

TEMA 5

SERVICIOS Y PROTOCOLOS

DE APLICACIÓN EN INTERNET

Fundamentos de Redes

2021/2022



ugr
Universidad
de Granada

➤ Bibliografía Básica:



Capítulo 2 (2.1, 2.2, 2.3, 2.4), James F. Kurose y Keith W. Ross. **COMPUTER NETWORKING. A TOP-DOWN APPROACH**, 7^a Edición, Addison-Wesley, 2017, ISBN: 9780133594140.



Capítulo 11, Pedro García Teodoro, Jesús Díaz Verdejo y Juan Manuel López Soler.
TRANSMISIÓN DE DATOS Y REDES DE COMPUTADORES, Ed. Pearson, 2017, ISBN: 978-0-273-76896-8.

➤ Para saber más...



Multimedia Networking Capítulo 9, James F. Kurose y Keith W. Ross. **COMPUTER NETWORKING. A TOP-DOWN APPROACH**, 7^a Edición, Addison-Wesley, 2017, ISBN: 9780133594140.

➤ Agradecimientos:

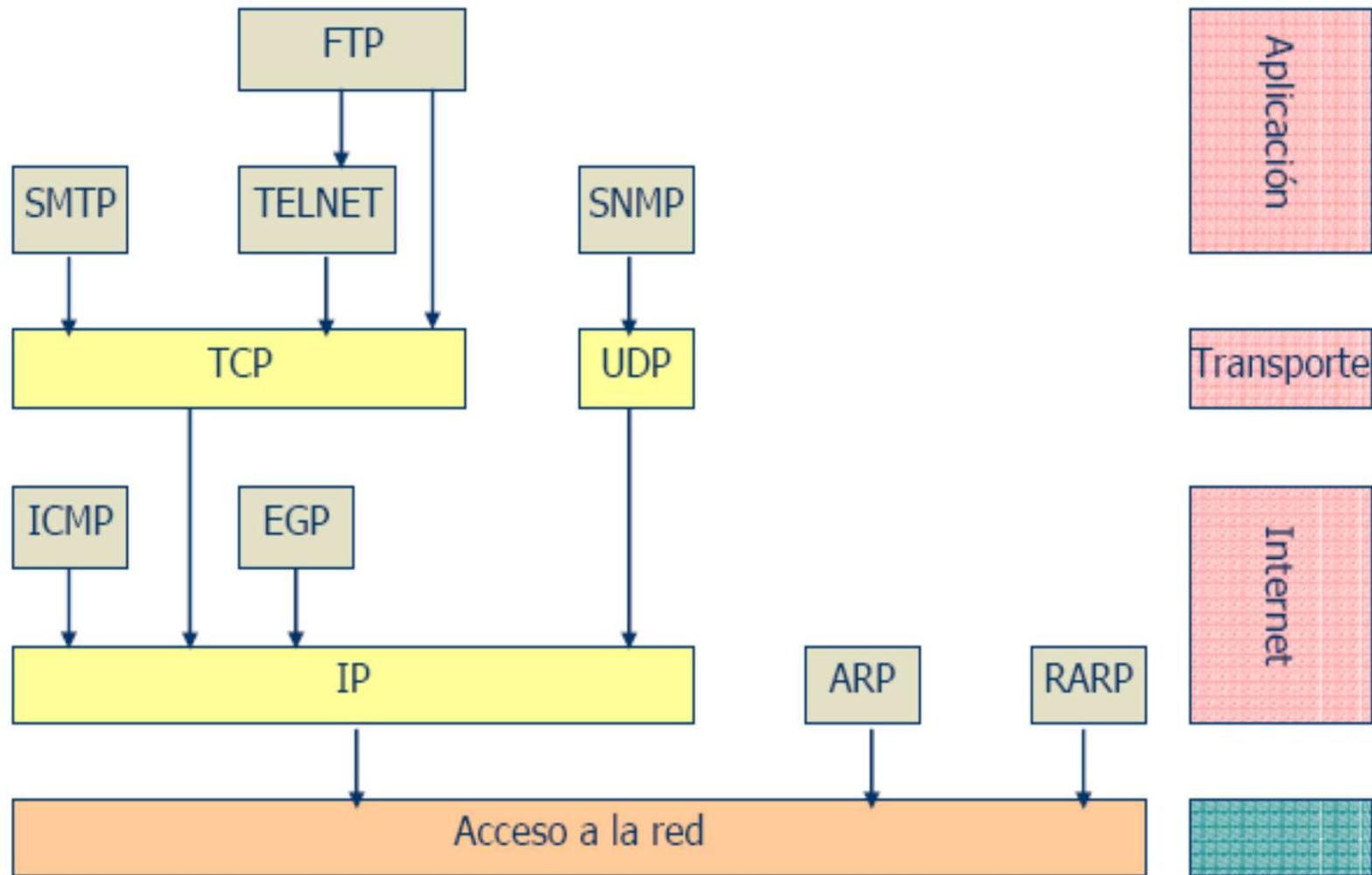
Estas transparencias están inspiradas en las transparencias utilizadas por Kurose y Ross en la Universidad de Massachusetts.

Tema 5.

SERVICIOS Y PROTOCOLOS DE APLICACIÓN EN INTERNET

- 1. Introducción a las aplicaciones de red**
2. Servicio de Nombres de Dominio (DNS)
3. La navegación Web (HTTP)
4. El Correo electrónico (SMTP/IMAP/POP3)
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

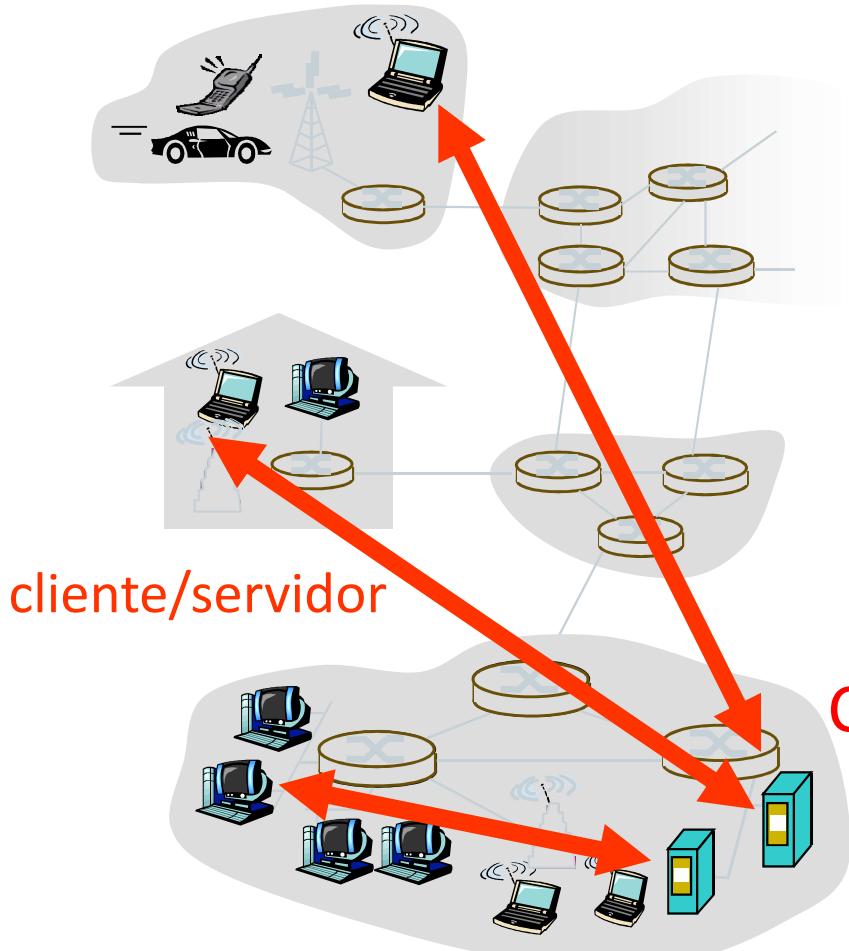
INTRODUCCIÓN A LAS APLICACIONES DE RED: PROTOCOLOS TCP/IP



INTRODUCCIÓN A LAS APLICACIONES DE RED: INTERACCIÓN CLIENTE/SERVIDOR

Servidor:

- Siempre en funcionamiento
- IP permanente & pública
- Agrupados en “granjas”
- <http://www.xatakandroid.com/mundo-android/la-imagen-de-la-semana-google-muestra-el-corazon-de-internet>
- <https://www.youtube.com/watch?v=zRwPSFpLX8I>



Clientes:

- Funcionando intermitentemente
- Pueden tener IP dinámica & privada
- Se comunican con el servidor
- No se comunican entre sí

INTRODUCCIÓN A LAS APLICACIONES DE RED: INTERFAZ SOCKET

Proceso Cliente: proceso que inicia la comunicación

Proceso Servidor: proceso que espera a ser contactado

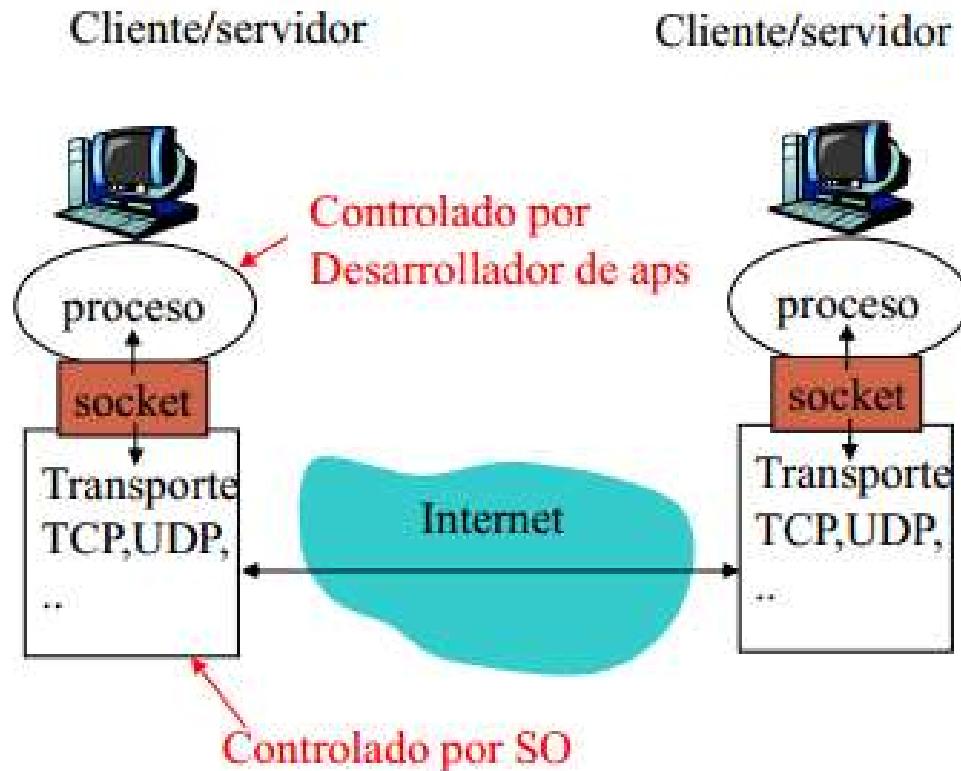
→ IP permanente & pública

- Proceso envía/recibe mensajes a/desde su **socket**
- Para recibir mensajes un proceso debe tener un **identificador** (IP + puerto)

Ej: servidor web **gaia.cs.umass.edu**:

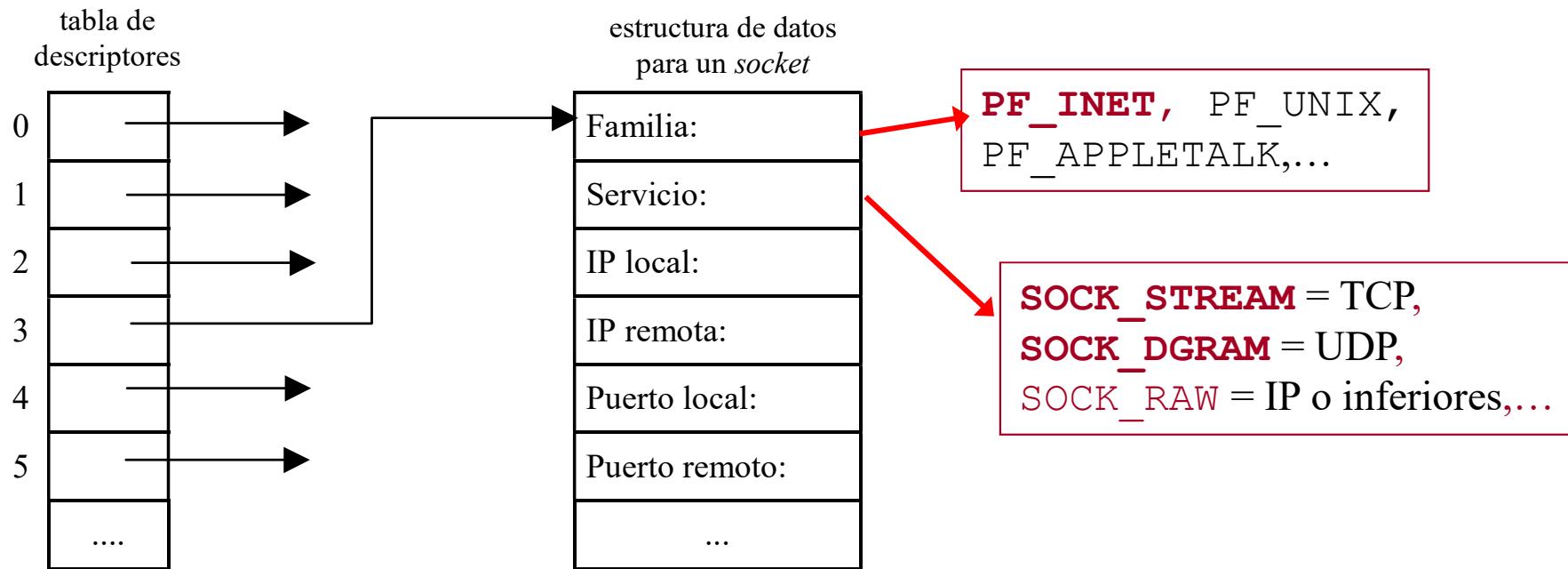
Dirección IP: **128.119.245.12**

Número de puerto: **80**



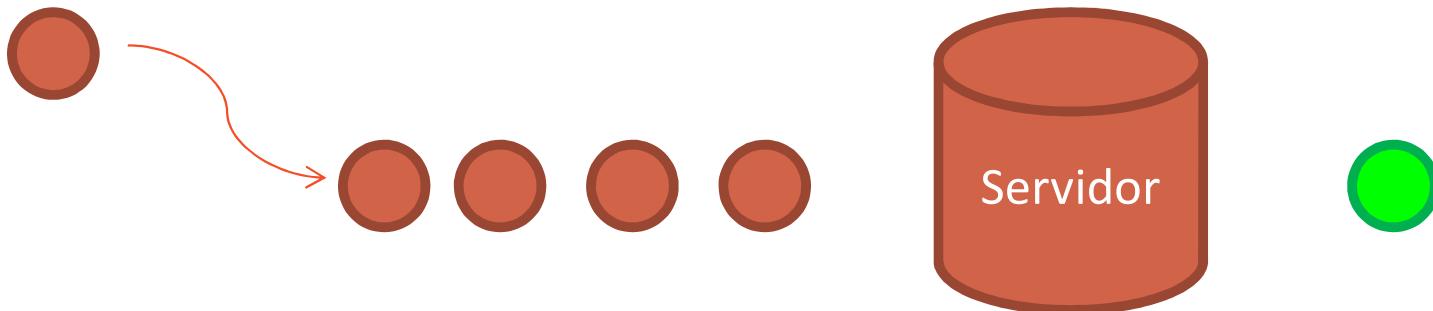
INTRODUCCIÓN A LAS APLICACIONES EN RED: LA INTERFAZ SOCKET

- Definimos **SOCKET** como un **descriptor** de una transmisión a través del cual la aplicación puede enviar y/o recibir información hacia y/o desde otro proceso de aplicación.
- Es una “**puerta**” de acceso (metafóricamente) entre la **aplicación** y los servicios de **transporte**.
- En la práctica un **socket** es una variable tipo **puntero** a una estructura:



INTRODUCCIÓN A LAS APLICACIONES DE RED: RETARDO EN COLA

- Para estimar los retardos (tiempos) en cola DE UN SERVIDOR se usa la **teoría de colas**.
- El uso de un servidor se modela con un sistema M/M/1 (para más información ver el apéndice Capítulo 3 del libro TRANSMISIÓN DE DATOS Y REDES DE COMPUTADORES)



- El retardo en cola es:
- $$R = \frac{\lambda \cdot (T_s)^2}{1 - \lambda \cdot T_s}$$
- donde T_s (distribución Exponencial) es el valor esperado del tiempo de servicio y λ (distribución de Poisson) es el valor esperado de la tasa de llegada de solicitudes.
- Esta misma expresión se puede utilizar para calcular el retardo en cola en un router.

INTRODUCCIÓN A LAS APLICACIONES DE RED: ¿QUÉ DEFINEN LOS PROTOCOLOS DE APLICACIÓN?

➤ ¿Qué es y qué define un protocolo?

➤ El tipo de servicio

- Orientado o no orientado a conexión
- Realimentado (confirmado) o no

➤ El tipo de mensaje

ej., *request* (solicitud) o *response* (respuesta)

➤ La sintaxis:

Definición y estructura de “campos” en el mensaje.
 Hay protocolos orientados a texto (HTTP)
 frente a no orientados a texto (en binario) (DNS)
 Tendencia : usar formato Type-Length-Value

➤ La semántica:

Significado de los “campos”

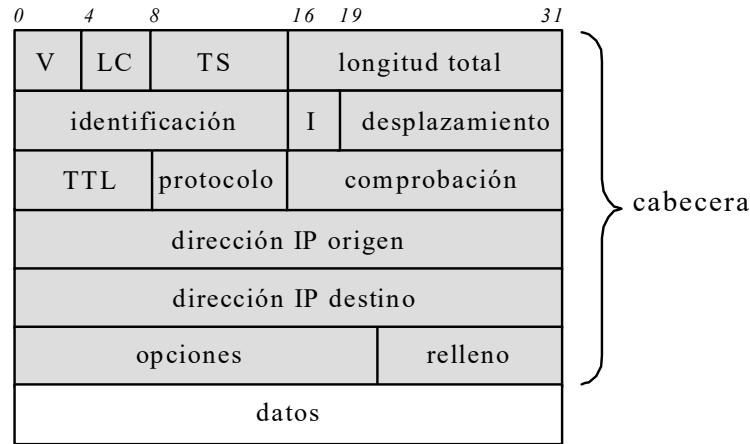
➤ Las reglas:

Cuándo y para qué las entidades envian o responden mensajes

➤ Tipos de protocolos:

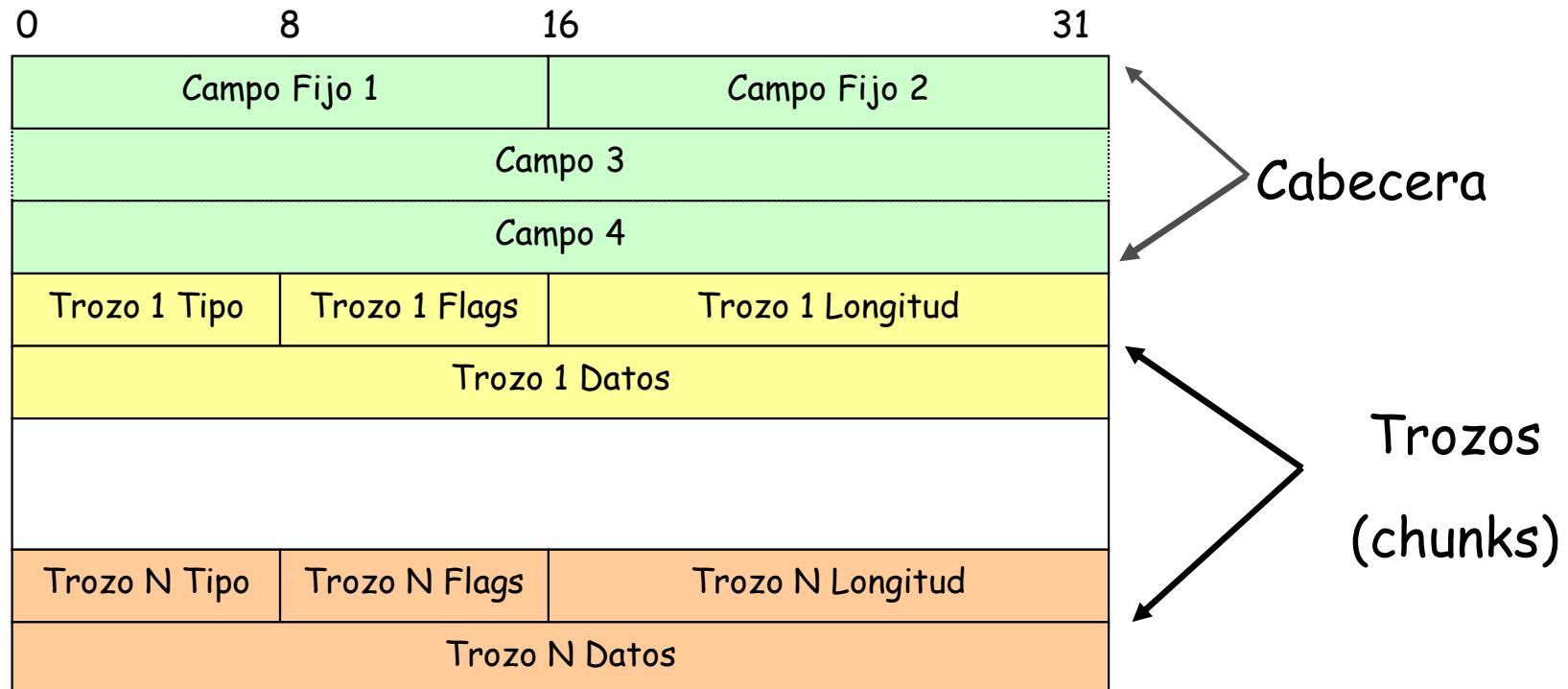
- Protocolos de dominio público (Definidos en RFCs (ej., HTTP, SMTP)) **versus** propietarios → (ej., Skype, IGRP)
- Protocolos in-band **versus** out-of-band
- Protocolos stateless **versus** state-full
- Protocolos persistentes **versus** no-persistentes (sobre servicios SOC)

Ejemplo: protocolo IP



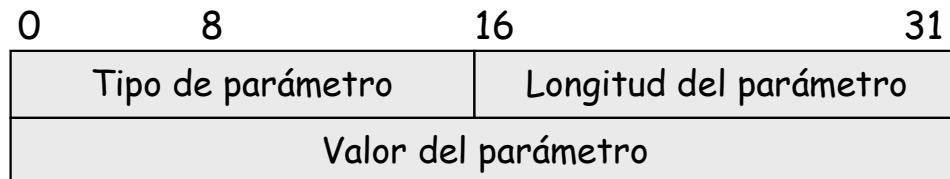
INTRODUCCIÓN A LAS APLICACIONES DE RED: ¿QUÉ DEFINEN LOS PROTOCOLOS DE APLICACIÓN?

- Tendencia: hacer los protocolos **flexibles** con
 - Una cabecera fija
 - Una serie de “trozos” (obligatorios y opcionales)



INTRODUCCIÓN A LAS APLICACIONES DE RED: ¿QUÉ DEFINEN LOS PROTOCOLOS DE APLICACIÓN?

- Tendencia: hacer los protocolos **flexibles** con:
 - Una cabecera fija
 - Una serie de “trozos” (**obligatorios y opcionales**)
 - Los trozos pueden incluir una cabecera específica más una serie de datos en forma de parámetros:
 - Parámetros fijos: en orden
 - Parámetros de longitud variable u opcionales.
 - Para los parámetros se usa Formato TLV (*Type-Length-Variable*)



INTRODUCCIÓN A LAS APLICACIONES DE RED: CARACTERÍSTICAS

Características/requisitos de las aplicaciones:

Tolerancia a pérdidas de datos (errores): Algunas *apps* (ej., audio) pueden tolerar algunas pérdida de datos; otras (ej. FTP, telnet, HTTP) requieren transferencia 100% fiable (sin errores ni pérdidas).

Exigencia de requisitos temporales: Algunas *apps* denominadas *inelásticas* (ej., telefonía Internet, juegos interactivos) requieren retardo (*latencia*) acotado para ser efectivas, otras aplicaciones no.

Demanda de ancho de banda (tasa de transmisión o *throughput*): Algunas *apps* requieren envío de datos a una tasa determinada (p. ejemplo un *codec* de vídeo), otras no

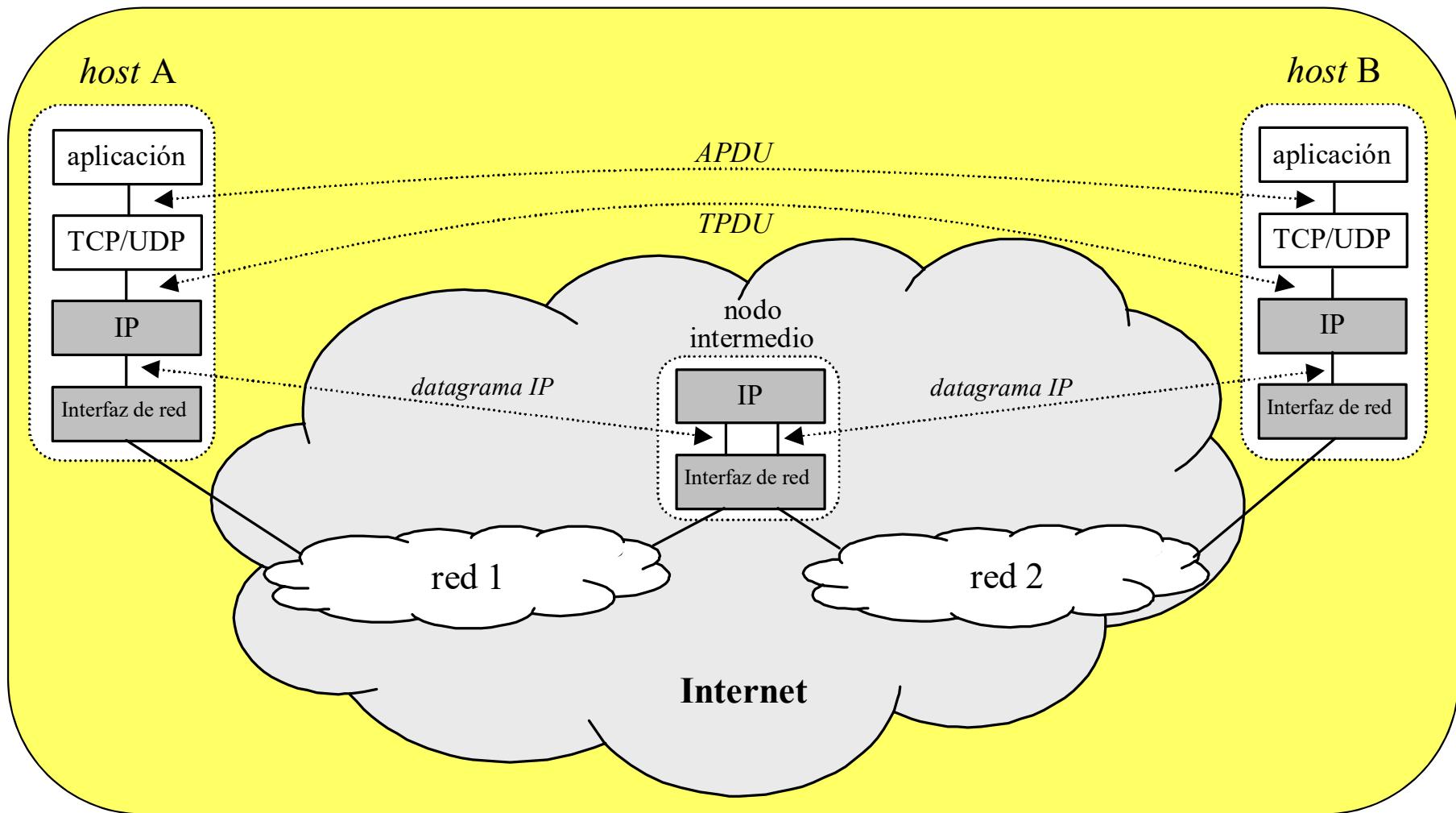
Nivel de seguridad: Los requisitos de seguridad para las distintas *apps* son muy variables (Encriptación, autenticación, no repudio, integridad...)

Conclusión: las distintas aplicaciones tienen requisitos HETEROGÉNEOS

INTRODUCCIÓN A LAS APLICACIONES DE RED: 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

INTRODUCCIÓN A LAS APLICACIONES DE RED: PROTOCOLOS DE TRANSPORTE



Servicio TCP:

Orientado a conexión

Transporte fiable con control de errores

Control de flujo

Control de congestión

Servicio UDP:

No orientado a conexión

Transporte no fiable

Sin control de flujo

Sin control de congestión,

¿Para qué existe UDP?

- TCP y UDP (capa de transporte) al ser usuarios del protocolo IP (capa de red) **no garantizan Calidad de Servicio (QoS)**, es decir:
 - El retardo NO está acotado
 - Las fluctuaciones en el retardo NO están acotadas
 - No hay una velocidad de transmission mínima garantizada
 - No hay una probabilidad de pérdidas acotada
- Tampoco hay garantías de seguridad.

INTRODUCCIÓN A LAS APLICACIONES DE RED

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

Tema 5.

SERVICIOS Y PROTOCOLOS DE APLICACIÓN EN INTERNET

1. Introducción a las aplicaciones de red
- 2. Servicio de Nombres de Dominio (DNS)**
3. La navegación Web (HTTP)
4. El Correo electrónico (SMTP/IMAP/POP3)
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

SERVICIO DE NOMBRES DE DOMINIO (DNS)

- Internet, en la capa de red para realizar el encaminamiento necesita las **direcciones IP**
- Los usuarios prefieren usar “nombres de dominio” (más de 300×10^6)
- Domain Name System (DNS): básicamente realiza la traducción de nombres de dominio a direcciones IP (resolución de nombres)

www.ugr.es <-----> **150.214.204.25**

- El espacio de nombres de dominio tiene una estructura jerárquica en forma de árbol centrada en el dominio raíz (root) “.”:
Parte_local.dominio_niveln.dominio_nivel2.dominio_nivel1.
- A los dominios de nivel1 se les denomina **Top Level Domians (TLD)** (.com .es .edu etc).
- El dominio raíz (“.”) está gestionado por el **ICANN** (Internet Corporation for Assigned Names and Numbers; <http://www.icann.org>). ICANN delega la gestión de algunos dominios TLD a centros regionales.



SERVICIO DE NOMBRES DE DOMINIO (DNS)

Lecturas complementarias recomendadas

- Tutorial sobre “los nombres de dominios”:

<https://www.icann.org/en/system/files/files/domain-names-beginners-guide-06dec10-es.pdf>

- Instrucciones para registrar un nombre de dominio en .es:

<http://www.dominios.es/dominios/es/todo-lo-que-necesitas-saber/sobre-registros-de-dominios>

- Instalación y ejemplos de ficheros configuración de *named*

<https://www.tldp.org/HOWTO/DNS-HOWTO.html>

- Libro “Pro DNS and BIND 10” Ron Aitchison:

<https://learning.oreilly.com/library/view/pro-dns-and/9781430230489/>

SERVICIO DE NOMBRES DE DOMINIO (DNS)

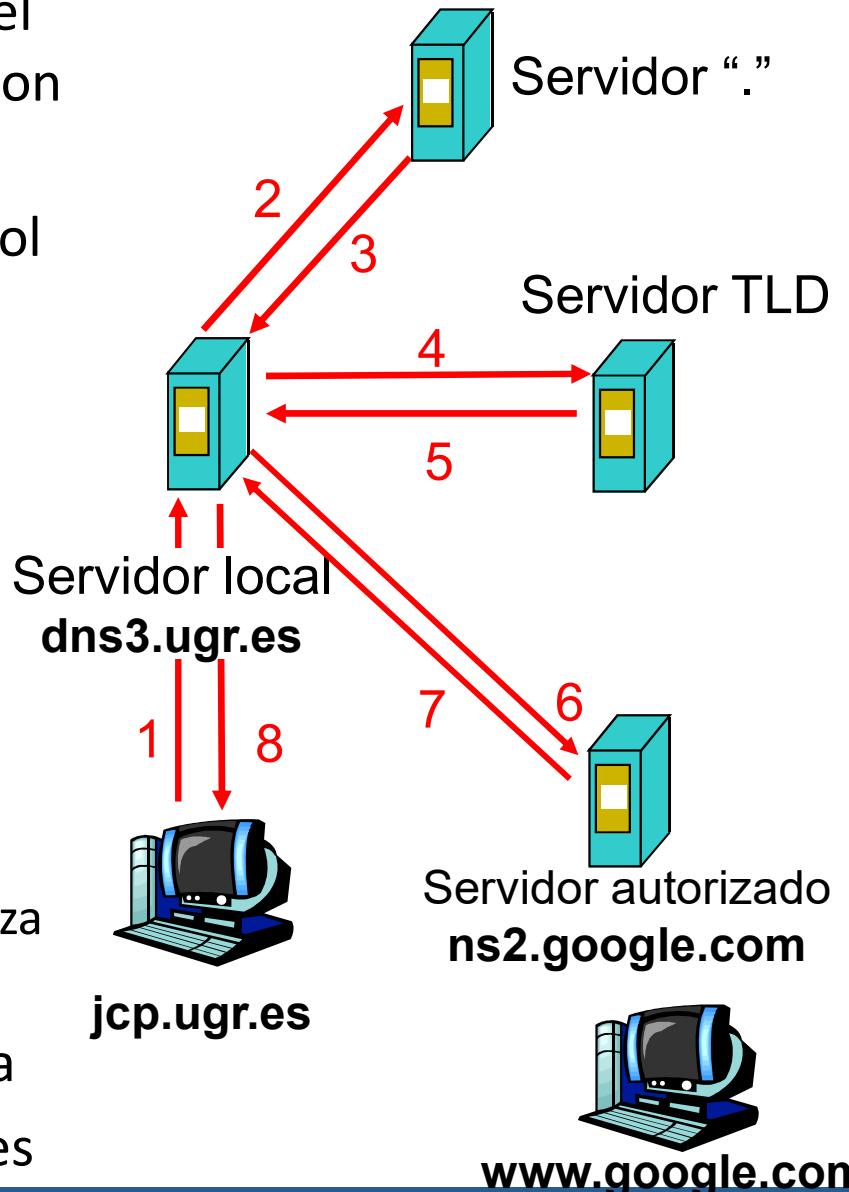
Inicialmente fueron definidos los siguientes TLDs (RFC 1591):

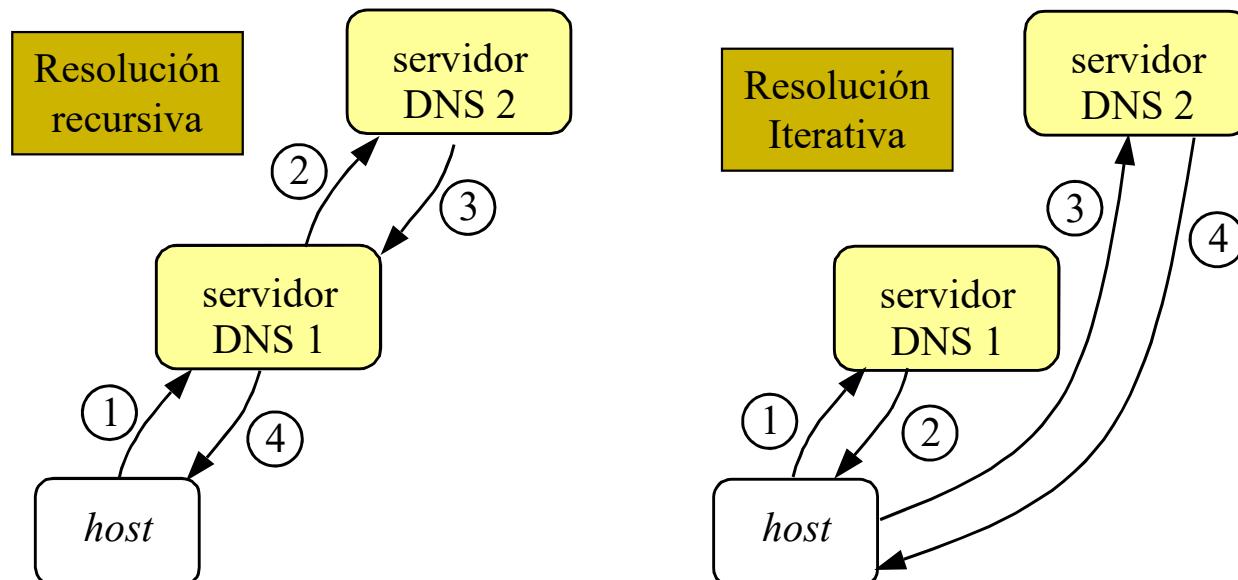
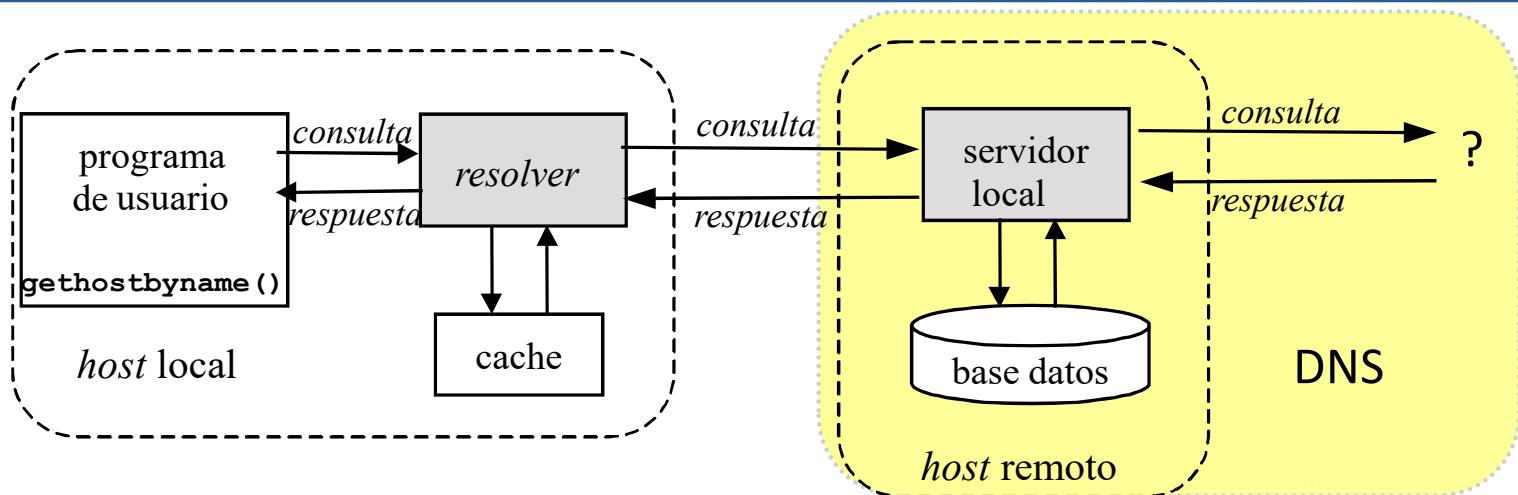
- .com** -> organizaciones comerciales
- .edu** -> instituciones educativas, como universidades, de EEUU.
- .gov** -> instituciones gubernamentales estadounidenses
- .mil** -> grupos militares de estados unidos
- .net** -> proveedores de Internet
- .org** -> organizaciones diversas diferentes de las anteriores
- .arpa**-> propósitos exclusivos de infraestructura de Internet
- .int** -> organizaciones establecidas por tratados internacionales entre gobiernos
- .xy** -> indicativos de la zona geográfica (ej. es (España); pt (portugal); jp (Japón)...

En Julio de 2020 se contabilizan 1511 TLDs

SERVICIO DE NOMBRES DE DOMINIO (DNS)

- DNS es un **protocolo** de aplicación para el acceso a una base de datos distribuida con una gestión distribuida.
- 3 niveles de servidores, asociados al árbol de nombres de dominio:
 - Servidores raíz “.” (*Root servers*)
 - Top-Level Domain (*TLD servers*)
 - Servidores Locales (*Local servers*)
- En jcp.ugr.es, ¿IP de www.google.com?
 1. Consulta al “resolver” local (cache)
 2. Si no hay respuesta, pregunta al server DNS local. ¿Cómo se conoce su IP?
 3. Si no hay respuesta, el server local realiza la “resolución” (ver página siguiente)
- Resoluciones: iterativa, recursiva o mixta
- Para mejorar prestaciones se usan caches

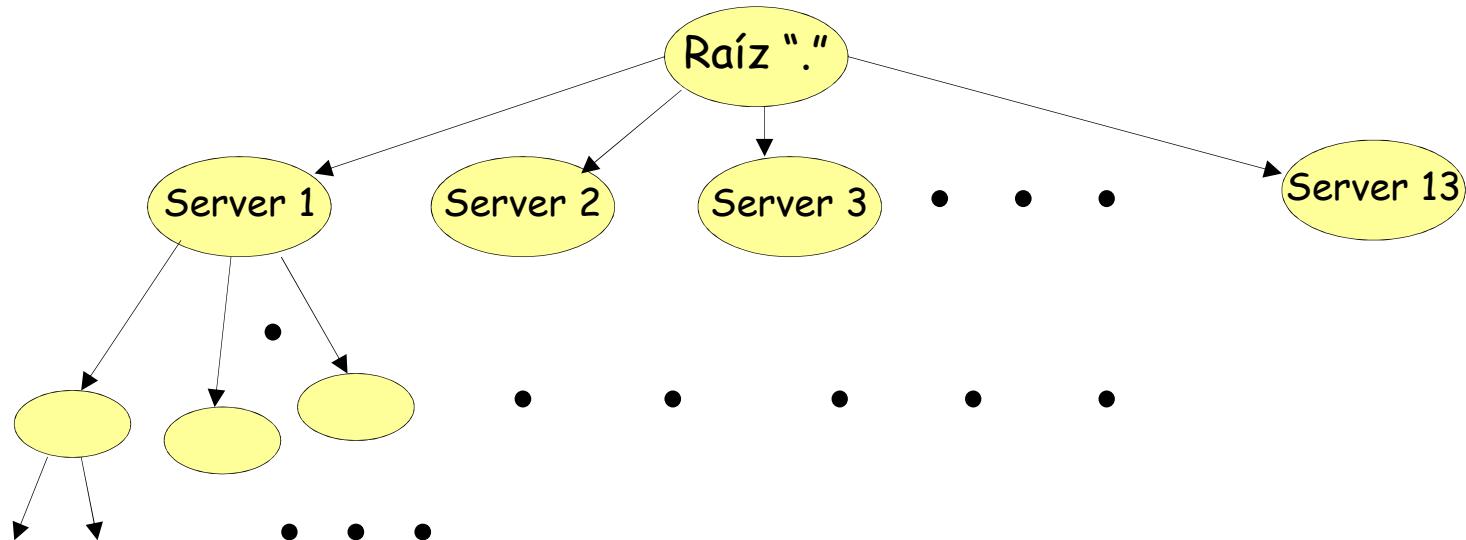




➤ Normalmente, la consulta del host al server local es recursiva, mientras que las de este son iterativas

Conceptos y gestión de la base de datos (distribuida y jerárquica):

- ❑ El sistema DNS está formado por un conjunto de servidores cooperativos, organizados en árbol, que almacenan parcialmente la base de datos, denominada BIND (Berkeley Internet Name Domain).
- ❑ Cada servidor es responsable de lo que se denomina una **ZONA**.
- ❑ Una *zona* es el conjunto de nombres de dominio completo (por debajo de un nodo en el árbol de nombres de dominio) de los que un servidor tiene toda la información y es su **autoridad**.
- ❑ La autoridad puede **delegarse** jerárquicamente a otros servidores (hijos)



Gestión de la base de datos DNS:

- Delegar autoridad** de una zona implica ceder la gestión y responsabilidad de la misma al servidor correspondiente.
- En cada zona hay servidores **primarios** (almacenan una copia *master* de la db en discos locales) y servidores **secundarios** (obtienen la db por transferencia)
- Además, existe un servicio de **cache** para mejorar prestaciones.
- Para garantizar escalabilidad existen **13 servidores** raíz (identificados de A a M) (ver <http://www.root-servers.org>) con múltiples instancias (réplicas con la misma IP) repartidas por el mundo
- El root-server F (y otros) tiene un servidor en Madrid (**Espanix: punto neutro**)
- Cuando un cliente (a través de un *resolver local*) 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.

Root-servers <http://www.root-servers.org/>

Servidor A: Network Solutions, Herndon, Virginia, USA.

Servidor B: Instituto de Ciencias de la Información de la Universidad del Sur de California, USA.

Servidor C: PSINet, Virginia, USA.

Servidor D: Universidad de Maryland, USA.

Servidor E: NASA, en Mountain View, California, USA.

Servidor F: Internet Software Consortium, Palo Alto, California, USA.

Servidor G: Agencia de Sistemas de Información de Defensa, California, USA.

Servidor H: Laboratorio de Investigación del Ejercito, Maryland, USA.

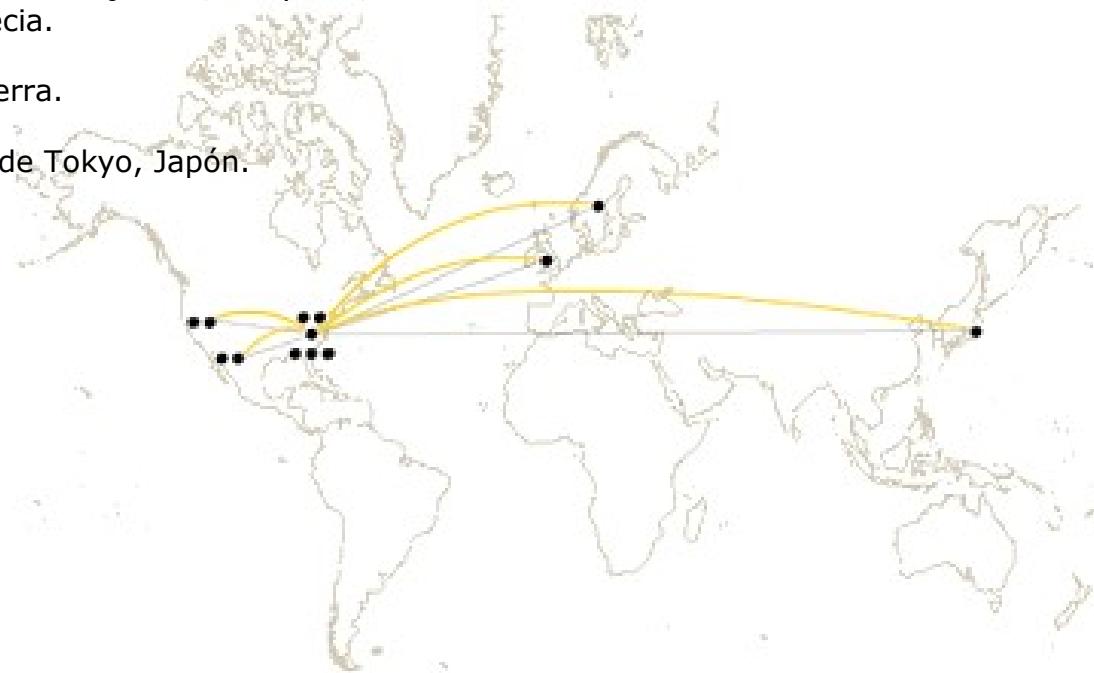
Servidor I: NORDUnet, Estocolmo, Suecia.

Servidor J: (TBD), Virginia, USA.

Servidor K: RIPE-NCC, Londres, Inglaterra.

Servidor L: (TBD), California, USA.

Servidor M: Wide Project, Universidad de Tokyo, Japón.



¿Cómo es la base de datos DNS?

- La base de datos se almacena en ficheros de texto (txt) denominados *zone files*.
- Cada *zone file* contiene registros denominados **Resource Record**.
- Cada *zone file* define un **TTL** (Time to Live): tiempo de validez de los RRs en cache.
- Cada **RR** contiene:

Nombre del dominio: nombre del dominio al que se refiere el RR.

Clase: en Internet siempre IN.

Tipo: Tipo de registro.

SOA	Registro (S tart O f A uthority) con la autoridad de la zona.
NS	Registro que contiene un servidor de nombres.
A	Registro que define una dirección IPv4.
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.
PTR	Registro que contiene un nombre de dominio (para resoluciones inversas)

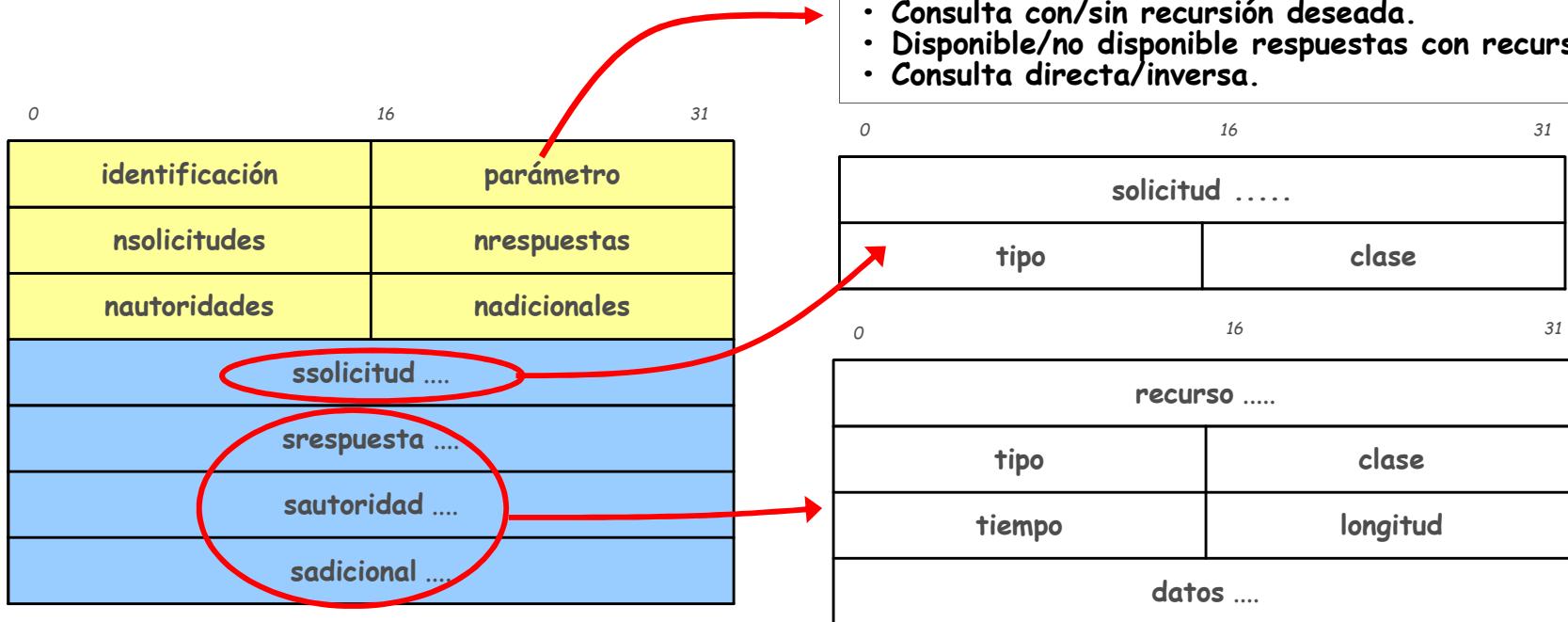
Valor: Contenido que depende del campo tipo

- Existe una base de datos asociada de **resolución inversa** para traducir direcciones IP en nombres de dominio. (in-addr.arpa)
- Cuando un cliente DNS quiere obtener el nombre de dominio correspondiente a una IP por ejemplo 1.2.3.4, hace una resolución inversa, preguntando el RR tipo PTR asociado a 4.3.2.1.in-addr.arpa.

```
; IPv4 zone file for example.com
$TTL 2d      ; default TTL for zone
$ORIGIN example.com. ; base domain-name
; Start of Authority record defining the key characteristics
; of the zone (domain)
@       IN      SOA    ns1.example.com. hostmaster.example.com. (
                        2003080800 ; se = serial number
                        12h          ; ref = refresh
                        15m          ; ret = refresh retry
                        3w          ; ex = expiry
                        2h          ; nx = nxdomainttl
)
; name servers Resource Records for the domain
        IN      NS     ns1.example.com.
; the second name server is
; external to this zone (domain).
        IN      NS     ns2.example.net.
; mail server Resource Records for the zone (domain)
; value 10 denotes it is the most preferred
        3w      IN      MX   10  mail.example.com.
; the second mail server has lower preference (20) and is
; external to the zone (domain)
        IN      MX   20  mail.example.net.
; domain hosts includes NS and MX records defined previously
; plus any others required
ns1       IN      A      192.168.254.2
mail      IN      A      192.168.254.4
joe       IN      A      192.168.254.6
www       IN      A      192.168.254.7
; aliases ftp (ftp server) to an external location
ftp       IN      CNAME  ftp.example.net.
```

Ejemplo de *zone file* para el dominio **example.com**

Formato mensajes DNS:



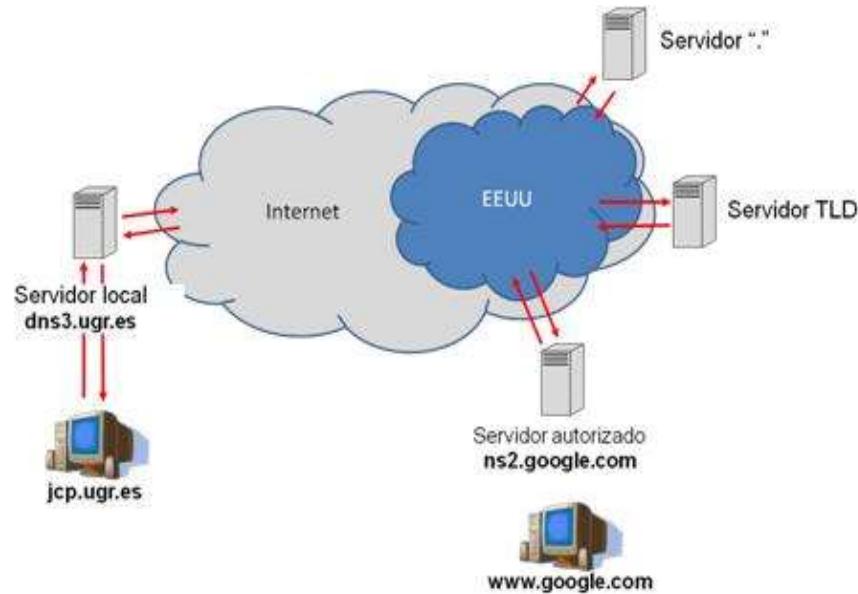
DNS se ofrece en el puerto 53 mediante UDP normalmente o TCP (para respuestas grandes > 512 bytes).

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 http://www.dominios.es/dominios/sites/dominios/files/1318333648229_0.pdf

SERVICIO DE NOMBRES DE DOMINIO (DNS)

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 μ s 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:



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

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?

Tema 5.

SERVICIOS Y PROTOCOLOS DE APLICACIÓN EN INTERNET

1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
- 3. La navegación Web (HTTP)**
4. El Correo electrónico (SMTP/IMAP/POP3)
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

LA NAVEGACIÓN WEB

- Una página web es un fichero formado por objetos:
Contenido en HTML, imágenes JPEG, GIFs, Java applets, contenido de audio, vídeo, etc.
- Cada objeto se dirige por una URL (o URI):

[esquema:] [// [user [:password] @] dominio [:puerto] [/path]
[/recurso] [?solicitud] [#fragmento]

Name	Used for	Example
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
news	Newsgroup	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Sending e-mail	mailto:JohnUser@acm.org
telnet	Remote login	telnet://www.w3.org:80

- Las páginas web se sirven con el protocolo HTTP (*Hyper Text Transfer Protocol*) en el que se adopta el modelo cliente-servidor
 - cliente*: browser que solicita, recibe y muestra objetos web
 - servidor*: envía objetos web en respuesta a peticiones
- Las páginas web pueden ser **estáticas** (contenido invariable) o **dinámicas** (con contenido variable).
 - Usando lenguajes de *scripting en el cliente*: JavaScript o Flash etc
 - Usando lenguajes de *scripting en el servidor*: Perl, PHP, Ruby, Python etc. Se utilizan incrustando etiquetas dentro de la página web. Cuando el cliente solicita esa página web, el servidor web interpreta estas etiquetas para realizar acciones en el servidor generando contenido dinámico. Por ejemplo, insertando información de una base de datos.

➤ Características del protocolo HTTP:

- **Usa los servicios de TCP (S.O.C.) en el puerto 80 →**
implica el inicio de una conexión TCP, envío de mensajes HTTP y cierre de la conexión TCP
- **HTTP es “stateless” → para simplificar la interacción al usuario usa Cookies**
El servidor no mantiene información sobre las peticiones de los clientes (su estado) y así ahorra recursos aunque hace más compleja la interacción
- **HTTP es un protocolo *in-band*, orientado a texto.**
- **Existen dos tipos de servidores HTTP**
No persistente (HTTP/v1.0) → Se envía únicamente un objeto en cada conexión TCP.
Persistente (HTTP/v1.1) → Pueden enviarse múltiples objetos sobre una única conexión TCP entre cliente y servidor.

LA NAVEGACIÓN WEB: MENSAJES HTTP



1. El cliente HTTP (navegador) solicita un objeto identificado por su URL, en el ejemplo www.ugr.es/pages/Universidad. Según la configuración del servidor, si no se especifica nada, por defecto se sirve el fichero index.html
2. El cliente consulta al *resolver* de DNS por la dirección IP de www.ugr.es
3. DNS contesta 150.214.204.231
4. El cliente abre una conexión TCP al puerto 80 de 150.214.204.231 (3 bandas)
5. El cliente envía una petición “GET /pages/universidad/ ...” (más otra información adicional: cabeceras, cookies, variables, etc)
6. El servidor responde enviando el fichero “index.html” por la misma conexión TCP
7. Al usar TCP el cliente y servidor de HTTP reciben un servicio orientado a conexión, fiable, sin errores, con control de flujo, con control de congestión, etc. Es decir una comunicación TRANSPARENTE y FIABLE.
8. Si es *persistente* se siguen solicitando objetos de la página (“GET...”) por la conexión
9. Se cierra la conexión TCP y se liberan recursos en el servidor y cliente
10. El cliente visualiza el contenido

LA NAVEGACIÓN WEB: MENSAJES HTTP



- 1a. El Cliente HTTP inicia conexión TCP al servidor HTTP (proceso) en www.ugr.es en el puerto 80 (segmento SYNC de TCP sin datos)
2. El Cliente HTTP envía *request message* para el objeto
- tiempo 4. Si es persistente → Envío de más objetos por la misma conexión TCP
3. El servidor HTTP acepta la conexión y solicita al cliente abrir la conexión (SYNC+ACK)
- 1c. El cliente confirma (ACK)
5. Cierre de conexión TCP (liberación de recursos)
6. Nuevas conexiones TCP

HTTP define dos tipos de mensajes (*request, response*):

1. *HTTP request message* (solicitudes del cliente al servidor):

Línea de petición

(GET, POST,
HEAD)

GET /somedir/page.html HTTP/1.1

Host: www.someschool.edu

User-agent: Mozilla/4.0

Connection: close

Accept-language: fr

Líneas de cabecera

Carriage return +
line feed

(extra carriage return, line feed)

Indican fin del mensaje

HTTP define dos tipos de mensajes (*request, response*):

2. HTTP *response message*: (respuestas del servidor al cliente):

Línea de estado

HTTP/1.1 200 OK

Connection: close

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998

Content-Length: 6821

Content-Type: text/html

200 OK
301 Moved Permanently
400 Bad Request
404 Not Found
505 HTTP Version Not Supported

Líneas de cabecera

Datos,
ej. fichero html

data data data data data ...

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

LA NAVEGACIÓN WEB: Protocolo HTTP 1.1 (RFC 2616)

- **MÉTODOS** (acciones solicitadas por los clientes en los *request messages*):
 - **OPTIONS**: solicitud de información sobre las opciones disponibles
 - **GET**: solicitud de un recurso (puede ser condicional)
 - **HEAD**: igual que GET pero el servidor no devuelve el “cuerpo” sólo cabeceras
 - **POST**: solicitud al servidor para que acepte y subordine a la URI especificada, los datos incluidos en la solicitud
 - **PUT**: solicitud de sustituir la URI especificada con los datos incluidos en la solicitud.
 - **DELETE**: solicitud de borrar la URI especificada.
- **CÓDIGOS DE RESPUESTA** (para los *response messages del servicor*):
 - **1xx** indican mensajes exclusivamente informativos
 - **2xx** indican algún tipo de éxito
 - **3xx** redireccion al cliente a otra URL
 - **4xx** indican un error
 - **5xx** indican un error
- **CABECERAS** (*47 request headers* y *49 response headers*)

From: , User-Agent: , Content-Type: , Content-Length: ,

http://en.wikipedia.org/wiki/List_of_HTTP_header_fields

- <https://developer.mozilla.org/es/docs/Web/HTTP/Headers>
- https://www.tutorialspoint.com/http/http_quick_guide.htm

✓ **Cabeceras comunes para peticiones y respuestas**

- **Content-Type:** descripción MIME de la información contenida en este mensaje.
- **Content-Length:** longitud en bytes de los datos enviados, expresado en base decimal.
- **Content-Encoding:** formato de codificación de los datos enviados en este mensaje. Sirve, por ejemplo, para enviar datos comprimidos o encriptados.
- **Date:** fecha local de la operación. Las fechas deben incluir la zona horaria en que reside el sistema que genera la operación. Por ejemplo: Sunday, 12-Dec-96 12:21:22 GMT+01. No existe un formato único en las fechas.

✓ **Cabeceras sólo para peticiones del cliente**

- **Accept**: campo opcional que contiene una lista de tipos MIME aceptados por el cliente.
- **Authorization**: clave de acceso que envía un cliente para acceder a un recurso de uso protegido o limitado. La información incluye el formato de autorización empleado, seguido de la clave de acceso propiamente dicha. La explicación se incluye más adelante.
- **From**: campo opcional que contiene la dirección de correo electrónico del usuario del cliente Web que realiza el acceso.

- **If-Modified-Since**: permite realizar operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la fecha proporcionada. Puede ser utilizada por los sistemas de almacenamiento temporal de páginas. Es equivalente a realizar un HEAD seguido de un GET normal.
- **Referer**: contiene la URL del documento desde donde se ha activado este enlace. De esta forma, un servidor puede informar al creador de ese documento de cambios o actualizaciones en los enlaces que contiene. No todos los clientes lo envían.
- **User-agent**: cadena que identifica el tipo y versión del cliente que realiza la petición. Por ejemplo, los *browsers* de Netscape envían cadenas del tipo User-Agent: Mozilla/3.0 (WinNT; I)

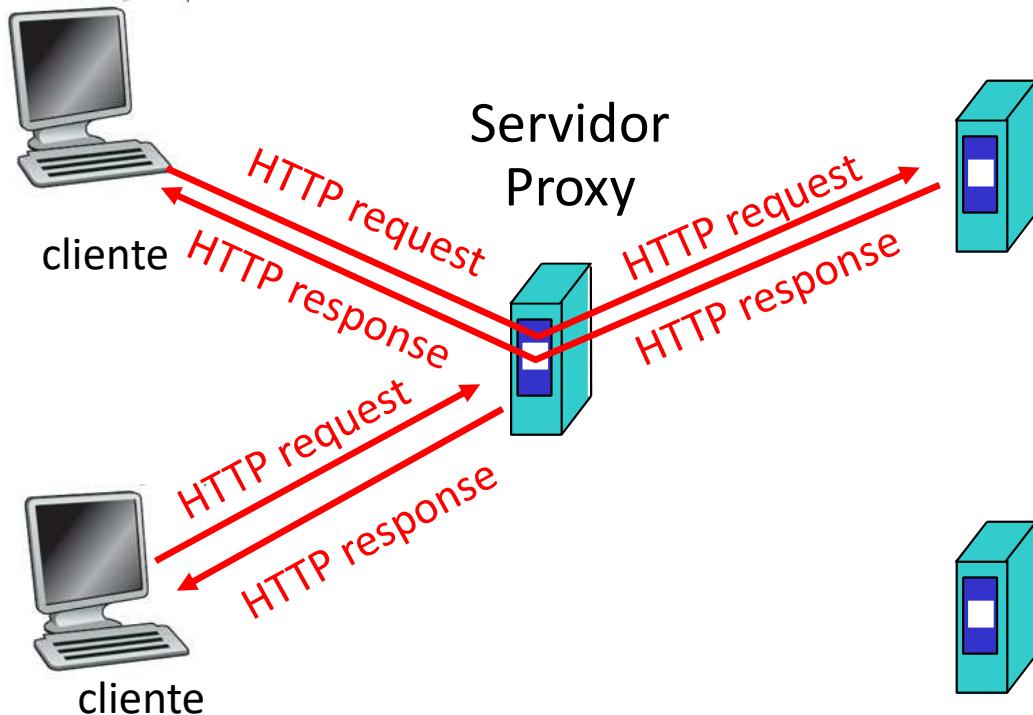
✓ **Cabeceras sólo para respuestas del servidor HTTP**

- **Allow:** informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere esta respuesta. Por ejemplo, Allow: GET, POST.
- **Expires:** fecha de expiración del objeto enviado. Los sistemas de cache deben descartar las posibles copias del objeto pasada esta fecha. Por ejemplo, Expires: Thu, 12 Jan 97 00:00:00 GMT+1. No todos los sistemas lo envían.
- **Last-modified:** fecha local de modificación del objeto devuelto. Se puede corresponder con la fecha de modificación de un fichero en disco, o, para información generada dinámicamente desde una base de datos, con la fecha de modificación del registro de datos correspondiente.

LA NAVEGACIÓN WEB: Web cache

Objetivo: reducir el tráfico generado sin implicar servir contenido desactualizado.

- El usuario configura el navegador para que todas las solicitudes se cursen por el proxy/cache
- La cache puede residir en el propio ordenador del usuario
- El navegador enviará todos los requerimientos HTTP al cache
 - Si objeto está en la cache: se retorna el objeto local
 - Si no, la cache solicita el objeto desde al servidor destino, actualiza la cache con el objeto, y sirve el objeto al cliente



- Ejemplo de una respuesta típica de un servidor configurado para gestionar caches:

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

- Las cabeceras se asocian al fichero en la cache local

LA NAVEGACIÓN WEB: Web cache

- **Objetivo:** reducir el tráfico si cache la cache tiene una versión actualizada del objeto
- La cache solicita el objeto condicionado a la fecha de la copia local, usando:

If-modified-since:
<date> ó

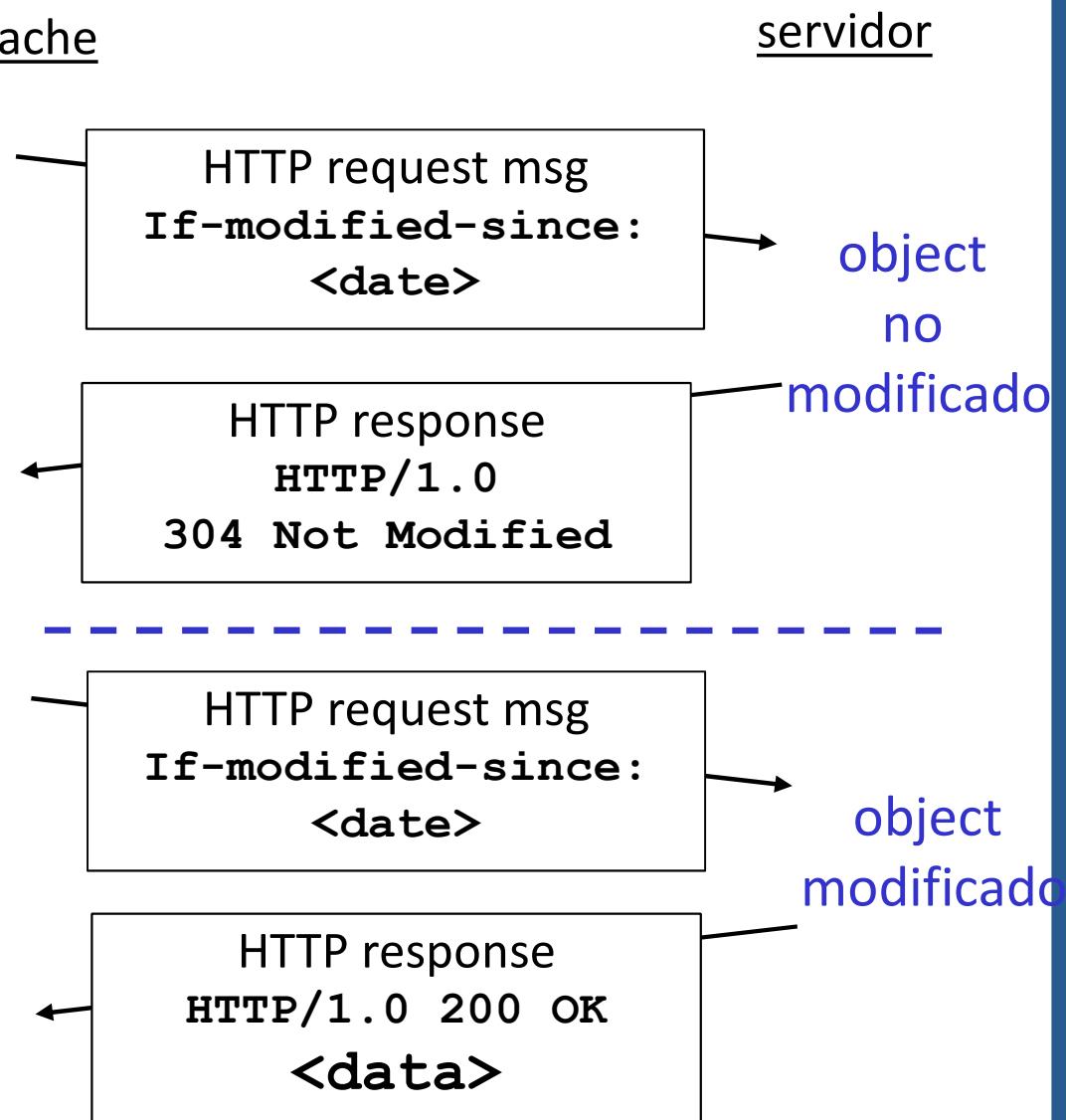
If-None-Match:
"686897696a7c876b7e"

- El servidor responde sin el objeto si la copia de la cache está actualizada:

HTTP/1.0 304 Not Modified

- Para más detalles ver

<https://www.keycdn.com/blog/http-cache-headers>



LA NAVEGACIÓN WEB: COOKIES

- ✓ Las **cookies** son pequeños ficheros de texto que se intercambian los clientes y servidores HTTP, para solucionar una de las principales deficiencias del protocolo: la falta de información de estado entre dos transacciones. Fueron introducidas por Netscape, y ahora han sido estandarizadas en el RFC 2109.
- La primera vez que un usuario accede a un determinado documento de un servidor, éste proporciona una *cookie* que contiene datos que relacionarán posteriores operaciones.
 - El cliente almacena la *cookie* en su sistema para usarla después. En los futuros accesos a este servidor, el *navegador* podrá proporcionar la *cookie* original, que servirá de nexo entre este acceso y los anteriores.
 - Todo este proceso se realiza automáticamente, sin intervención del usuario.

LA NAVEGACIÓN WEB: COOKIES

Uso de las *cookies*

- ✓ Una *cookie* es simplemente una serie de líneas de texto, con pares variable/valor. Existe un conjunto predefinido de nombres de variable, necesarias para el correcto funcionamiento de las *cookies*.
 - **Domain**= conjunto de direcciones Internet para el que es válida la *cookie*. Se puede dar una dirección única (www.mitienda.es) o un rango (.netscape.com).
 - **Path**= fija el subconjunto de URLs para las que sirve esta *cookie*.
 - **Version**= Permite seleccionar entre diferentes versiones del modelo de *cookies*.
 - **Expires**= Fecha de expiración de la información. Si no se incluye, los datos son descartados al finalizar la sesión con el cliente Web.

➤ <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>

LA NAVEGACIÓN WEB: COOKIES

- ✓ Un servidor HTTP envía los diferentes campos de una *cookie* con la nueva cabecera HTTP Set-Cookie:

Set-Cookie: Domain=www.unican.es; Path=/; Nombre=Luis; Expires Fri, 15-Jul-97 12:00:00 GMT

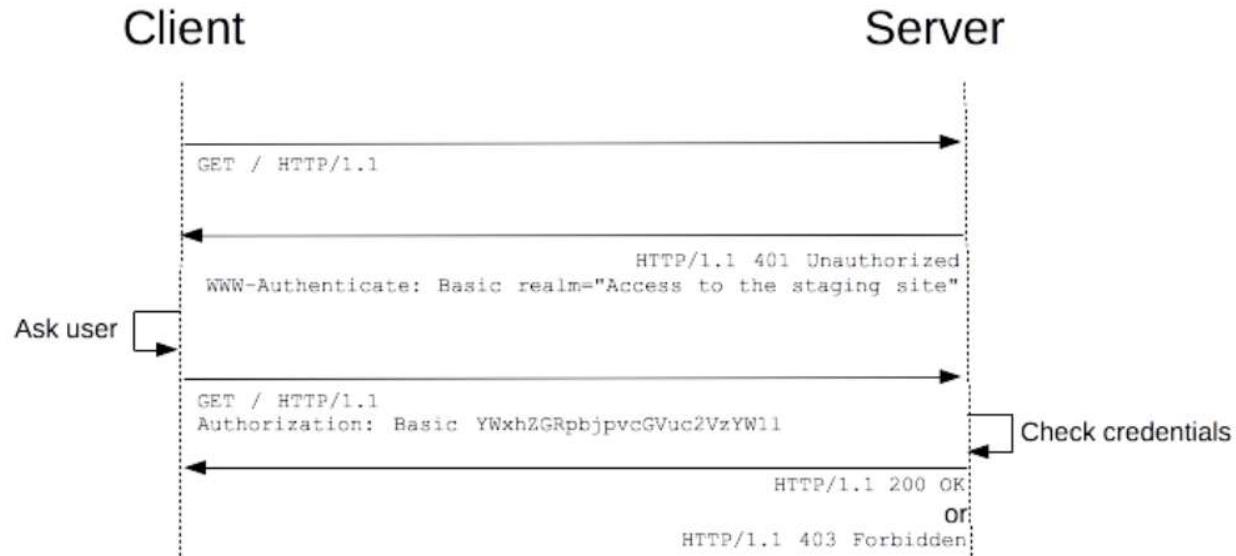
- Cuando se accede a una URL que verifica el par dominio/path registrado, el cliente enviará automáticamente la información de los diferentes campos de la cookie con la nueva cabecera HTTP Cookie:



LA NAVEGACIÓN WEB: ACCESO RESTRIGIDO

- HTTP no es seguro, pero incluye cabeceras (WWW-Authenticate y Authorization) para restringir el acceso a recursos.

- Es vulnerable a ataques por repetición.



- WWW-Authenticate: <type> realm=<realm>[, charset="UTF-8"]
- Authorization: <type> <credentials>
<credentials> si <type> es BASIC incluye el username:password codificado en BASE64

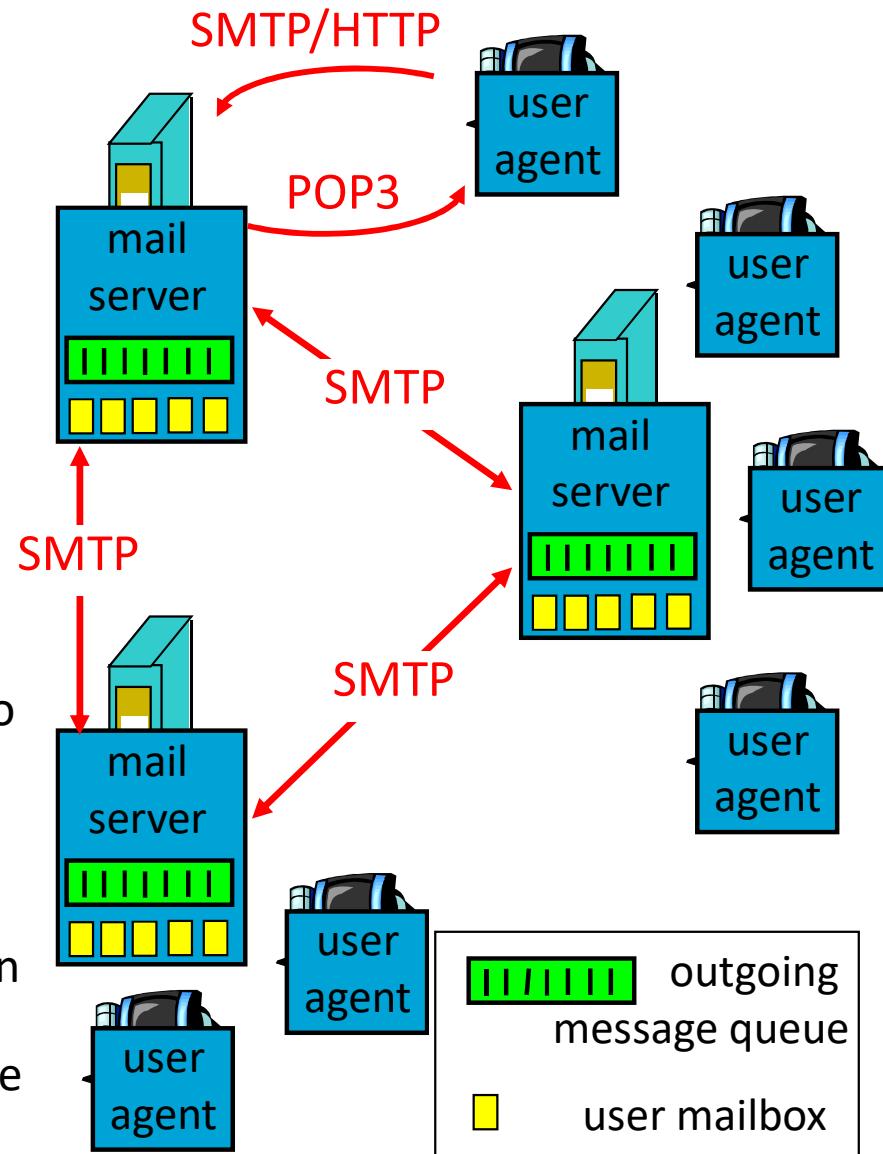
Tema 5.

SERVICIOS Y PROTOCOLOS DE APLICACIÓN EN INTERNET

1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
3. La navegación Web (HTTP)
- 4. El Correo electrónico (SMTP/IMAP/POP3)**
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

EL CORREO ELECTRÓNICO

- Elementos y protocolos principales:
 - Cliente de correo (*Mail User Agent*)
 - Servidor de correo (*Mail Server* o *Mail Transfer Agent*)
 - Protocolo de envío:
Simple Mail Transfer Protocol (SMTP)
 - Protocolos de descarga (o lectura):
POP3, IMAP, HTTP
- Agente de usuario (*MUA*):
 - Compone, edita y lee mensajes de correo del buzón. Ej. Outlook, Thunderbird, etc
- Servidor de correo (*MTA*)
 - Reenvía mensajes salientes y almacena en buzones los mensajes entrantes de cada usuario. Permite desacoplar temporalmente a remitente y destinatario



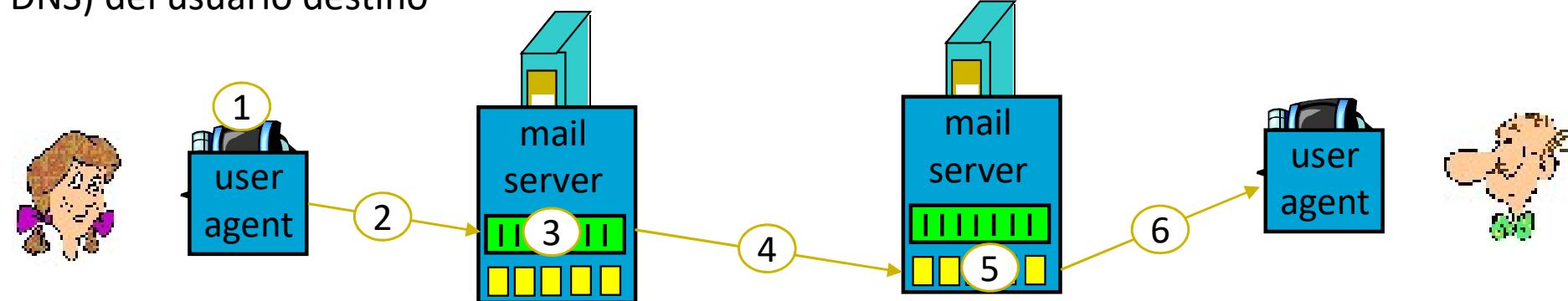
EL CORREO ELECTRÓNICO: SMTP (RFC 2821)

- SMTP se implementa mediante dos programas (incluidos ambos en cada *mail server*):
 - Cliente SMTP: se ejecuta en el *mail server (MTA)* que está enviando correo
 - Servidor SMTP: se ejecuta en el *mail server (MTA)* que está recibiendo correo
 - "sendmail" <http://en.wikipedia.org/wiki/Sendmail>
- SMTP usa TCP en el puerto 25. Es un protocolo orientado a texto.
- SMTP es un protocolo orientado a conexión, es *in-band* y es *state-full*: implica tres fases
 - *Handshaking* (“saludo”)
 - Transferencia de mensajes
 - Cierre
- La interacción entre cliente SMTP y servidor SMTP se realiza mediante comandos/respuesta
 - **comandos:** texto ASCII
 - **respuestas:** código de estado y frases explicativas
- Inicialmente los mensajes estaban codificados en ASCII de 7 bits!! ➔ Con la definición posterior de las extensiones MIME se puede enviar ASCII de 8 bits y formatos enriquecidos

EL CORREO ELECTRÓNICO: SMTP (RFC 2821)

➤ Pasos en el envío/recepción de correo

- 1) El usuario origen compone mediante su Agente de Usuario (MUA) un mensaje dirigido a la dirección de correo del usuario destino
- 2) Se envía con SMTP (ó HTTP) el mensaje al servidor de correo (MTA) 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 (MTA) (obtenido por DNS) del usuario destino



EL CORREO ELECTRÓNICO: SMTP (RFC 2821)

```
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
```

➤ *Propuesta de ejercicio:
dibujar el diagrama de
estados de SMTP*

EL CORREO ELECTRÓNICO: SMTP (RFC 2821)

➤ Comandos SMTP: cliente

Comando	Descripción
HELO (ahora EHLO)	Identifica el remitente al destinatario.
MAIL FROM	Identifica una transacción de correo e identifica al emisor.
RCPT TO	Se utiliza para identificar un destinatario individual . Si se necesita identificar múltiples destinatarios es necesario repetir el comando.
DATA	Permite enviar una serie de líneas de texto. El tamaño máximo de una línea es de 1.000 caracteres. Cada línea va seguida de un retorno de carro y avance de línea <CR><LF>. La última línea debe llevar únicamente el carácter punto ":" seguido de <CR><LF>.
RSET	Aborta la transacción de correo actual.
NOOP	No operación. Indica al extremo que envíe una respuesta positiva. Keepalives
QUIT	Pide al otro extremo que envíe una respuesta positiva y cierre la conexión.
VRFY	Pide al receptor que confirme que un nombre identifica a un destinatario valido.
EXPN	Pide al receptor la confirmación de una lista de correo y que devuelva los nombres de los usuarios de dicha lista.
HELP	Pide al otro extremo información sobre los comandos disponibles.
TURN	El emisor pide que se inviertan los papeles , para poder actuar como receptor. El receptor puede negarse a dicha petición.
SOML	Si el destinatario está conectado, entrega el mensaje directamente al terminal, en caso contrario lo entrega como correo convencional.
SAML	Entrega del mensaje en el buzón del destinatario. En caso de estar conectado también lo hace al terminal.
SEND	Si el destinatario está conectado, entrega el mensaje directamente al terminal.

➤ Códigos de respuesta SMTP: servidor

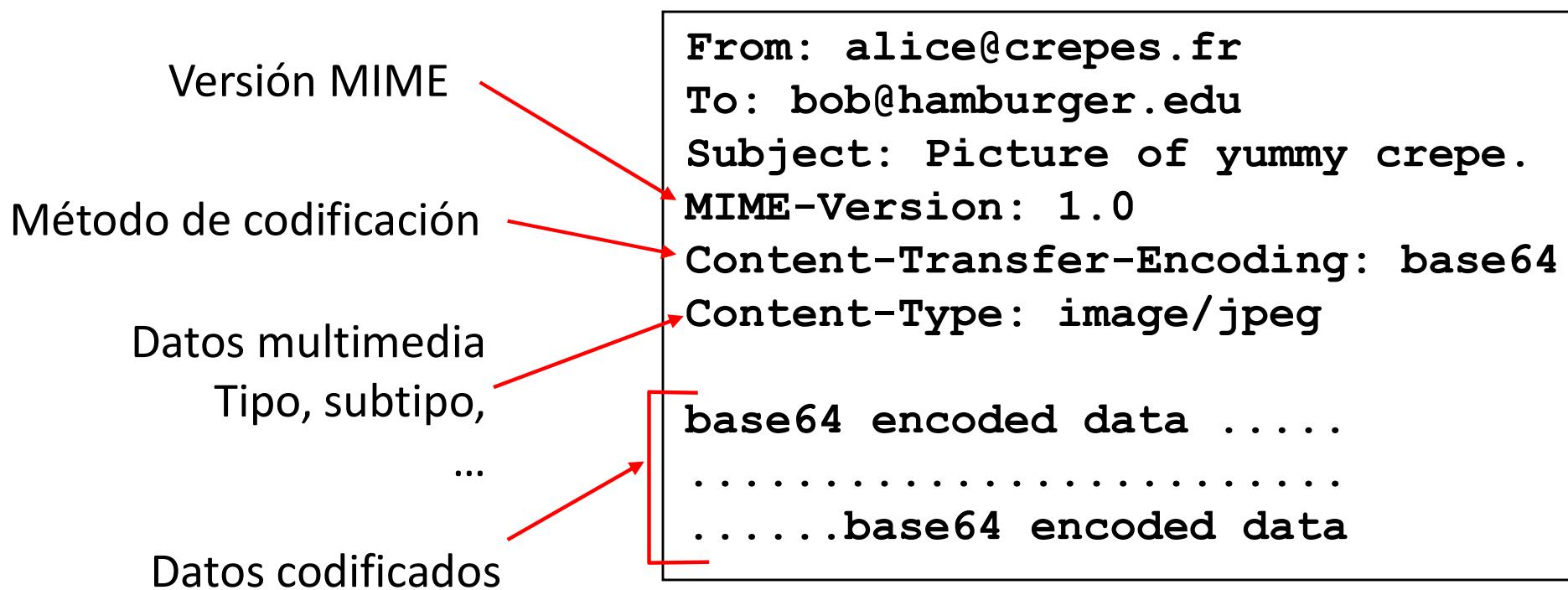
<u>Código</u>	<u>Descripción</u>
211	Estado del sistema.
214	Mensaje de ayuda.
220	Servicio preparado.
221	Servicio cerrando el canal de transmisión.
250	Solicitud completada con éxito.
251	Usuario no local, se enviará a <dirección de reenvío>
354	Introduzca el texto, finalice con <CR><LF>.<CR><LF>.
421	Servicio no disponible.
450	Solicitud de correo no ejecutada, servicio no disponible (buzón ocupado).
451	Acción no ejecutada, error local de procesamiento.
452	Acción no ejecutada, insuficiente espacio de almacenamiento en el sistema.
500	Error de sintaxis, comando no reconocido.
501	Error de sintaxis. P.ej contestación de SMTP a ESMTP
502	Comando no implementado.
503	Secuencia de comandos errónea.
504	Parámetro no implementado.
550	Solicitud no ejecutada, buzón no disponible.
551	Usuario no local, pruebe <dirección de reenvío>. Si no se tiene cuenta
552	Acción de correo solicitada abortada.
553	Solicitud no realizada (error de sintaxis).
554	Fallo en la transacción.

Multipurpose Internet Mail Protocol Extensions (MIME):

- Nada cambia respecto a la arquitectura de correo anterior.
- Las extensiones de MIME van encaminadas a soportar:
 - Texto en conjuntos de caracteres distintos de US-ASCII;
 - Adjuntos que no son de tipo texto;
 - Cuerpos de mensajes con múltiples partes (multi-part);
 - Información de encabezados con conjuntos de caracteres distintos de ASCII.
- MIME está especificado en seis RFCs: RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 y RFC 2077.

EL CORREO ELECTRÓNICO: EXTENSIONES MIME

- No confundir los mensajes del protocolo con el formato de almacenamiento de los mensajes



EL CORREO ELECTRÓNICO: EXTENSIONES MIME

➤ Cabeceras de mensajes MIME

Cabecera	Descripción
MIME-Version:	Identifica la versión de MIME. Si no existe se considera que el mensaje es texto normal en inglés.
Content-Description:	Cadena de texto que describe el contenido. Esta cadena es necesaria para que el destinatario sepa si desea descodificar y leer el mensaje o no.
Content-Id:	Identificador único, usa el mismo formato que la cabecera estándar Message-Id.
Content-Transfer-Encoding:	Indica la manera en que está envuelto el cuerpo del mensaje.
Content-Type:	Especifica la naturaleza del cuerpo del mensaje.

➤ Content-Transfer-Encoding

- Indica la manera en que está envuelto el cuerpo para su transmisión, ya que podría haber problemas con la mayoría de los caracteres distintos de letras, números y signos de puntuación.
- Existen 5 tipos de codificación (RFC1521) : *ASCII 7, ASCII 8, codificación binaria, base64 y entrecomillada-imprimible.7.2*

EL CORREO ELECTRÓNICO: EXTENSIONES MIME

MIME: Content-Type: tipos y subtipos

- La lista inicial de tipos y subtipos especificada por el RFC 1521 es:

Tipo	Subtipo	Descripción
Text	Plain	Texto sin formato.
	Richtext	Texto con comandos de formato sencillos.
Image	Gif	Imagen fija en formato GIF.
	Jpeg	Imagen fija en formato JPEG.
Audio	Basic	Sonido.
Video	Mpeg	Película en formato MPEG.
Application	Octet-stream	Secuencia de bytes no interpretada.
	Postscript	Documento imprimible PostScript.
Message	Rfc822	Mensaje MIME RFC 822.
	Partial	Mensaje dividido para su transmisión.
	External-body	El mensaje mismo debe obtenerse de la red.
Multipart	Mixed	Partes independientes en el orden especificado.
	Alternative	Mismo mensaje en diferentes formatos.
	Parallel	Las partes deben verse simultáneamente.
	Digest	Cada parte es un mensaje RFC 822 completo.

MIME: Content-Type: tipo application:

- El tipo ***application*** es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos.
- El subtipo ***octet-stream*** simplemente es una secuencia de bytes no interpretados, tal que a su recepción, un agente de usuario debería *presentarla en la pantalla sugiriendo al usuario que se copie en un archivo y solicitando un nombre de archivo*.
- El subtipo ***postscript***, se refiere al lenguaje PostScript de Adobe Systems. Aunque un agente de usuario puede llamar a un intérprete PostScript externo para visualizarlo, hacerlo no está exento de riesgos al ser PostScript un lenguaje de programación completo.

EL CORREO ELECTRÓNICO: EXTENSIONES MIME

MIME: Content-Type: tipo message:

- El tipo ***message*** permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar, correo electrónico.
- El subtipo ***rfc822*** se utiliza cuando se encapsula un mensaje RFC 822 completo en un mensaje exterior.
- El subtipo ***partial*** hace posible dividir un mensaje encapsulado en pedazos y enviarlos por separado. **Los parámetros hacen posible ensamblar correctamente todas las partes en el destino.** Ej: 1/3, 2/3, 3/3.
- El subtipo ***external-body*** puede usarse para mensajes muy grandes, por ejemplo películas de vídeo. En lugar de incluir el archivo mpeg en el mensaje, se da una dirección de FTP y el agente de usuario del receptor puede obtenerlo a través de la red cuando se requiera.

EL CORREO ELECTRÓNICO: EXTENSIONES MIME

MIME: Content-Type: tipo multipart:

- El tipo es ***multipart***, que permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte claramente delimitados.
- El subtipo ***mixed*** permite que cada parte sea diferente.
- El subtipo ***alternative*** indica que cada parte contiene el mismo mensaje, pero expresado en un medio o codificación diferente.
- El subtipo ***parallel*** se usa cuando todas las partes deben “verse” simultáneamente, por ejemplo, en los canales de audio y vídeo de las películas.
- El subtipo ***digest*** se usa cuando se juntan muchos mensajes en un mensaje compuesto.

EL CORREO ELECTRÓNICO: EXTENSIONES MIME (EJEMPLO Multipart/mixed)

From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"

This is the preamble. It is to be ignored, though it
is a handy place for mail composers to include an
explanatory note to non-MIME compliant readers.

--simple boundary

This is implicitly typed plain ASCII text.

It does NOT end with a linebreak.

--simple boundary

Content-type: text/plain; charset=us-ascii

This is explicitly typed plain ASCII text.

It DOES end with a linebreak.

--simple boundary--

This is the epilogue. It is also to be ignored.

EL CORREO ELECTRÓNICO: PROTOCOLOS DE ACCESO (POP3)

Ej: POP3 PROTOCOL TCP PORT = 110

Fase de autorización

Comandos del cliente:

user: nombre de usuario**pass**: contraseña

Respuestas del servidor

+OK**-ERR**

Fase de transacción, cliente:

list: lista mensajes por número**retr**: obtiene mensajes por num.**dele**: borra**quit**Fase de actualización del servidor
(tras desconexión)

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

EL CORREO ELECTRÓNICO: PROTOCOLOS DE ACCESO (POP3)

Comandos POP3

Comando	Descripción
USER identification	Este comando permite la autenticación. Debe estar seguido del nombre de usuario, es decir, una cadena de caracteres que identifique al usuario en el servidor. El comando <i>USER</i> debe preceder al comando <i>PASS</i> .
PASS password	El comando <i>PASS</i> permite especificar la contraseña del usuario cuyo nombre ha sido especificado por un comando <i>USER</i> previo.
STAT	Información acerca de los mensajes del servidor
RETR	Número del mensaje que se va a recoger
DELE	Número del mensaje que se va a eliminar
LIST [msg]	Número del mensaje que se va a mostrar
NOOP	Permite mantener la conexión abierta en caso de inactividad
TOP <messageID> <n>	Comando que muestra <i>n</i> líneas del mensaje, cuyo número se da en el argumento. En el caso de una respuesta positiva del servidor, éste enviará de vuelta los encabezados del mensaje, después una línea en blanco y finalmente las primeras <i>n</i> líneas del mensaje.
UIDL [msg]	Solicitud al servidor para que envíe una línea que contenga información sobre el mensaje que eventualmente se dará en el argumento. Esta línea contiene una cadena de caracteres denominada <i>unique identifier listing (lista de identificadores únicos)</i> que permite identificar de manera única el mensaje en el servidor, independientemente de la sesión. El argumento opcional es un número relacionado con un mensaje existente en el servidor POP, es decir, un mensaje que no se ha borrado.
QUIT	El comando <i>QUIT</i> solicita la salida del servidor POP3. Lleva a la eliminación de todos los mensajes marcados como eliminados y envía el estado de esta acción.



EL CORREO ELECTRÓNICO: PROTOCOLOS DE ACCESO

➤ Ventajas de IMAP4:

- Permite organización en carpetas en el lado del servidor (MTA)
- Para ello, mantiene información entre sesiones (asociando *flags* a los mensajes).
- Permite la descarga de 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 cualquier cliente con HTTP.
- Seguridad: Uso extendido de HTTPS

EL CORREO ELECTRÓNICO: PROTOCOLOS DE ACCESO EJEMPLO IMAP

telnet capone.rutgers.edu 143

```
Trying 192.168.5.240...
Connected to 192.168.5.240.
Escape character is '^]'.
* OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS AUTH=LOGIN] capone.rutgers.edu
IMAP4rev1 2003.339 at Wed, 13 Apr 2005 01:38:58 -0400 (EDT)
```

Client A1 LOGIN mailtest Password

Server A1 OK [CAPABILITY IMAP4REV1 IDLE NAMESPACE MAILBOX-REFERRALS BINARY
UNSELECT SCAN SORT THREAD=REFERENCES THREAD=ORDEREDSUBJECT MULTIAPPEND]
User mailtest authenticated

Client A2 SELECT Inbox

```
* 2 EXISTS
* 2 RECENT
* OK [UIDVALIDITY 1113370837] UID validity status
* OK [UIDNEXT 3] Predicted next UID
* FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
* OK [PERMANENTFLAGS (\* \Answered \Flagged \Deleted \Draft \Seen) ]
Permanent flags
* OK [UNSEEN 1] first unseen message in /var/mail/mailtest
```

Server A2 OK [READ-WRITE] SELECT completed

Client A3 FETCH 2 BODY[HEADER]

```
* 2 FETCH (BODY[HEADER] {670}
```

Return-Path:

X-Original-To: mailtest@capone.rutgers.edu

Delivered-To: mailtest@capone.rutgers.edu



EL CORREO ELECTRÓNICO: PROTOCOLOS DE ACCESO EJEMPLO IMAP

```
Received: from node18.rutgers.edu (node18 [192.168.5.38])
        by capone.rutgers.edu (Postfix) with ESMTP id A291B2B15C
        for ; Tue, 12 Apr 2005 22:23:53 -0400 (EDT)
Received: from me?here.com (unknown [192.168.5.250])
        by node18.rutgers.edu (Postfix) with SMTP id 4653B14112
        for ; Tue, 12 Apr 2005 22:24:03 -0400 (EDT)
To: some_guru@somewhere.com
From: pp@pp.com
Subject: Forged e-mail
Message-Id: <20050413022403.4653B14112@node18.rutgers.edu>
Date: Tue, 12 Apr 2005 22:24:03 -0400 (EDT)

)
* 2 FETCH (FLAGS (\Recent \Seen))
Server A3 OK FETCH completed
Client A4 FETCH 2 BODY[TEXT]
* 2 FETCH (BODY[TEXT] {88}
Hey,
The "To:" and "From:" are non-existent, but you still get the e-mail.
bye, bye
)
Server A4 OK FETCH completed
Client A5 LOGOUT
* BYE capone.rutgers.edu IMAP4rev1 server terminating connection
Server A5 OK LOGOUT completed
Connection closed by foreign host.
```

➤ 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

Tema 5.

SERVICIOS Y PROTOCOLOS DE APLICACIÓN EN INTERNET

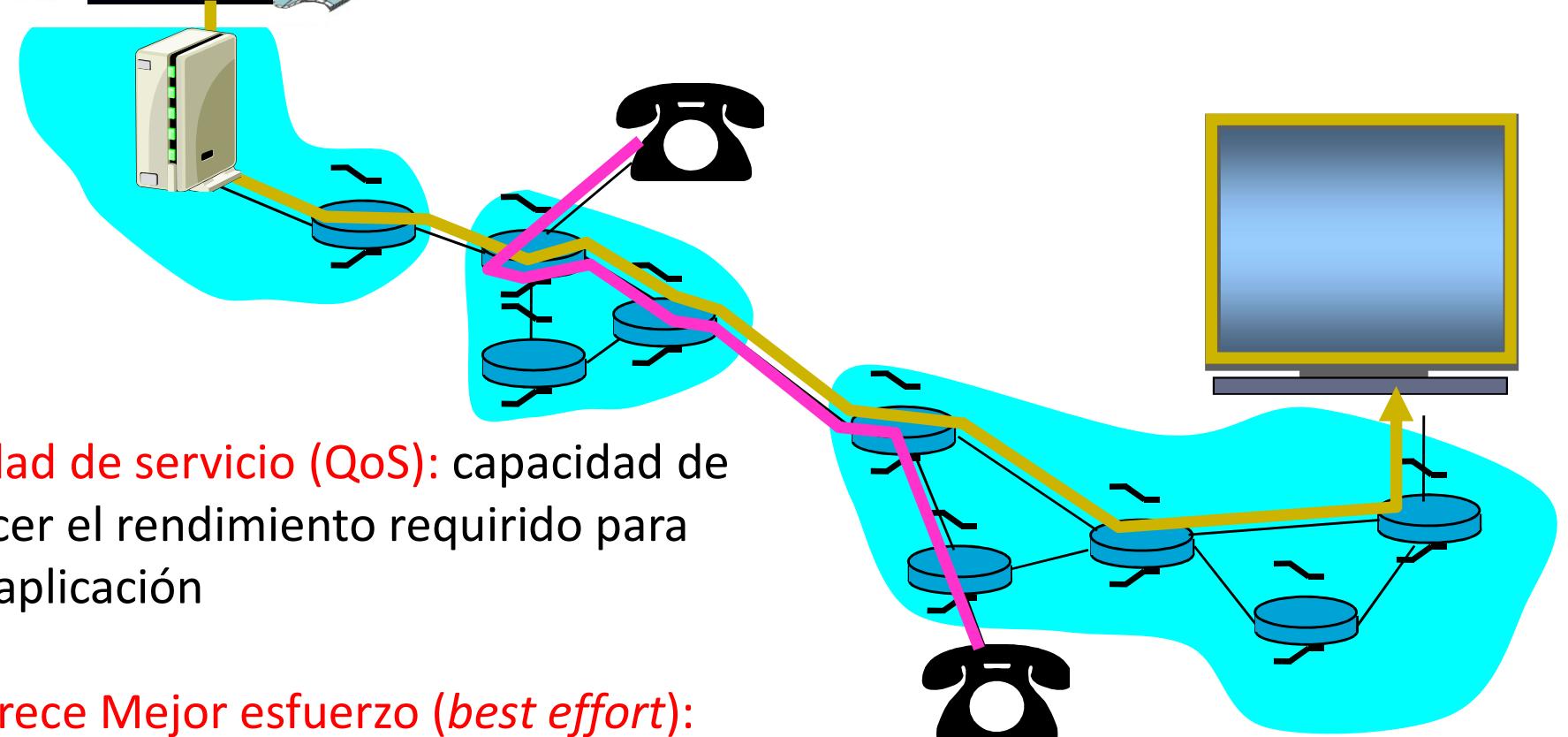
1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
3. La navegación Web (HTTP)
4. El Correo electrónico (SMTP/IMAP/POP3)
- 5. Aplicaciones multimedia**
6. Cuestiones y ejercicios

APLICACIONES MULTIMEDIA

➤ Conceptos: IP = “*tecnología de convergencia*” →



Aplicaciones Multimedia: audio, vídeo, juegos, real-time



Calidad de servicio (QoS): capacidad de ofrecer el rendimiento requerido para una aplicación

IP ofrece Mejor esfuerzo (*best effort*): sin garantías de QoS

APLICACIONES MULTIMEDIA

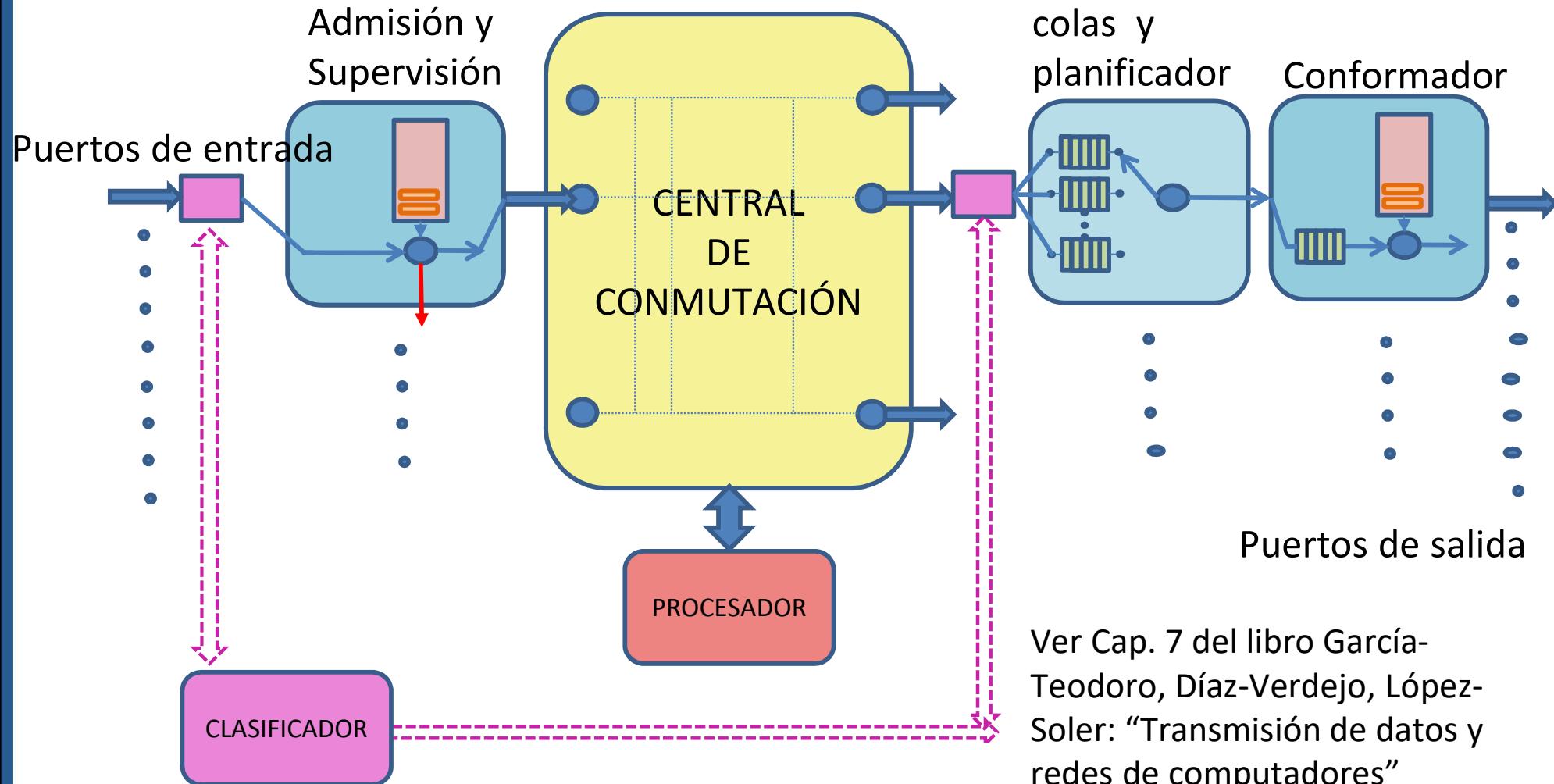
➤ **Tipos de aplicaciones**

- Flujo de audio y vídeo (*streaming*) almacenado ➔ Ej. YouTube
- Flujo de audio y vídeo en vivo ➔ Ej. emisoras de radio o IPTV
- Audio y vídeo interactivo ➔ Ej. Skype

➤ **Características fundamentales**

- Elevado ancho de banda
- Tolerantes relativamente a la pérdida de datos
- Exigen *Delay* (retardo) acotado
- Exigen *Jitter* (fluctuación del retardo) acotado
- Se pueden beneficiar de usar direcciones *multicast* (*direcciones destino de grupo*)

➤ ¿Cómo es un *router* con QoS?



Tema 5.

SERVICIOS Y PROTOCOLOS DE APLICACIÓN EN INTERNET

1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
3. La navegación Web (HTTP)
4. El Correo electrónico (SMTP/IMAP/POP3)
5. Aplicaciones multimedia
- 6. Cuestiones y ejercicios**



CUESTIONES Y EJERCICIOS

3. Discuta las características de las siguientes aplicaciones en términos de su tolerancia a la pérdida de datos, los requisitos temporales, la necesidad de rendimiento mínimo y la seguridad.

La telefonía móvil

WhatsApp

YouTube

Spotify

Comercio electrónico

CUESTIONES Y EJERCICIOS

9. Una sucursal con 50 empleados en Granada tiene una red interna basada en *FastEthernet* (100Mbps) que se conecta a Internet con una red de acceso ADSL de 0,5 Mbps de subida y 1,5 Mbps de bajada. Cada empleado, en el desempeño de su trabajo, realiza un promedio de 2000 solicitudes de información a la hora a un servidor de Base de Datos ubicado en la central del banco, en Madrid, donde cada solicitud supone el envío por parte del servidor de 10 registros de 1KB cada uno. Adicionalmente, la modificación de datos tras algunas de estas solicitudes supone el envío de 100 actualizaciones, de 10 registros, a la hora desde la sucursal al servidor. El resto de los servicios telemáticos se restringe.

- a. Calcule la velocidad de transmisión requerida. ¿Es la velocidad del enlace de acceso suficiente?
- b. ¿y si se dobla la velocidad del enlace? ¿cuál sería el tiempo de cola que esperaría en promedio cada solicitud en el enlace descendente antes de ser enviada? Considere que cada registro se envía por separado, con una cabecera de tamaño despreciable
- c. Si, alternativamente, se diseña una caché que permite evitar un 70% de los accesos a la BD ¿cuál sería el tiempo de cola que esperaría en promedio cada solicitud en el enlace descendente? ¿qué solución es mejor, la b. o esta?