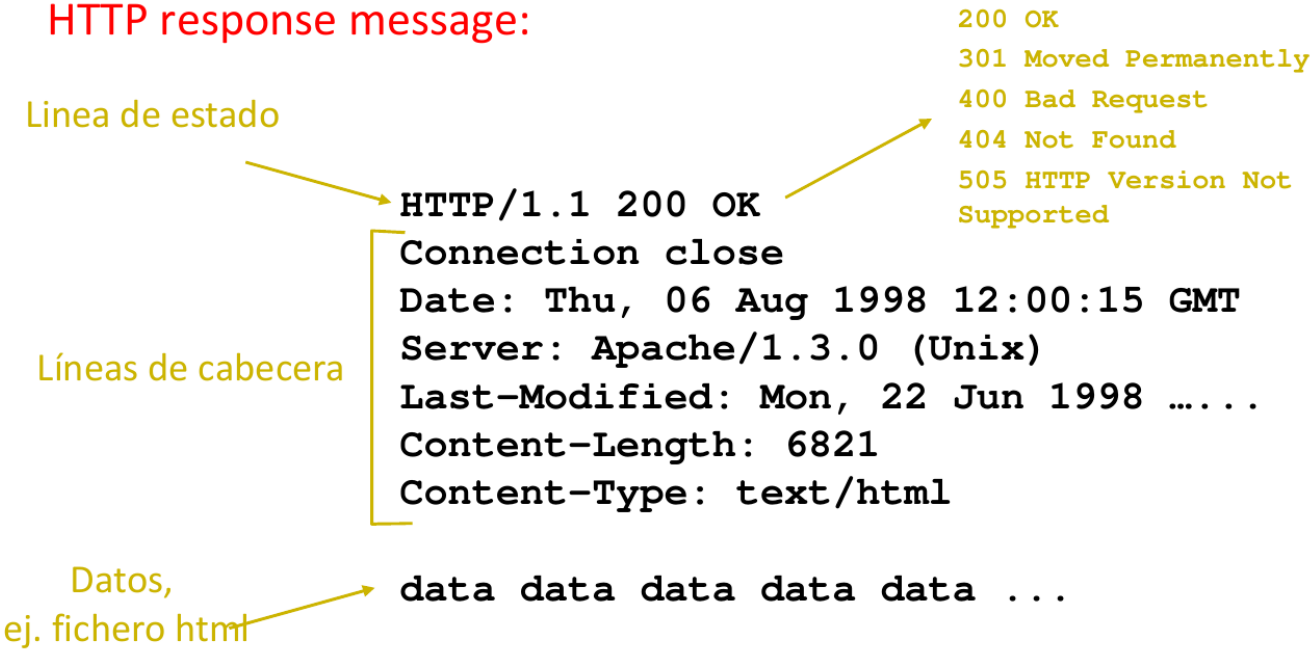
Tipos de mensajes HTTP

get -> solicitud de página. Después se pone el path y luego la versión que entiende el navegador de http. host -> el nombre de host sirve para que si tiene varios nombres se reconozca. user-agent -> el programa del cliente. connection -> tiene que ver con la persistencia. Si pone false es que no tiene que ser persistente.



Dos tipos de mensajes HTTP: *request, response*

HTTP response message:

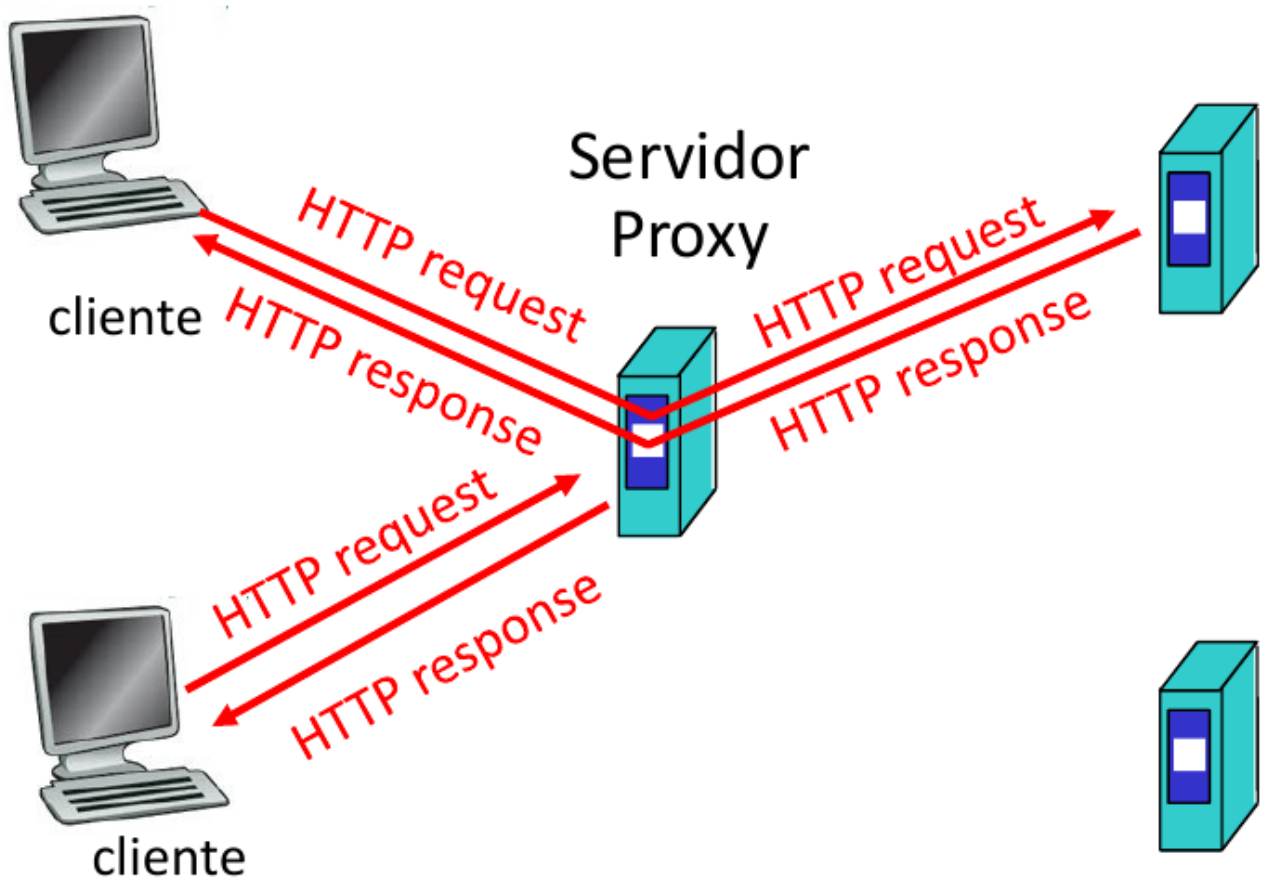


Web Caché

Tenerla bien configurada es muy importante.

- Servidor proxy -> algún dispositivo por donde pasan todas las conexiones de la red

El utiliza el proxy relentiza un poco la conexión, pero luego nos salva en ancho de banda.



La caché manda la solicitud al servidor y le pregunta si ha sido modificada desde la última fecha. Si sí, envía todos los datosm si no solo envía un paquete de control.

4. El correo electrónico

Tiene 4 componentes principales:

- Cliente de correo(user agent)
- Servidor de correo (mail server o mail transfer agent)
- Simple Mail Transfer Protocol: SMTP
- Protocolos de descarga: POP3, IMAP, HTTP

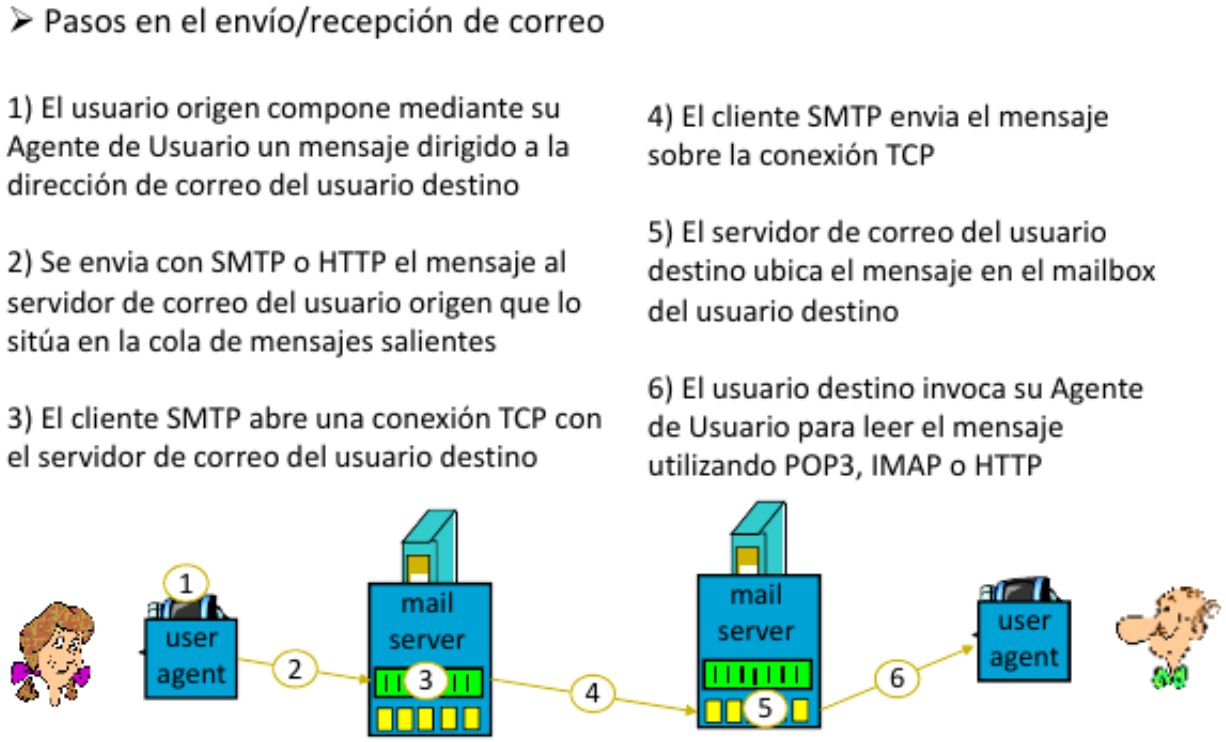
Por cada origen y destino hay un cliente y un servidor. Hay distintos procesos de comunicación: subida de correo, transmisión de correo, descarga de correo…

SMTP

Tu envías el correo electrónico pero la otra persona no tiene porque estar mirando. El modelo cliente servidor qe hemos visto antes no nos vale, ya que el otro

ordenador no tiene porque estar siempre escuchando. Hay que tener un sitio(backup intermedio) donde se guarden los correos y que se descarguen cuando quieras miralos. El servidor web tiene dos colas:

- **La de salida**: donde están los mensajes enviados. Cada cierto tiempo se libera la cola enviando los correos a su destino.
- **La de llegada**: cuando el quiere acceder a su correo se envían los que hay en esta cola.



Ventajas de IMAP

- Permite la organización en carpetas en el lado del servidor(MTA)
- Para ello, mantiene información entre sesiones.
- Permite la descarga por partes de los mensajes.
- Posible acceder con varios clientes (POP también, pero en modo descargar y guardar).

Ventajas de Web Mail

- Organización total en el servidor, accesible desde cualquir cliente con HTTP.
- Seguridad: uso extendido de HTTP.

➢Listado de puertos relacionados con e-mail:

POP3 - port 110
IMAP - port 143
SMTP - port 25
HTTP - port 80
Secure SMTP (SSMTP) - port 465
Secure IMAP (IMAP4-SSL) - port 585
IMAP4 over SSL (IMAPS) - port 993
Secure POP3 (SSL-POP) - port 995

5. Seguridad y protocolos seguros.

Protocolos seguros

- *Confidencialidad* -> que nadie vea lo que hago. La medida típica es ka encriptación.
- *Responsabilidad* -> poder identificar a los agentes que intervienen en una comunicación. No repudio, que no puedas negar haber hecho algo en la red. Tiene que ver con temas bancarios.
- *Integridad* -> que no se haya modificado. Ni que nadie haya modificado una comunicación. Está muy ligado a la confidencialidad.
- *Disponibilidad* -> está muy generalizado. Ahora cobra especial importancia por las infraestructuras críticas (agua, electricidad…)

Mecanismos de seguridad

- *Protocolos seguros* -> como mantener las comunicaciones garantizando todo lo que hemos descrito arriba.

Cifrar es cambiar lo que dice un mensaje de manera que lo podamos recuperar. Lo importante dentro del cifrado es la clave.

- *Cifrado simétricp* -> se utiliza la misma clave para cifrar que para descifrar. Ej: DES, 3DES,AES,RC4. El principal problema es que tienes que pasarle la clave. Por ello surge el cifrado asimétrico.
- *Cifrado asimétrico* -> Yo genero dos claves. si yo cifro con la clave positiva

puedo recuperar el texto con la clave negativa. Si alguien tiene la clave positiva no es capaz de de recuperar la clave negativa.Uno de ellos genera la clave privada y la clave pública. Este individuo le pasa al otro la clave pública, y el otro cifra el mensaje con la clave pública. Y cuando llega al otro lo descrifra con la clave privada. Antes, cuando ya tiene la clave pública, encripta su clave de cifrado simétrico, y se la envía, para que descrifre los mensajes, ya que es mucho más rápido.

- *Message Authentication* -> es un **hash**(obtener un identificador unívoco de algo, en principio si está bien hecho, si yo cambio algo, el hash cambia) y lo combina con criptografía. Al mensaje se le hace una operación de hash y se lo añade al final del mensaje, pero antes encripto este hash.
- *Firma digital* -> sirve para autenticar lo anterior de un determinado documento. Hago un MAC pero con clave privada. Uso de la criptografía asimétrica pero al revés. Hago un hash del documento usando mi clave privada. Ya que con mi clave pçublica qu tenga la gente, pueda comprobar que es mía. Hay un agujero ya que puedo engañar diciendo que soy alghuien que no soy y les doy mi clave pública.
- *Autoridad certificadora* -> alguien en quien confiamos para comunicarnos. La autoridad hace una firma digital con la clave pública de la persona. Y es lo que se pone en el certificado. Esto se hace automáticamente en https.
- *Certificado* -> asocia identidad de persona y clave pública y viene firmado por la autoridad certificadora.
- *Seguridad perimetral* -> regular el acceso de comunicaciones a los servidores de una red. Ej: firewall, sistemas de detección de intrusiones… kiKis firewall de redes bloquean aquellas cosas que no tienen lugar.

Tenemos distintos tipos de protocolos dependiendo de en que capa estoy. Un protocolo que encripta capa de aplicación solo encripta la parte de aplicación, mientras que por ejemplo el WIFI encripta la capa de red, la de transporte y la de aplicación.

- *Capa de aplicación* -> ssh, PGP
- *Capa se sesión* -> capa de aplicación debajo de otro protocolo de capa de aplicación.

6.Aplicaciones multimedia

Conceptos

Son aplicaciones que tienen que ver con el audio y el video. El término calidad de

servicio toma especial relevancia. Se trabaja siempre para que tenga clidad en un entorno muy negativo como es Internet, ya que no podemos asegurar que hay un throwput máximo ni nada. Internet no aporta ninguna garantía de calidad de servicio.

Tenemos que deficiniar tres tipos de aplicaciones:

- *Flujo de audio y video almacenado* -> no es tan importante el delay, ya que existe un buffer intermedio.
- *Flujo de audio y vídeo en vivo* -> aquí el delay es muy importante pero aun así podemos tener un buffer.
- *Audio y vídeo interactivo* -> aquí es muy importante el delay.

Características principales

- *Tolerantes a la pérdida de datos* -> se puede perder un frame.
- *Delay acortado* -> no puede quedarse el buffer sin espacio.
- *Jitter acotado* -> jitter(variación de delay) un vídeo necesita que el delay sea más o menos constante.
- *Uso de multicast* -> sirve para que existan destinos múltiples. Es una especie de difusión, pero en Internet. Solo se envía un flujo de datos, y les voy creando la conexión según la voy necesitando. Necesito un servicio de subscripción y le voy dando según la zona.

7. Aplicaciones para interconectividad de redes locales

- *DHCP* -> asignación dinámica de IP. Evitar hacer una configuración manual de IP. No se usa solo con direccionamiento privado, lo utilizan también los ISP, para minimizar el número de IPs que se utilizan, además de que cuestan finero por lo tanto gastas menos dinero.

Una tarjeta de red de solo pasar los paquetes qye ke ubtereseiab a ese ordenador, salvo en las direcciones broadcast. Si el paquete es broadcast todos lo abren para ver su contenido. Se utiliza cuando no sabemos la IP del destino.

- *DHCPOFFER* -> es con lo que se responde a la petición del cliente DHCP.

En la línea de tiempo se añaden dos paquetes más (siendo todos broadcast) para que si hay otros servidores se sepa que ya no tiene que darle una IP, y se mandan como broadcast para que se sepa que el cliente ha requerido al otro.

El identificador sirve para solventar problemas cuando hay varios clientes a la vez queriendo la IP.

- *DunDNS* -> servicios que permiten introducir detrás de un router la instalación de servidores que son accesibles desde fuera pero no al revés. Estos sirven para que si nuestra IP cambia cambiemos nuestros servicio siga en activo por un nombre de dominio.

Configuración de un cliente Linux (Fedora Core dist.):

```
# Sample /etc/sysconfig/network-scripts/ifcfg-eth0 :

DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:0C:29:CE:63:E3
ONBOOT=yes
TYPE=Ethernet
```

Configuración de un servidor de Linux (*dhcpd*):

```
# Sample /etc/dhcpd.conf

default-lease-time 600;max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "mydomain.org";
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.10 192.168.1.100;
  range 192.168.1.150 192.168.1.200;
}

# Static IP address assignment
host haagen {
  hardware ethernet 08:00:2b:4c:59:23;
  fixed-address 192.168.1.222;
}
```

- *UPnP(Universal Plug and Play)* -> servicios se comunican con este protocolo para comunicarle el cambio de IP. Te saltas la parte de nombre de dominio