Redes de Ordenadores

Nivel de Aplicación

Área de Ingeniería Telemática
Dpto. Automática y Computación
http://www.tlm.unavarra.es/

En clases anteriores...

- Introducción a Internet a alto nivel
- Introducción a los protocolos y a las arquitecturas de protocolos

En este tema:

- TCP/IP empezando desde arriba:
  el Nivel de Aplicación
  aplicaciones y protocolos sobre TCP/IP

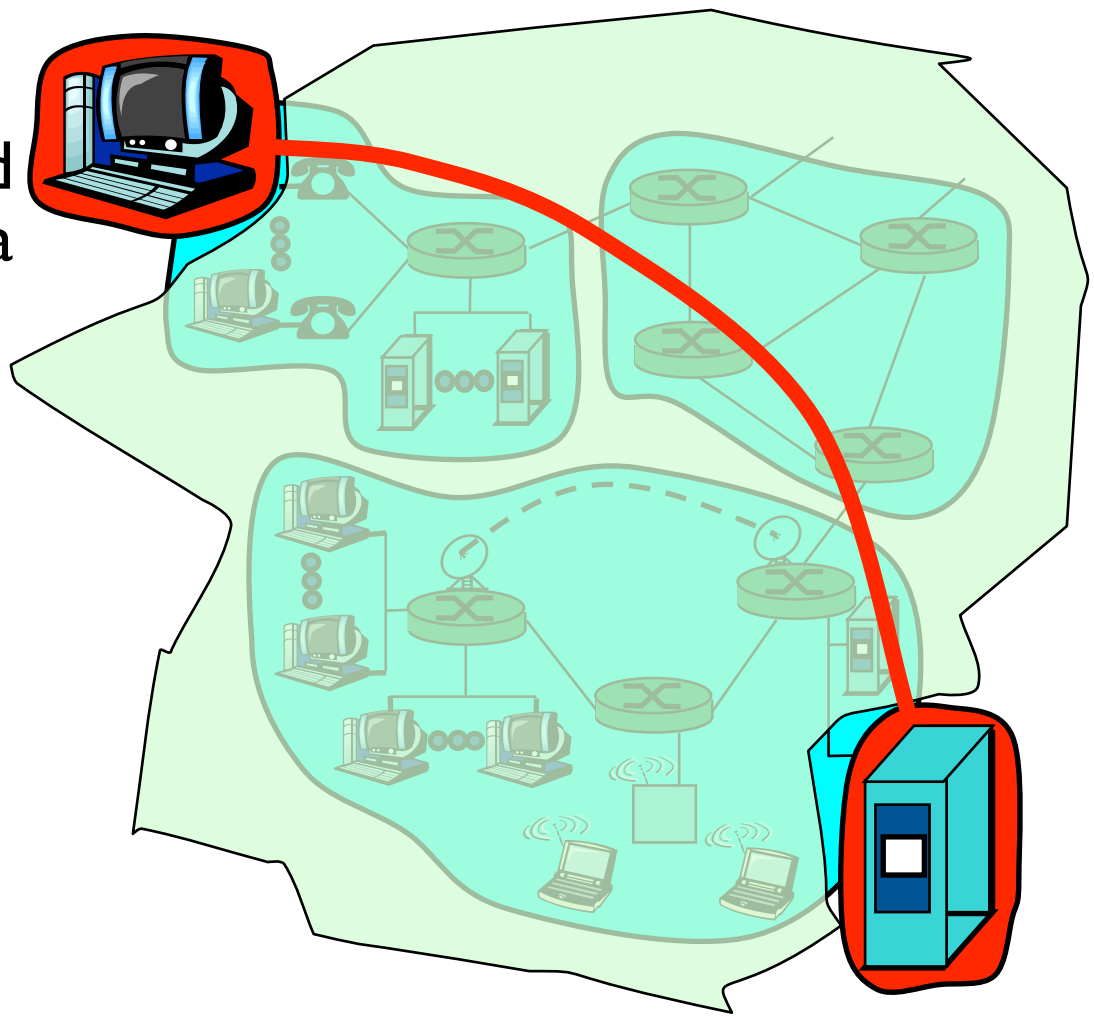
Tema 2: Nivel de Aplicación

Objetivos:

- Conceptos detrás de los protocolos de aplicación
  - Paradigma cliente-servidor
  - Paradigma peer-to-peer
  - Servicios de nivel de transporte
- Aprender sobre protocolos analizando protocolos de servicios populares
  - HTTP
  - FTP
  - SMTP / POP3
  - DNS
- Programación de aplicaciones de red
  - API de sockets

TCP/IP: Servicios ofrecidos

- Los hosts emplean Internet para comunicarse
- Los elementos de la red forman una "caja negra" para las aplicaciones...
- La red ofrece dos servicios de comunicaciones:
  - Fiable orientado a conexión
  - No fiable sin conexión



TCP: Orientado a conexión

Objetivo: Transferir datos entre hosts

Establecimiento (*handshaking*):

Intercambio de paquetes de control antes que los de datos

- Como el "Hola, hola"
- *Establece un "estado"* en los dos host pero no en la red = *orientado* a conexión

- TCP

  Transmission Control Protocol

  - Protocolo que ofrece en Internet el servicio orientado a conexión

TCP [RFC 793]

- Transferencia *fiable y en orden* de un flujo (stream) de datos
  - ¿Pérdidas?: confirmaciones y retransmisiones
- *Control de flujo:*
  - El emisor no saturará al receptor
- *Control de congestión:*
  - El emisor "reduce la velocidad a la que envía" cuando la red se congestiona

Aplicaciones que usan TCP:

- HTTP (Web), FTP (transferencia de ficheros), Telnet (login remoto), SMTP (email)

UDP: Servicio sin conexión

Objetivo: Transferir datos entre hosts

- ¡El mismo de antes!

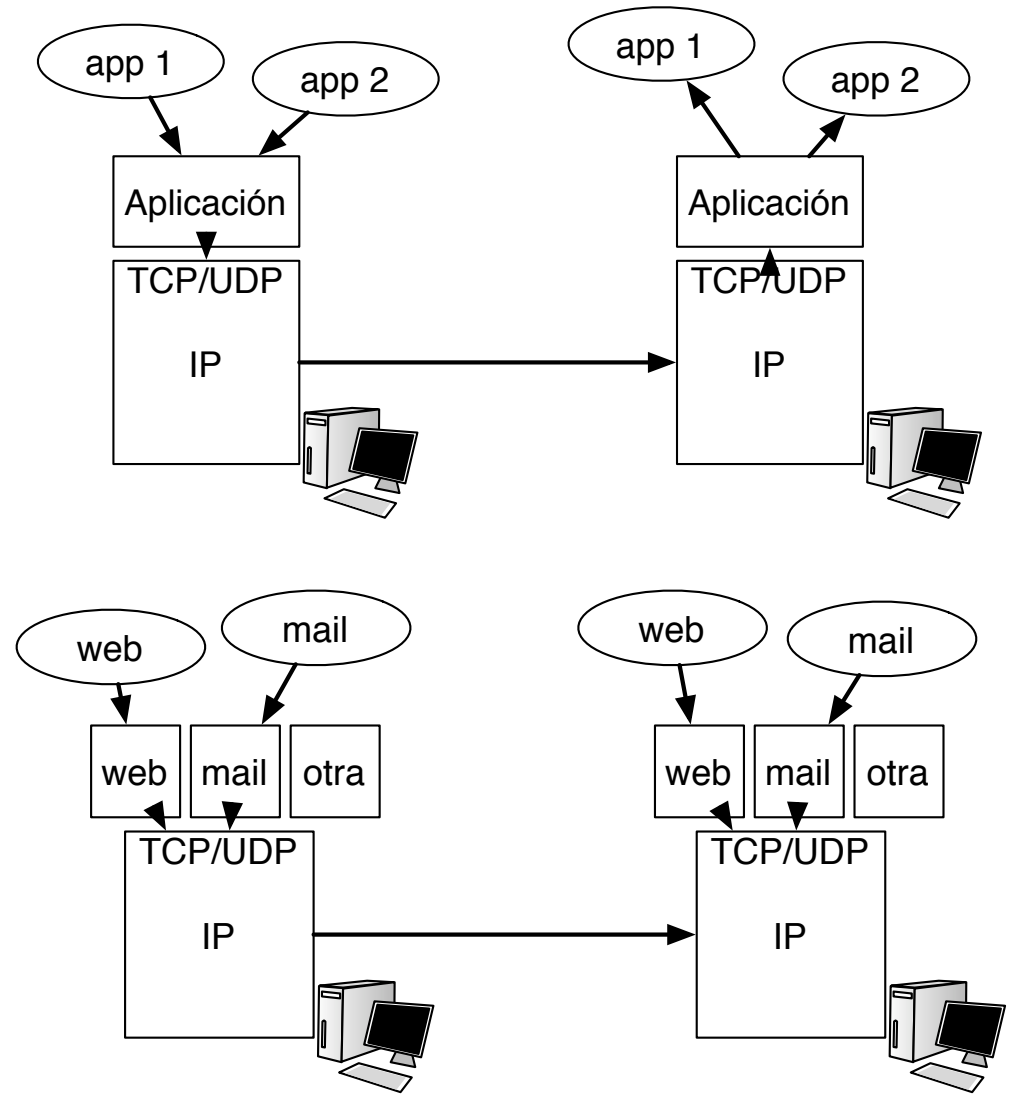
- UDP - User Datagram Protocol [RFC 768]:
  - Sin conexión
  - No fiable
  - Sin control de flujo
  - Sin control de congestión

Aplicaciones que usan UDP:

- Streaming, teleconferencia, DNS, telefonía por Internet

Nivel de Aplicación en Internet

- No es un nivel bien definido y uniforme
- Cada servicio tiene su propio nivel de aplicación para comunicarse con las entidades de ese servicio



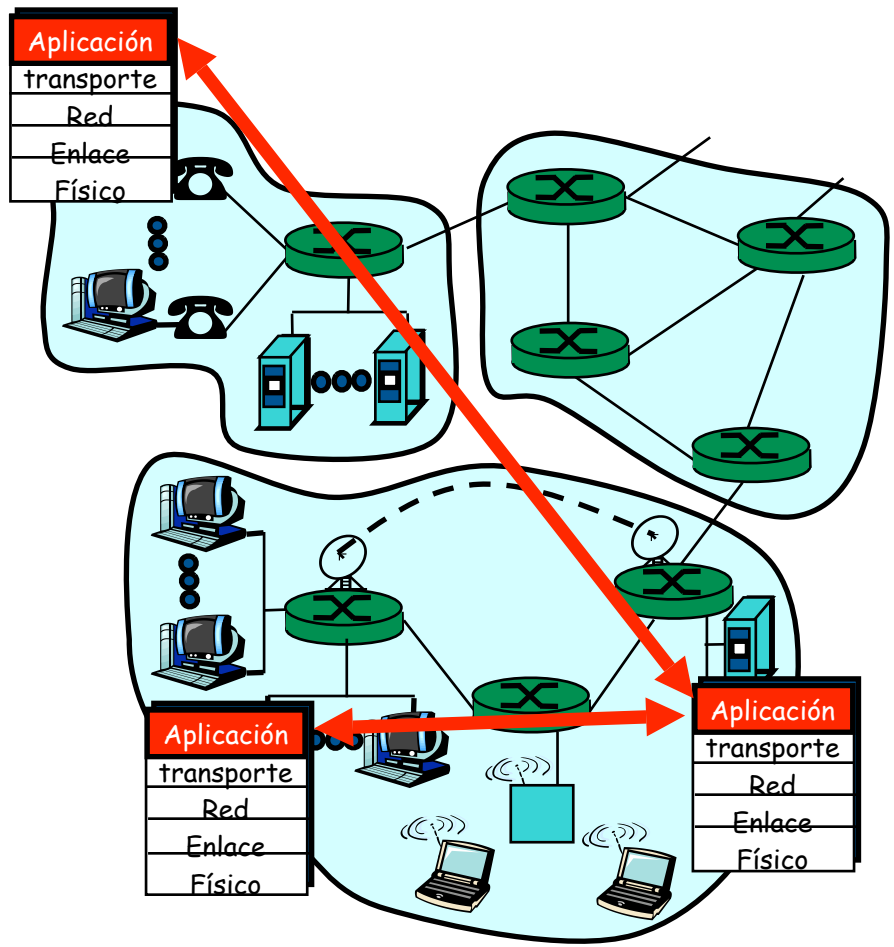
Algunas aplicaciones en red

- E-mail
- Web
- Mensajería instantánea
- login remoto
- Compartición de ficheros P2P
- Juegos multiusuario en red
- Streaming de video clips
- Telefonía por Internet
- Videoconferencia en tiempo real
- Computación masiva en paralelo

Aplicaciones en red

Las aplicaciones

- Son software
- Diferentes máquinas y Sistemas Operativos
- Quienes se comunican son procesos
- IPC: Inter Process Communication
- Nos interesan procesos ejecutándose en diferentes máquinas
- Se comunican a través de una red
- Intercambian mensajes
- Emplean Protocolos de nivel de aplicación...



Aplicaciones y Protocolos

Los Protocolos de aplicación son una parte de las aplicaciones de red... ...

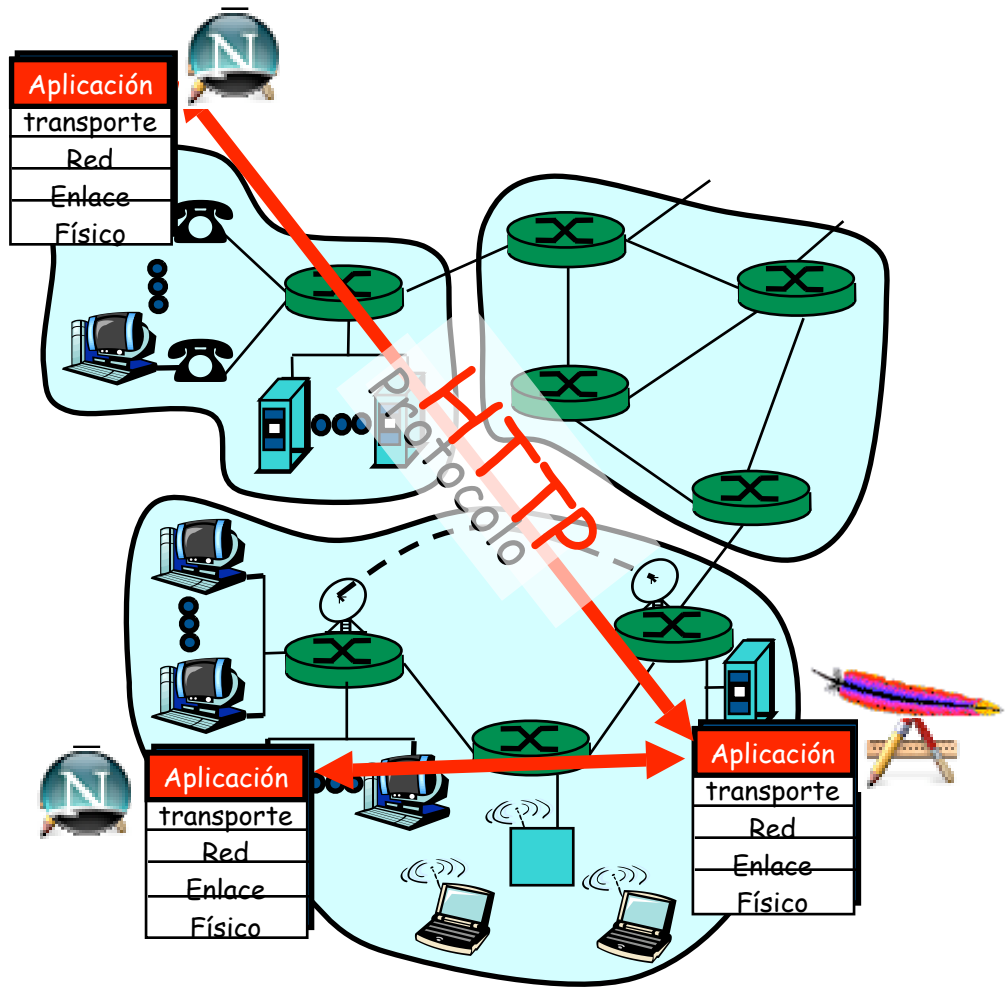
Definen:

- Tipos de mensajes
- Sintaxis/formato de mensajes
- Significado del contenido
- Reglas de funcionamiento

Ejemplo: La Web

- Navegador, Servidor Web...
- HTTP ...

Muchos protocolos son estándares abiertos (en RFCs)

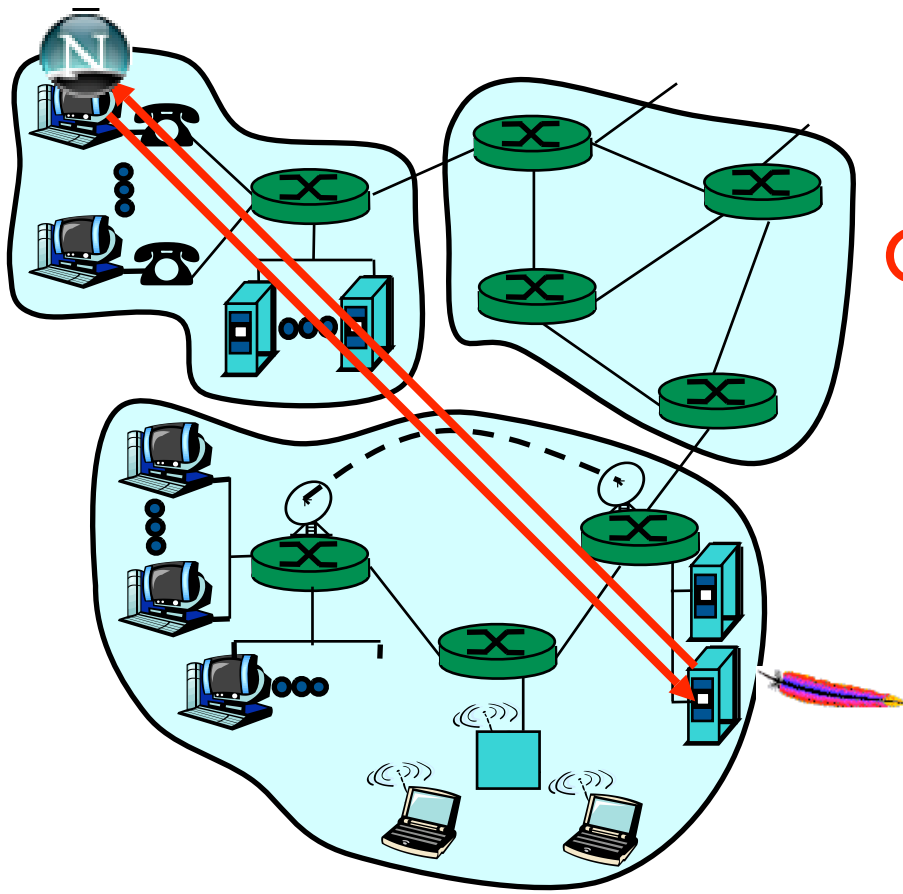


Paradigmas

Filosofías para escribir/organizar las aplicaciones distribuidas

- **Cliente-servidor**
  - Asimetría, hay proveedores de servicios y usuarios de los servicios
- **Peer-to-peer (P2P)**
  - Simetría, comunicación entre iguales (pares)
- *Híbrido de cliente-servidor y P2P*
  - Una aplicación puede usar las dos filosofías para diferentes cosas

Arquitectura cliente-servidor



Servidor:

- Comienza a ejecutarse primero...
- Espera a ser contactado
- Host siempre disponible
- Dirección permanente

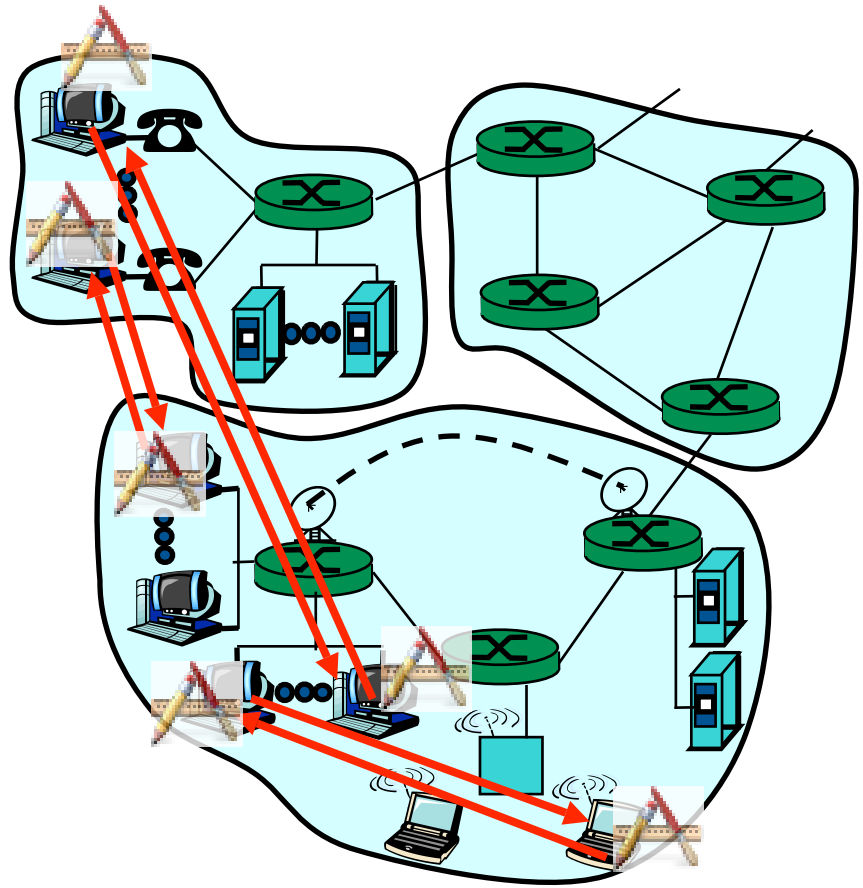
Cliente:

- Lanzado más tarde por el usuario...
- Inicia la comunicación con un servidor...
- No con clientes
- Termina cuando el usuario deja de usarlo
- Puede no tener siempre la misma dirección
- Ejemplos: casi todos los servicios clásicos

  Web, mail, FTP, News, IRC, Streaming...

Arquitectura Peer-to-Peer

- No hay un servidor siempre disponible
- Hosts extremos cualesquiera se comunican (peers)...
- Pueden no estar siempre conectados...
- Los peers pueden cambiar de dirección
- El mismo proceso puede ser cliente o servidor
- Ejemplo: Gnutella



Escalable

Difícil de controlar

Híbrido de cliente-servidor y P2P

Napster (y eMule y similares...)

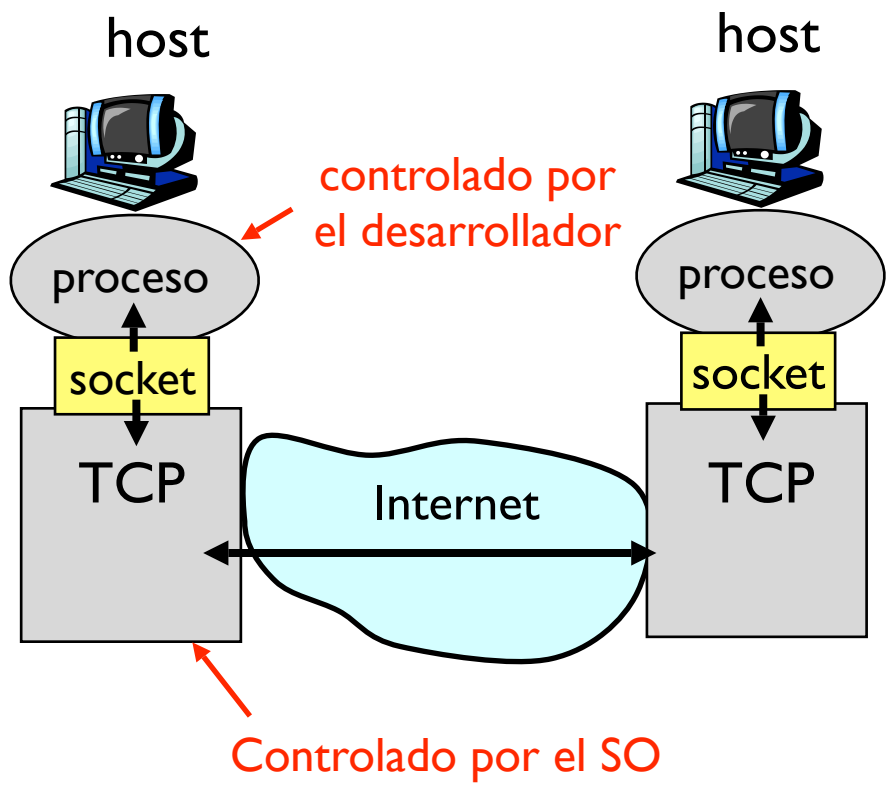
- Transferencia de ficheros P2P
- Búsqueda de ficheros centralizada:
  - Peers registran el contenido ofrecido en un servidor central
  - Peers preguntan al mismo servidor para buscar ficheros

Mensajería Instantánea (Instant messaging=IM)

- Conversación entre dos usuarios es P2P
- Detección de presencia y localización centralizada:
  - Los usuarios registran su dirección en un servidor central cuando se conectan a la red
  - Contactan con el servidor central para encontrar la dirección actual de sus contactos

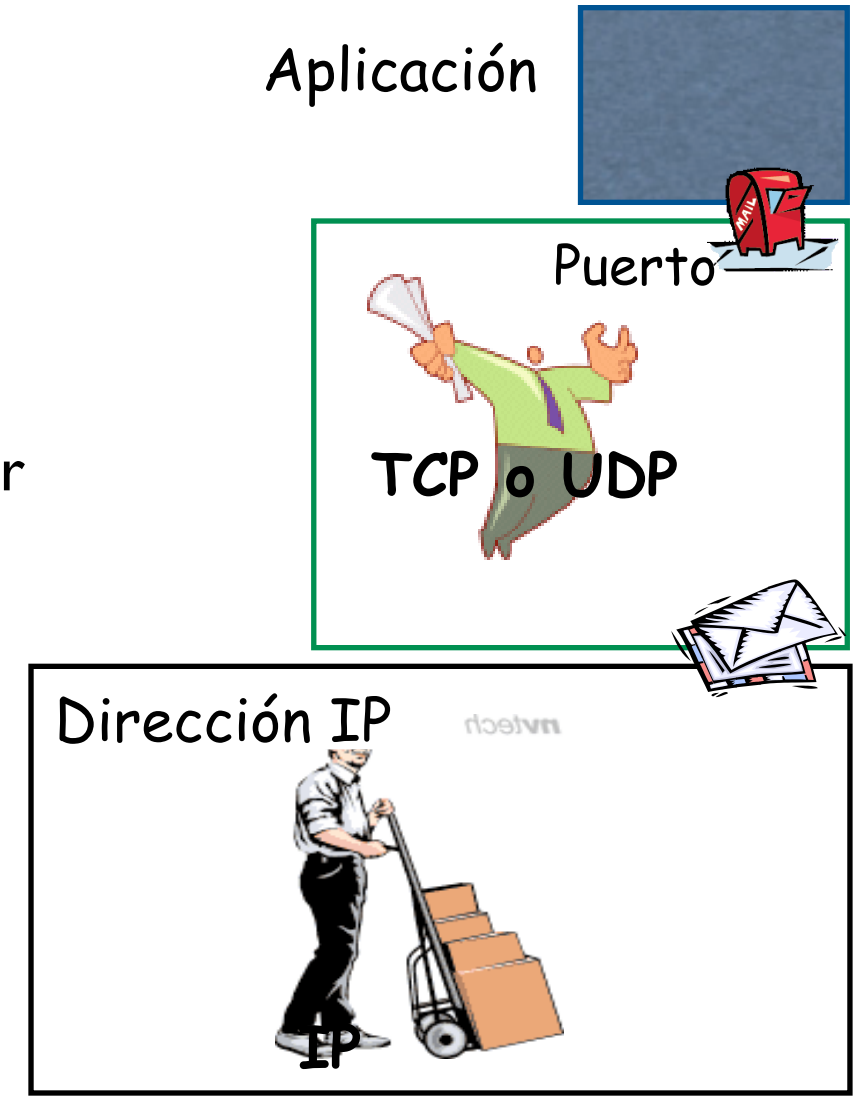
Sockets

- Los procesos envían y reciben mensajes a través de un socket
- Delega en el nivel de transporte para que haga llegar los mensajes al otro socket
- Acceso a través de un API
- Puede escoger el protocolo de transporte
- Puede configurar algunos parámetros del mismo
- No controla cómo se comporta



Identificando al proceso

- El emisor de un mensaje debe identificar al host receptor
- Un host (interfaz) tiene una dirección IP única (32 bits)
- Muchos procesos en el mismo host
- Debe identificar al proceso receptor que corre en ese host
- Número de puerto diferente asociado a cada proceso
- Ejemplos:
  - Servidor Web: puerto TCP 80
  - Servidor e-mail: puerto TCP 25



Servicios que necesitan las aplicaciones

Pérdidas

- Algunas apps soportan pérdidas (ej. audio)
- Otras requieren 100% de fiabilidad (ej. transferencia de ficheros)

Retardo

- Algunas apps requieren bajo retardo (ej. juegos en red)

Ancho de banda

- Algunas apps requieren un mínimo de ancho de banda (ej. audioconf)
- Otras (elásticas) funcionan con cualquier cantidad pero pueden sacar provecho a todo el disponible

Requisitos de aplicaciones comunes

Aplicación	Pérdidas	Ancho de banda	Retardo
Transf. ficheros	ninguna	elastico	no
e-mail	ninguna	elastico	no
Web	ninguna	elastico	no
audio/vídeo en RT	soporta	audio: 5kbps-1Mbps vídeo:10kbps-5Mbps	sí, 100's mseg
audio/vídeo diferido	soporta	idem	sí, unos segs
juegos interactivos	soporta	desde unos kbps	sí, 100's mseg
IM	ninguna	elastico	sí y no

Servicios ofrecidos por protocolos de transporte en Internet

TCP:

- orientado a conexión: establecimiento requerido entre ambos procesos
- transporte fiable: sin pérdidas
- control de flujo: el emisor no saturará al receptor
- control de congestión: limita el envío cuando la red está sobrecargada
- no ofrece: límite al retardo, mínimo ancho de banda garantizado

UDP:

- Transferencia de datos no fiable entre los dos procesos
- No ofrece: conexión, fiabilidad, control de flujo, control de congestión, límite al retardo ni ancho de banda garantizado

Aplicaciones de Internet: protocolos de aplicación y transporte

Aplicación	Protocolo de nivel de aplicación	Protocolo de nivel de transporte
e-mail	SMTP [RFC 2821]	TCP
acceso remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
transferencia de fichero	FTP [RFC 959]	TCP
streaming	Suele ser propietario (ej. RealNetworks)	TCP o UDP
Telefonía en Internet	Suele ser propietario (ej., Dialpad)	típicamente UDP

Servicios de Internet

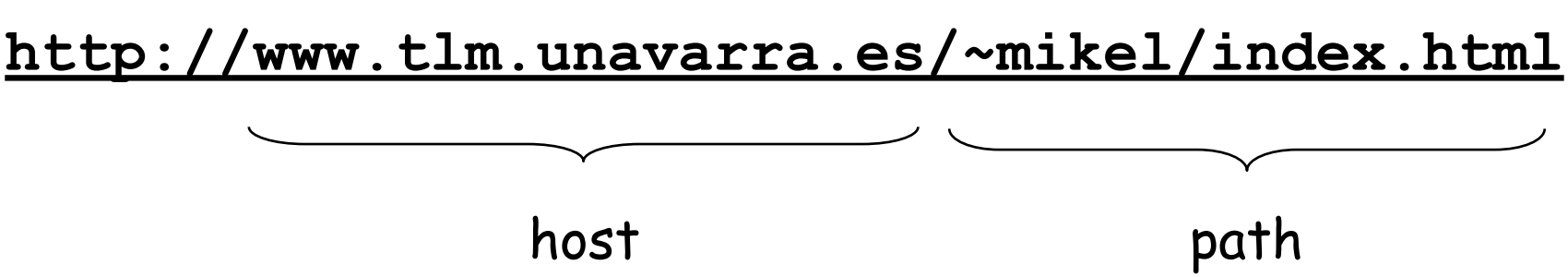
Objetivos:

- Aprender con el ejemplo: Funcionamiento de protocolos de nivel de aplicación
  - **Web y HTTP**
  - **DNS**
  - **SMTP/POP3**
  - **Telnet**
  - **FTP**
  - **P2P**

Web y HTTP

Términos

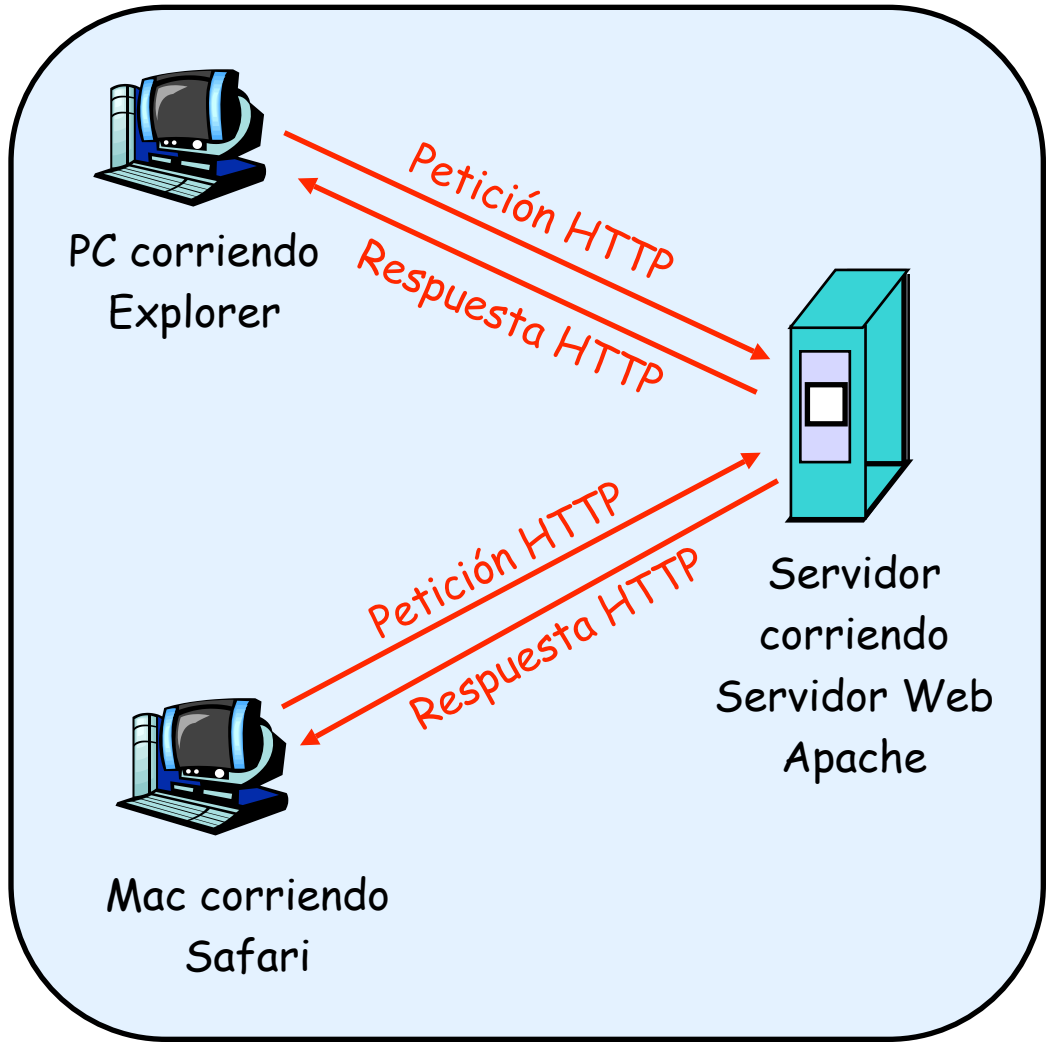
- Una Página Web está compuesta por objetos
- Un objeto puede ser un fichero HTML, una imagen JPEG, un applet JAVA, un fichero de sonido...
- La página Web está compuesta por un fichero HTML base que hace referencia a otros objetos
- Se hace referencia a cada objeto mediante un URL
- Ejemplo de URL:



HTTP

HTTP: HyperText Transfer Protocol

- Protocolo de nivel de aplicación de la Web
- Modelo cliente/servidor
  - *cliente:* browser (nevegador) que solicita, recibe y muestra objetos de la Web
  - *servidor:* el servidor Web envía objetos en respuesta a peticiones
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



HTTP

Usa TCP:

- El cliente inicia una conexión TCP al servidor, puerto 80
- El servidor acepta la conexión TCP del cliente
- Cada uno tiene un socket conectado con el otro
- Se intercambian mensajes HTTP entre el navegador y el servidor Web
- Se cierra la conexión TCP

HTTP es "sin estado"

- El servidor no mantiene ninguna información de peticiones anteriores del cliente

Nota

Los protocolos que mantienen "estado" son complejos

- Debe mantener la historia pasada (estado)
- Si el cliente/servidor falla, el estado entre ambos puede volverse incoherente

Empleo de las conexiones

HTTP no persistente

- En cada conexión TCP se envía como máximo un objeto
- HTTP/1.0

HTTP persistente

- En la misma conexión TCP se pueden enviar varios objetos entre el servidor y el cliente
- HTTP/1.1, funcionamiento por defecto

HTTP no persistente

Supongamos que el usuario solicita el URL

www.tlm.unavara.es/~mikel/index.html

(contiene texto y 1 referencia a una imagen JPEG)

1a. El cliente HTTP inicia la conexión TCP con el (proceso) servidor de HTTP en www.tlm.unavarra.es puerto 80

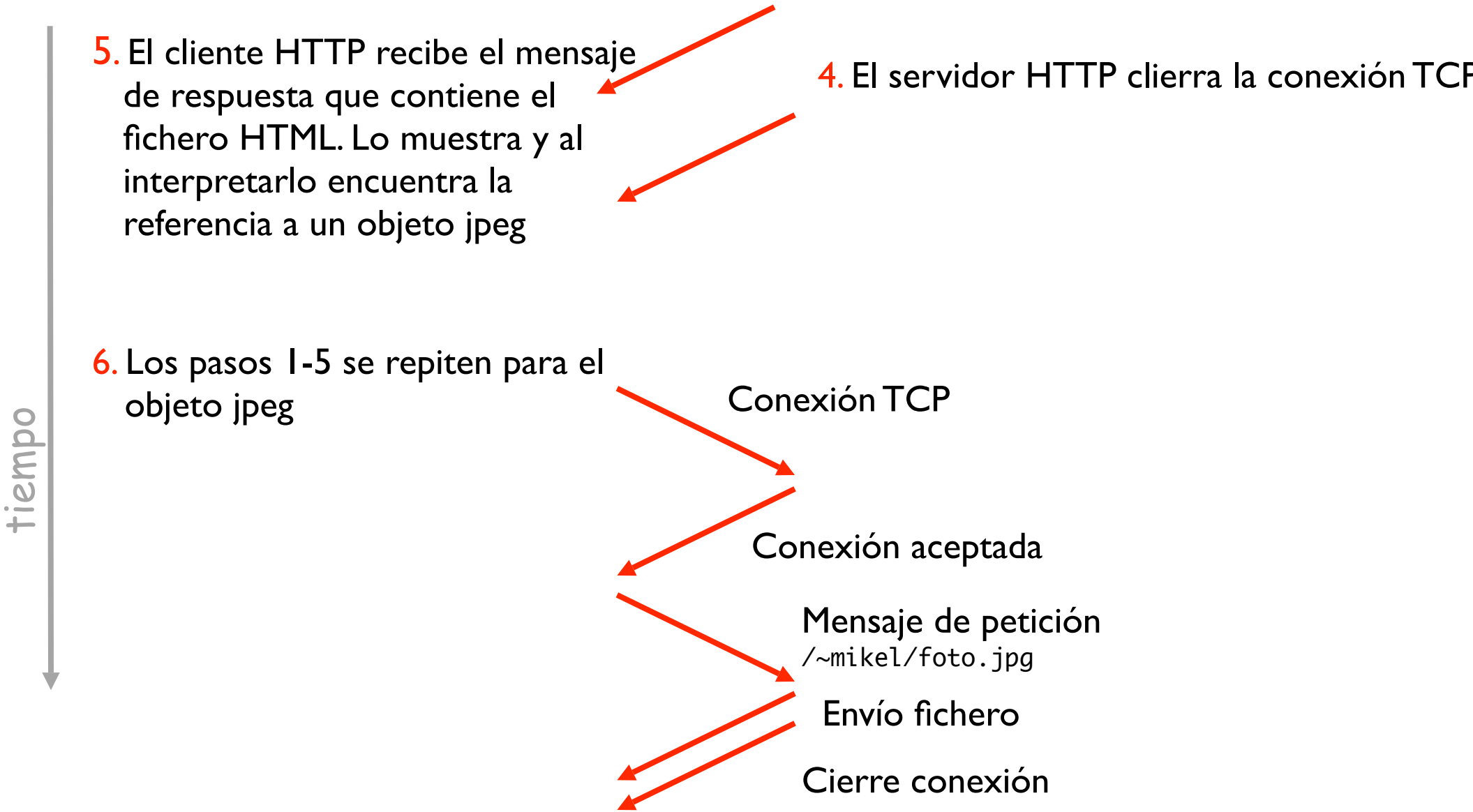
1b. El servidor HTTP en el host www.tlm.unavarra.es espera conexiones al puerto 80. Acepta la conexión, notificando al cliente

2. El cliente HTTP envía un mensaje de petición (contiene el URL) a través de la conexión TCP (empleando el socket). El mensaje indica que el cliente quiere el objeto /~mikel/index.html

3. El servidor HTTP recibe el mensaje de petición, forma un mensaje de respuesta que contiene el objeto solicitado y lo envía a través de su socket

tiempo

HTTP no persistente



Modelo del tiempo de respuesta

Definición de RTT:

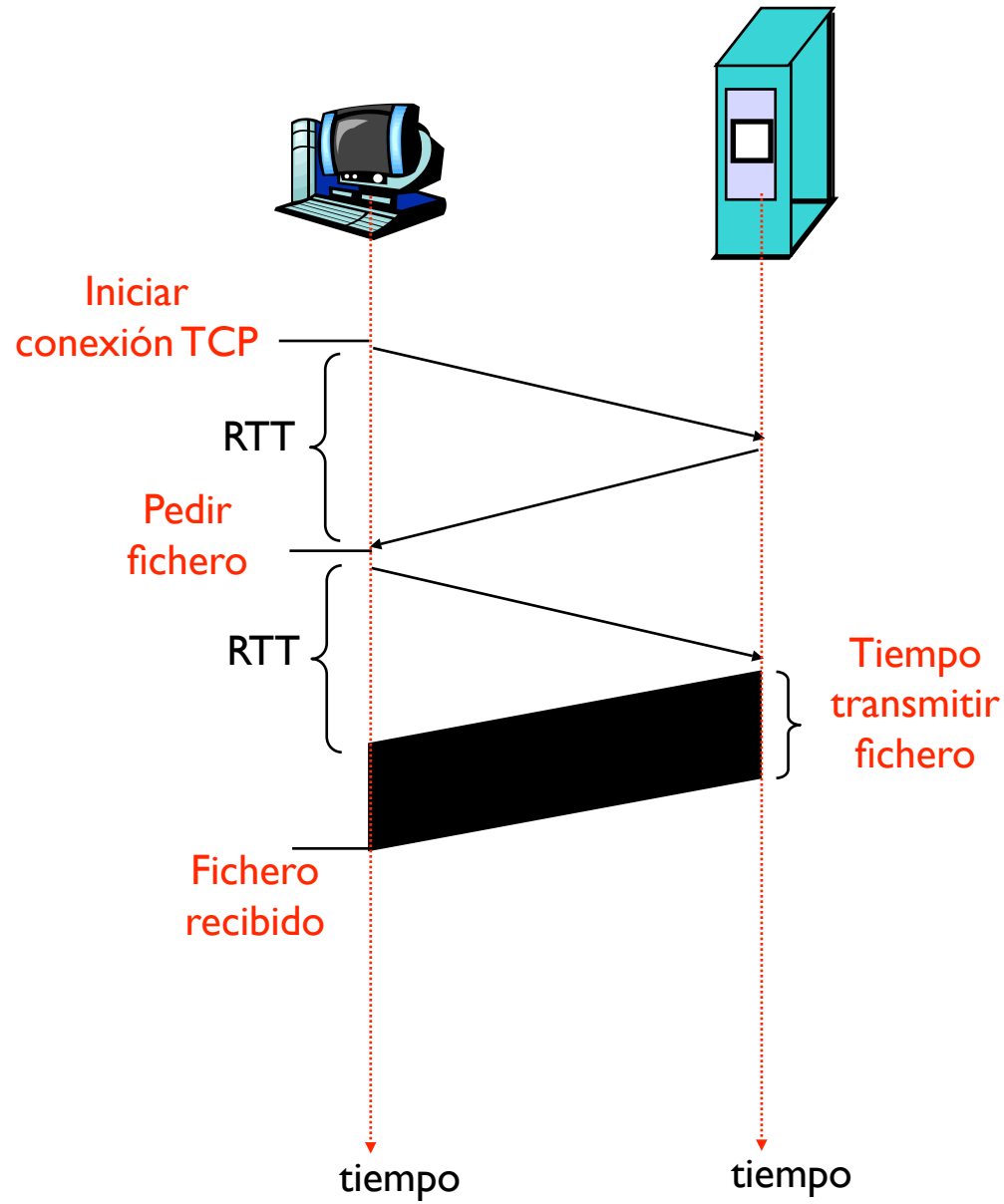
Round Trip Time

tiempo para que un paquete pequeño viaje de cliente a servidor y vuelta

Tiempo de respuesta:

- Un RTT para iniciar la conexión
- Un RTT para la petición HTTP y el comienzo de la respuesta
- Tiempo de transmisión del fichero

total = 2RTT+tiempo transmisión



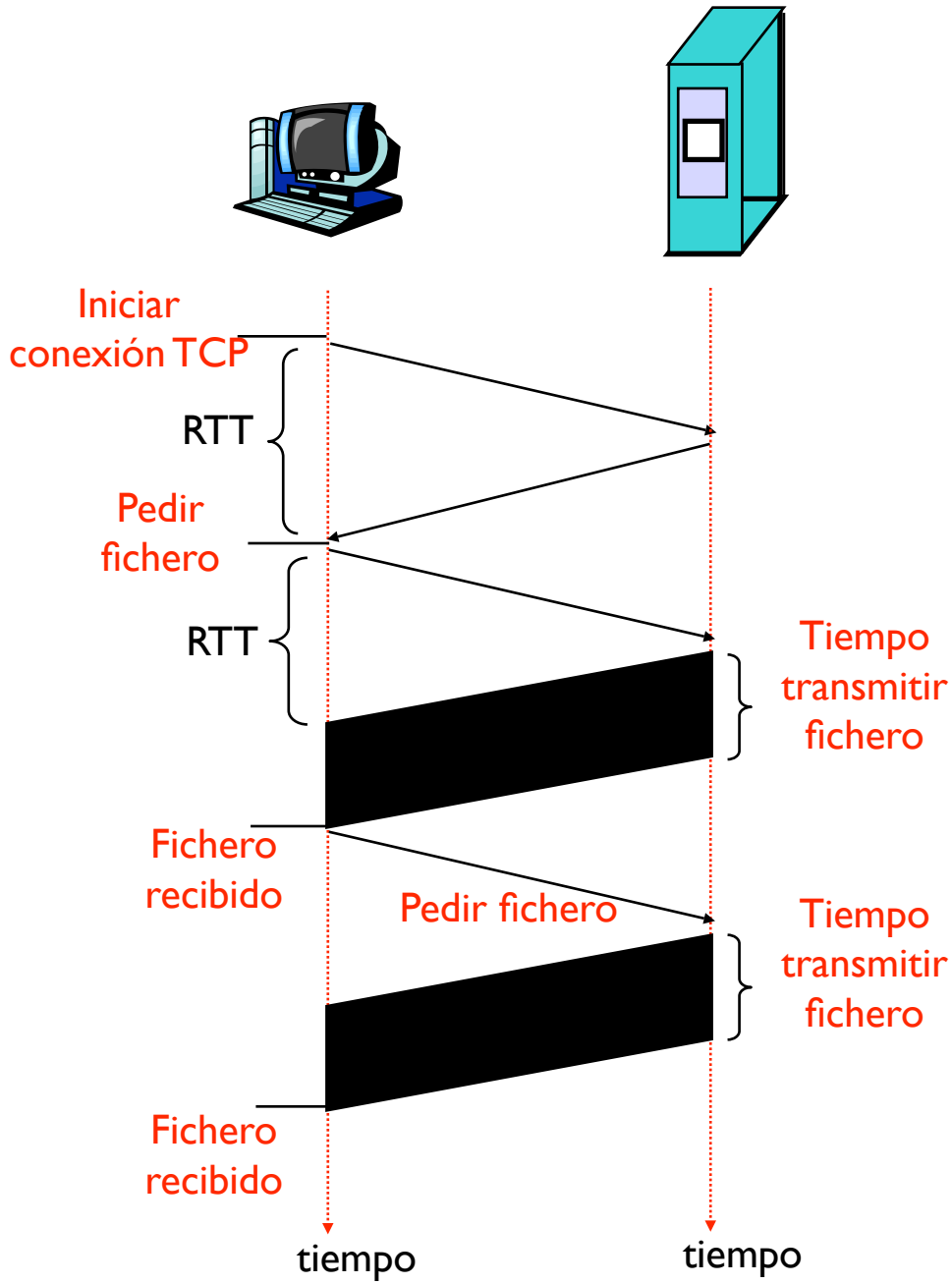
HTTP persistente

Con HTTP no persistente:

- Requiere 2 RTTs por objecto
- OS debe reservar recursos para cada conexión TCP
- Pero el navegador suele abrir varias conexiones TCP en paralelo

HTTP persistente:

- El servidor deja la conexión abierta tras enviar la respuesta
- Los siguientes mensajes HTTP entre cliente y servidor van por la misma conexión



HTTP persistente

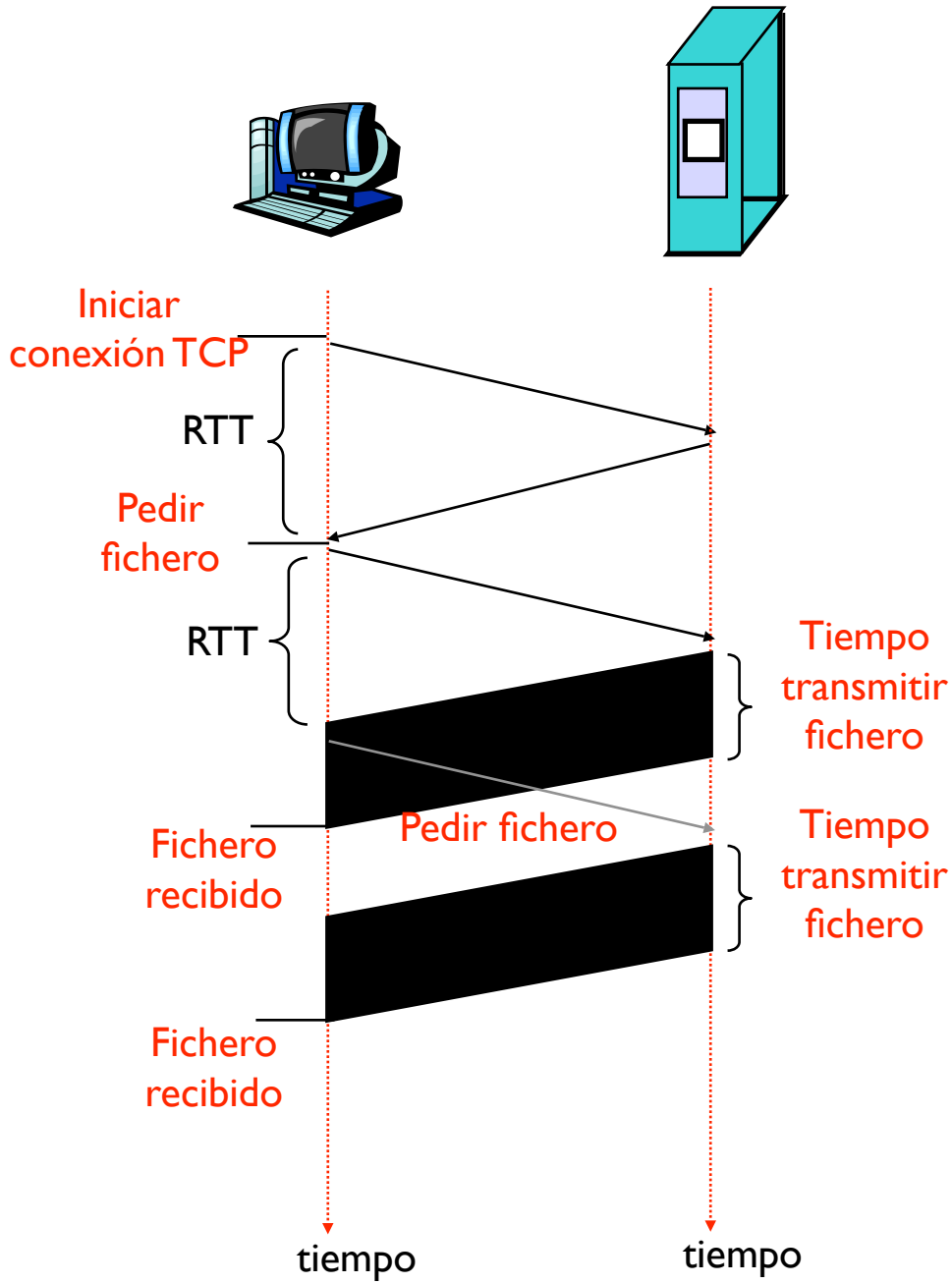
Persistente sin pipelining:

- El cliente manda la nueva petición cuando ha terminado de recibir la respuesta anterior
- Al menos un RTT por cada objeto

Persistente con *pipelining*:

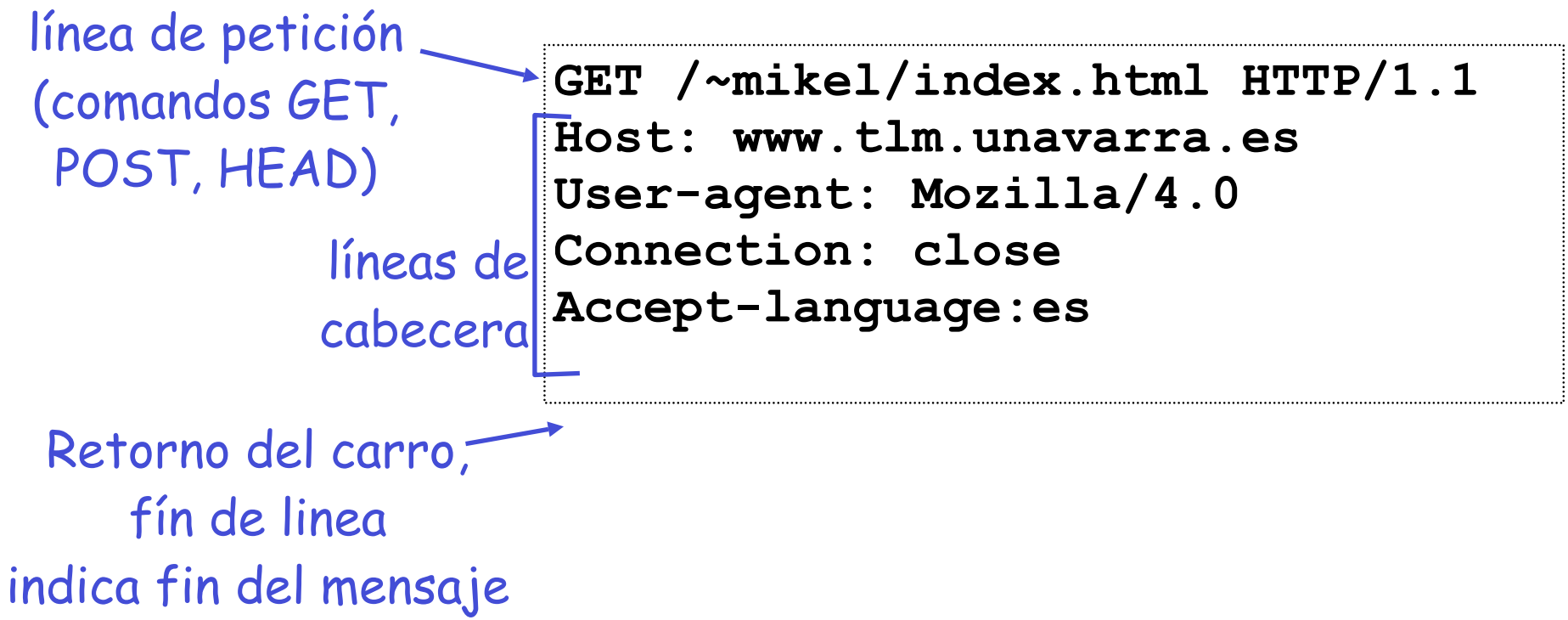
» *default* en HTTP/1.1

- El cliente envía petición tan pronto como encuentra una referencia a objeto
- Solo un RTT para todos los objetos referenciados en la página base

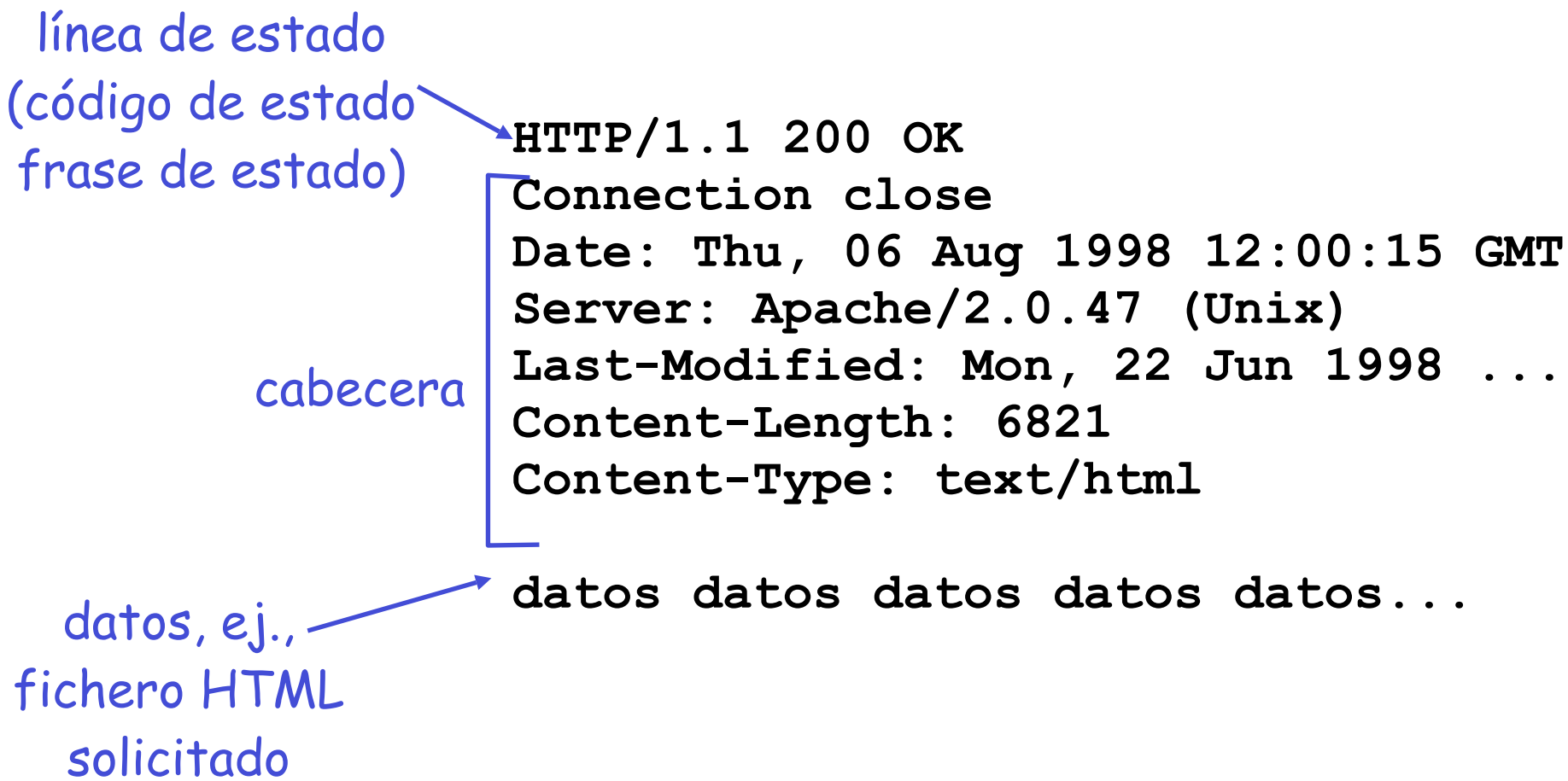


HTTP request message

- Dos tipos de mensajes messages: *request, response*
- Mensaje HTTP request :
  - ASCII (formato legible por humanos)



HTTP response message



Probando HTTP desde el cliente

- Haga telnet a su servidor Web favorito:

```
$ telnet www.tlm.unavarra.es 80
```

Abre una conexión TCP al puerto 80 (puerto por defecto del servidor HTTP) de www.tlm.unavarra.es
Lo que se escriba se envía por la conexión TCP

- Escriba una petición GET de HTTP:

```
GET /~mikel/ HTTP/1.1
Host: www.tlm.unavarra.es
```

Escribiendo esto (y retorno del carro dos veces) se envía un petición HTTP 1.1 mínima pero completa al servidor

- Vea el mensaje de respuesta del servidor

Servicios de Internet

Objetivos:

- Aprender con el ejemplo: Funcionamiento de protocolos de nivel de aplicación
  - **Web y HTTP** ✓
  - **DNS**
  - **SMTP/POP3**
  - **Telnet**
  - **FTP**
  - **P2P**

El problema de los nombres

- Las direcciones IP, que identifican a los interfaces de los hosts, son números de 32 bits
- Sencillas de manejar para las máquinas, complicado para los humanos
- Más sencillo memorizar nombres textuales
- Hace falta "traducir" el nombre textual en la dirección numérica para que se pueda realizar la comunicación. Esto se llama "resolver el nombre"
- La traducción se realiza mediante el Sistema de Nombres de Dominio o DNS (Domain Name System)

Domain Name System

- Es una base de datos distribuida con servidores de nombres organizados jerárquicamente
- Es un protocolo de aplicación que permite a los hosts traducir entre nombres y direcciones
  - Funcionalidad vital implementada como protocolo a nivel de aplicación
  - Complejidad en los extremos de la red

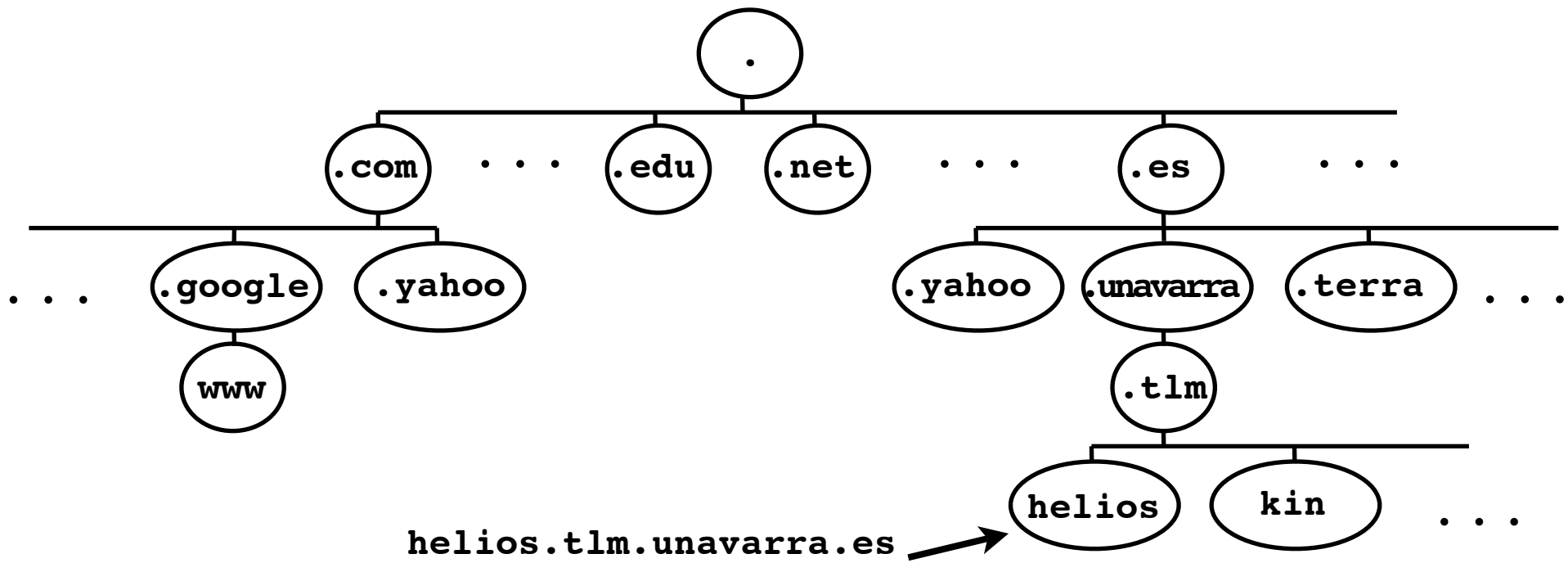
¿Por qué no centralizado?

- Punto de fallo
- Volumen de tráfico
- Base de datos centralizada lejana
- Mantenimiento

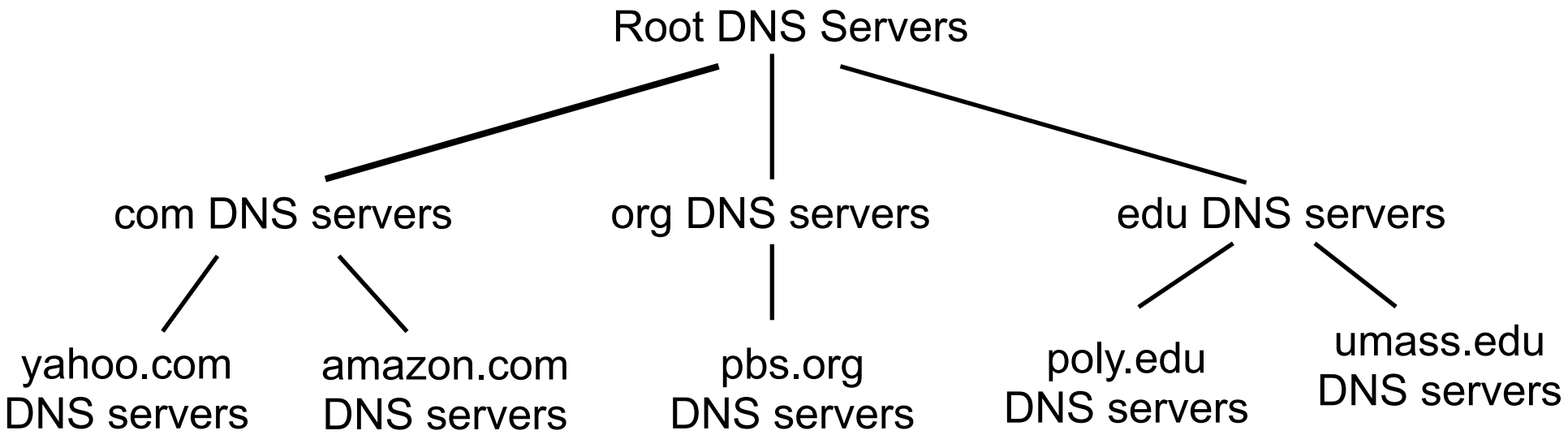
No escala!

Jerarquía de nombres

- Los nombres están formados por segmentos alfanuméricos separados por puntos (no distingue mayúsculas)
  - helios.tlm.unavarra.es
  - www.google.com
- Estructura jerárquica



Base de datos jerárquica distribuida



El cliente busca la IP de www.amazon.com, 1ª aproximación:

- El cliente pregunta a un servidor Root para encontrar el servidor de DNS del dominio com
- El cliente pregunta al servidor del dominio *com* para obtener el servidor del dominio amazon.com
- El cliente pregunta al servidor DNS del dominio amazon.com para obtener la IP de www.amazon.com

Implementación

- El servidor es un programa específico pero el cliente es generalmente solo unas funciones en una librería (*resolver*)...
- La aplicación cliente de DNS es la propia aplicación del usuario...
- El software típico que lo implementa es BIND (Berkeley Internet Name Domain) (el programa servidor se llama `named`)...

Funcionamiento

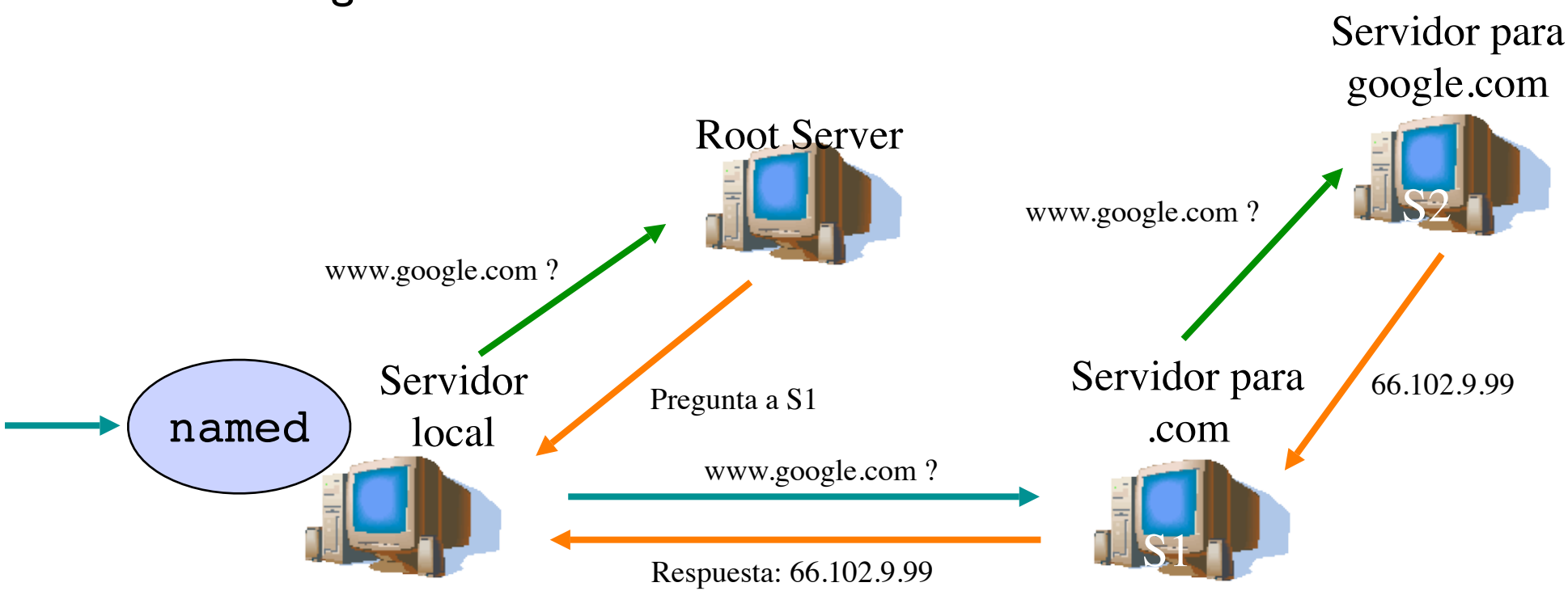
- Mensajes sobre UDP puerto 53
  (TCP puerto 53 para peticiones largas)
- Cada ISP posee un servidor de nombres local...
- Los hosts tienen configurado a su servidor local
- Cuando un host desea resolver un nombre hace la petición a su servidor local el cual le devuelve la respuesta...

Funcionamiento

- ¿Cómo conoce la respuesta el servidor local?
  - Si es el servidor autoritario (authoritative server) para el dominio en el que está esa máquina él tiene la porción de la base de datos distribuida en la que está el mapeo
  - Si no lo es preguntará a un Root Server

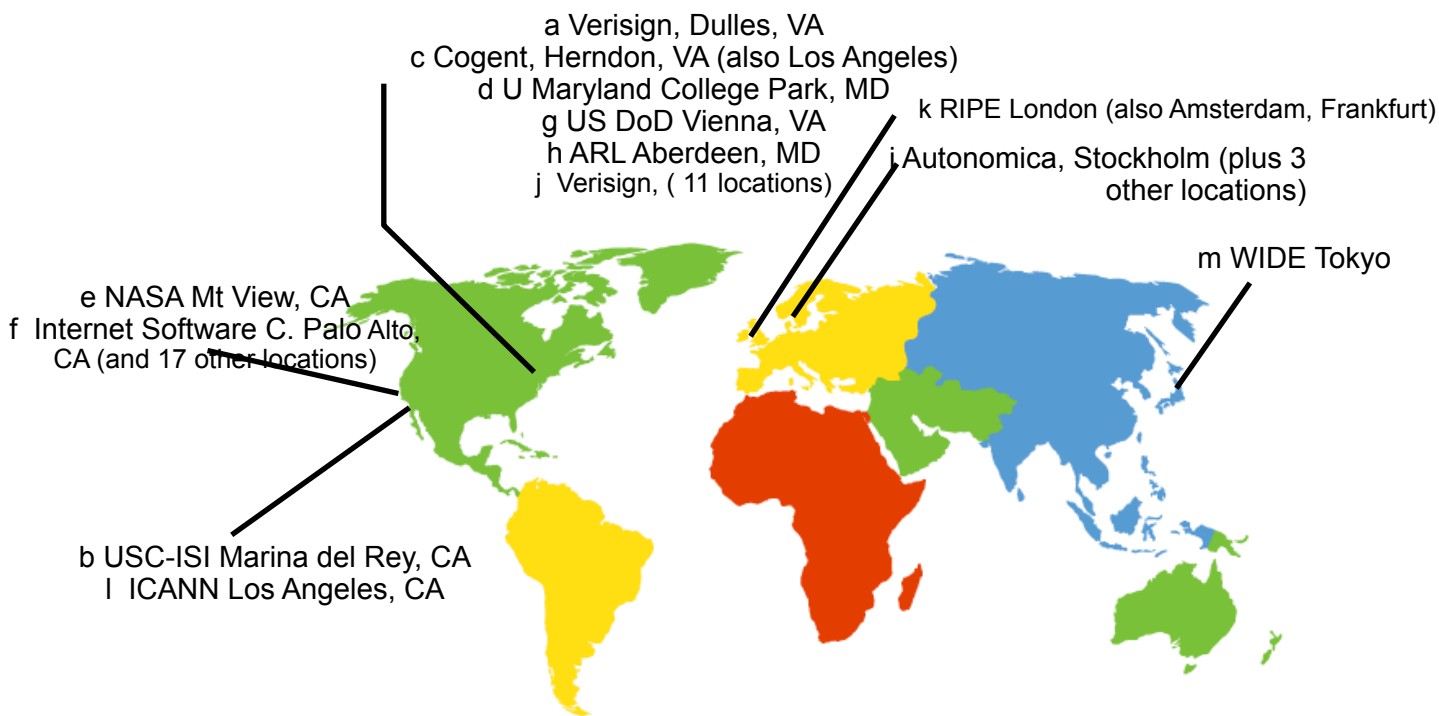
Funcionamiento

- El servidor local pregunta a un Root Server...
- Éste le devuelve la dirección de un servidor intermedio (petición iterativa)...
- El Servidor local hace una petición recursiva a ese servidor...
- Ese servidor continuará haciendo la petición (recursiva) hasta que llegue un servidor autoritario . . .
- Todas las peticiones son recursivas menos la petición al Root Server para reducir la carga sobre los Root



DNS: Root name servers

- 13 en el mundo
- En el fichero de configuración de cada servidor de DNS



Servicios de Internet

Objetivos:

- Aprender con el ejemplo: Funcionamiento de protocolos de nivel de aplicación
  - **Web y HTTP** ✓
  - **DNS** ✓
  - **SMTP/POP3**
  - **Telnet**
  - **FTP**
  - **P2P**

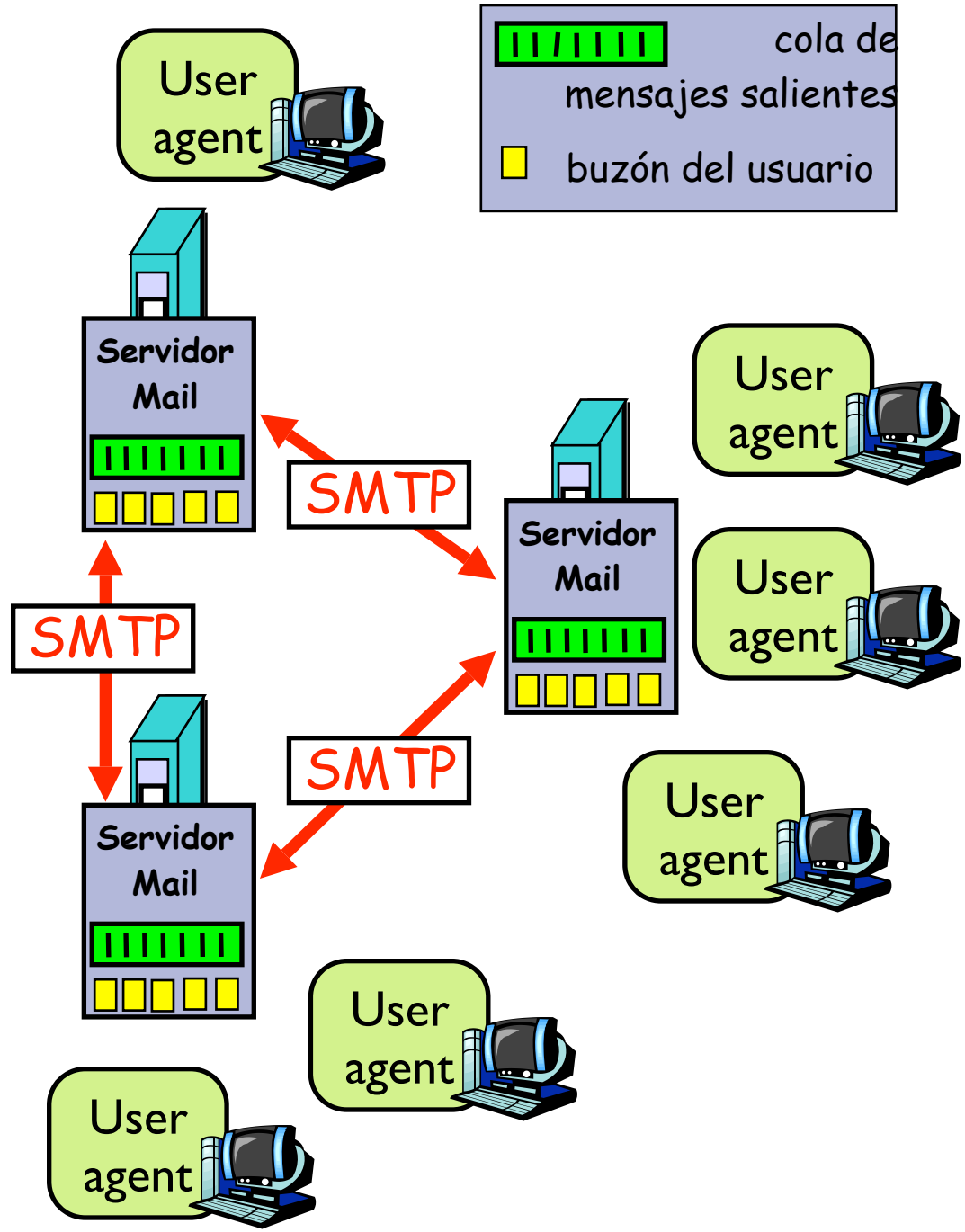
Electronic Mail

Tres elementos principales:

- Agentes de usuario (*user agents*)
- mail servers
- Simple Mail Transfer Protocol: SMTP

User Agent

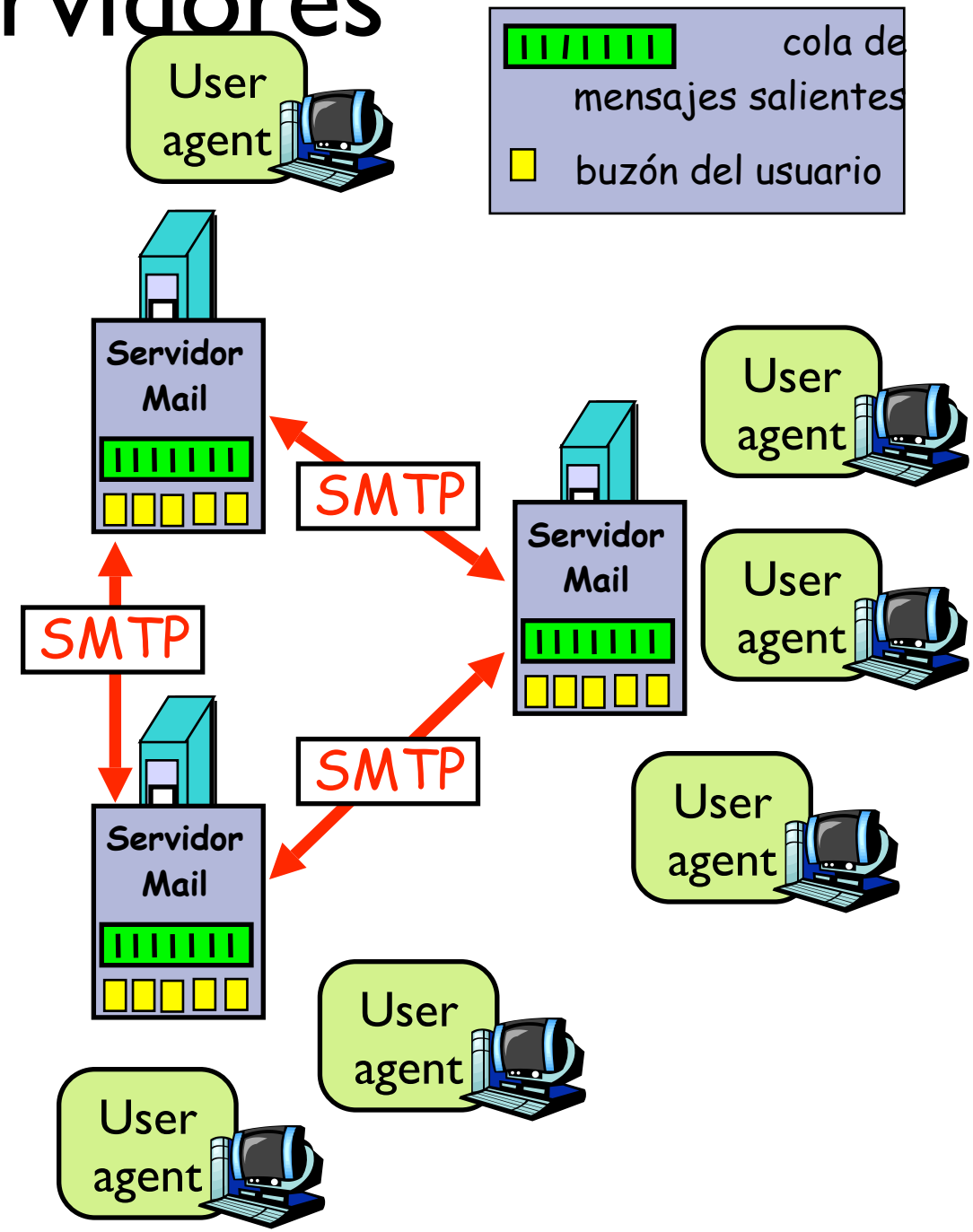
- alias "programa de correo"
- Componer, editar, leer mensajes de correo
- ej., Eudora, Outlook, elm, Netscape Messenger
- Mensajes salientes y entrantes en el servidor



Electronic Mail: Servidores

Servidores de Mail:

- mailbox contiene los mensajes entrantes para el usuario
- cola de mensajes salientes (a enviar)
- Protocolo SMTP entre servidores de correo para enviar mensajes
  - cliente: el servidor de correo que envía
  - "servidor": el servidor de correo que recibe



Electronic Mail: SMTP [RFC 2821]

- Emplea TCP para entregar de forma fiable los mensajes entre el cliente y el servidor
- Puerto 25
- Transferencia directa: del servidor del emisor al servidor del receptor
- Tres fases en al transferencia
  - *handshaking* (el saludo)
  - transferencia de mensajes
  - cierre
- Interacción mediante comandos y respuestas
  - comandos: texto ASCII
  - respuestas: código de estado y frase de estado
- Los mensajes deben estar en ASCII de 7 bits

Ejemplo: Alicia envía mensaje a Bob

1) Alicia emplea un UA para crear el mensaje para **`bob@someschool.edu`**

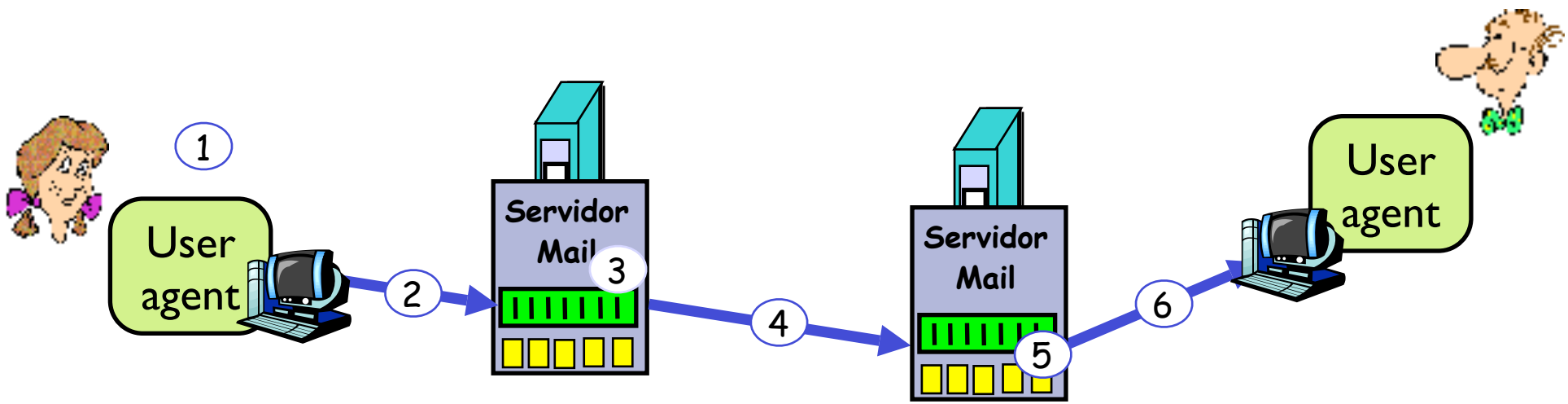
2) El programa envía el mensaje a su servidor de correo y lo coloca en una cola de mensajes

3) El Servidor de Mail, como cliente, abre una conexión TCP con el Servidor de Bob

4) Envía el mensaje de Alicia empleando SMTP sobre esa conexión TCP

5) El servidor de mail de Bob coloca el mensaje en su buzón

6) Bob lanza su UA para leer el mensaje (volveremos a esta parte)



Ejemplo de SMTP

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

[S]ervidor [C]liente

Probando SMTP

- Escriba:

  `$ telnet servername 25`

  - Pruebe los comandos HELO, MAIL FROM, RCPT TO, DATA, QUIT
  - Con esos comandos puede enviar un email sin emplear un programa de email

```
$ telnet correo.unavarra.es 25
Trying 130.206.166.108...
Connected to si.unavarra.es.
Escape character is '^]'.
220 unavarra.es ESMTP Sendmail 8.9.3/8.9.1 (IRIS 3.0); Sun, 7 Aug 2005
14:06:28 +0200 (MET DST)
HELO mikel.tlm.unavarra.es
250 unavarra.es Hello s169m177.unavarra.es [130.206.169.177], pleased to
meet you
MAIL FROM: mikel.izal@unavarra.es
250 mikel.izal@unavarra.es... Sender ok
RCPT TO: mikel.izal@gmail.com
250 mikel.izal@gmail.com... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Hola esto es para probar que va
.
250 OAA13441 Message accepted for delivery
QUIT
221 unavarra.es closing connection
Connection closed by foreign host.
```

Más sobre SMTP

- SMTP emplea conexiones persistentes
- SMTP requiere que el mensaje (cabecera y contenido) esté en ASCII de 7 bits
- El servidor de SMTP reconoce el fin del mensaje al ver una linea que sólo contenga .
  `CRLF.CRLF  "\r\n.\r\n"`

Comparación con HTTP:

- HTTP: pull
- SMTP: push
- Ambos usan comandos y respuestas en ASCII

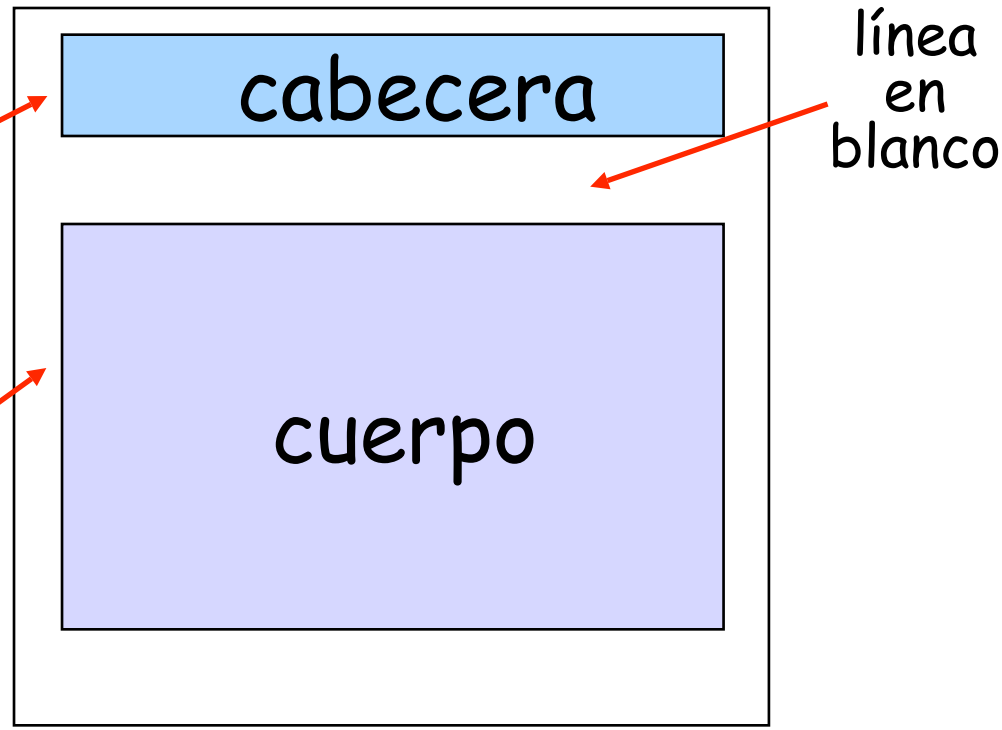
Formato del mensaje de email

SMTP: protocolo para intercambiar mensajes de email

RFC 822: estándar para el formato del mensaje:

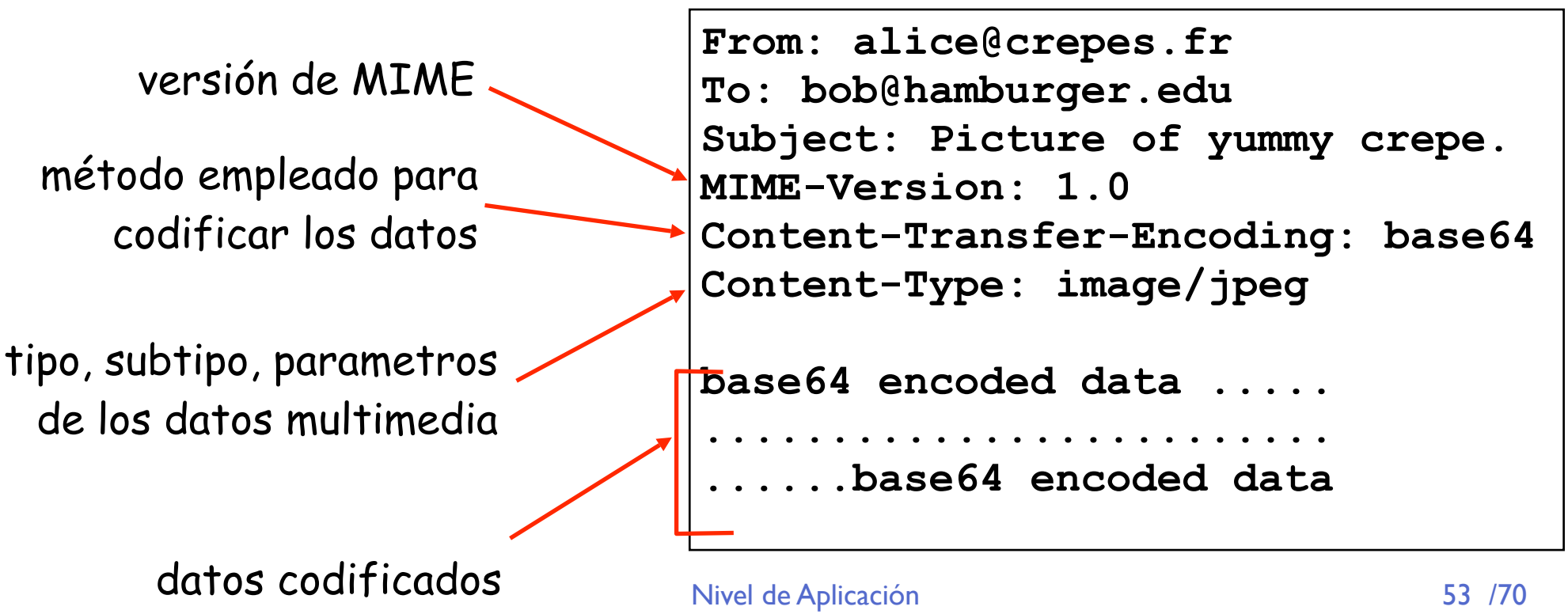
- líneas de cabecera, ej.,
  - `To:`
  - `From:`
  - `Subject:`

  diferentes de los comandos de SMTP
- cuerpo
  - el "mensaje", solo caracteres ASCII

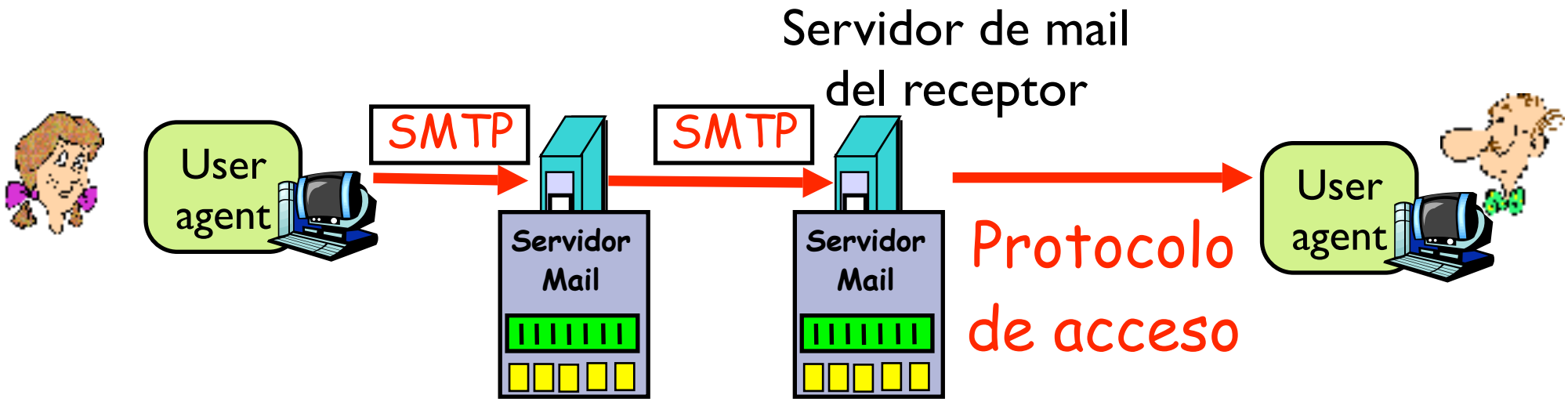


Formato del mensaje: multimedia

- MIME: MultImedia Mail Extension, RFC 2045, 2056
- Permite mandar contenido que no sea texto ASCII
- Líneas adicionales en la cabecera del mensaje para declarar el tipo del contenido



Protocolos de acceso al Mail



- SMTP: entrega/almacena en el servidor del receptor
- Protocolo de acceso al Mail: obtención de mensajes del servidor
  - POP: Post Office Protocol [RFC 1939]
    - Autorización (agente <-->servidor) y descarga
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - más funcionalidades (más complejo)
    - manipulación de mensajes almacenados en el servidor
  - HTTP: Hotmail , Yahoo! Mail, etc.

Protocolo POP3

- **Sobre TCP puerto 110**

Autorización

- Comandos del cliente:
  - `user:` declara el nombre de usuario
  - `pass:` clave
- Respuestas del servidor:
  - `+OK`
  - `-ERR`

Fase de transacción, cliente:

- `list:` lista números de mensajes
- `retr:` descarga mensaje por número
- `dele:` borrar
- `quit`

```
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: <contenido mensaje 1>
S: .
C: dele 1
C: retr 2
S: <contenido mensaje 2>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Más sobre POP3 e IMAP

Más sobre POP3

- El ejemplo anterior era "descargar y borrar"
- Bob no puede volver a leer los mensajes si cambia de cliente
- "Descargar y mantener": copia el mensaje pero no lo borra. Permite descargarlos en otro cliente
- POP3 es sin estado entre sesiones

IMAP

- Mantiene todos los mensajes en un lugar: el servidor
- Permite al usuario organizar los mensajes en carpetas
- IMAP mantiene el estado entre sesiones:
- Nombres de carpetas y relación entre ID de mensaje y carpeta en la que está

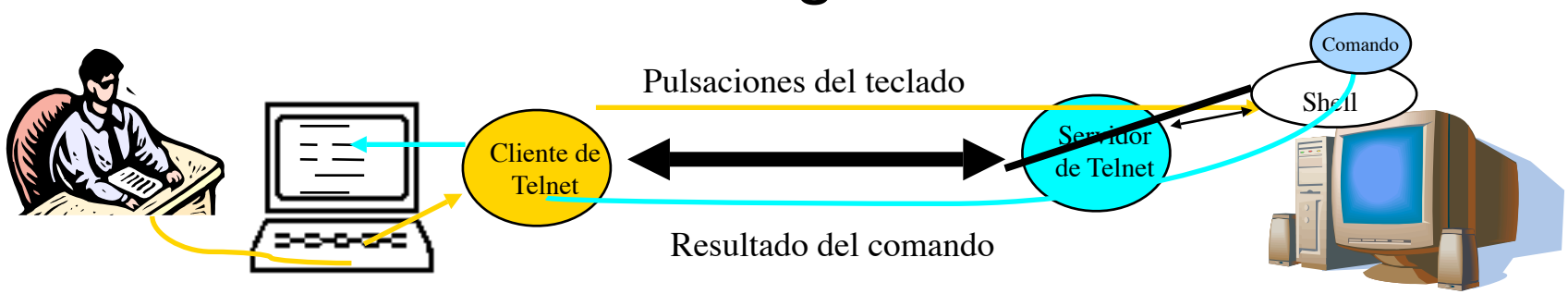
Servicios de Internet

Objetivos:

- Aprender con el ejemplo: Funcionamiento de protocolos de nivel de aplicación
  - **Web y HTTP** ✓
  - **DNS** ✓
  - **SMTP/POP3** ✓
  - **Telnet**
  - **FTP**
  - **P2P**

Login remoto (Telnet)

- Permite el uso interactivo de otra computadora (UNIX) de forma remota como desde un terminal
- Funcionamiento:
  - El usuario ejecuta un cliente de Telnet especificando una máquina servidor...
  - Se crea una conexión TCP con el servidor (puerto del servidor de Telnet=23)...
  - El servidor crea un proceso Shell que queda conectado a la conexión TCP...
  - Las pulsaciones del teclado del usuario se transmiten por la conexión a la Shell...
  - La shell ejecuta los comandos que escribe el usuario...
  - El resultado que el comando mandaría a la pantalla vuelve por la conexión TCP y sale en la pantalla del cliente...
- Otros servicios similares: rlogin, rsh, ssh



Login remoto (Telnet): Ejemplo

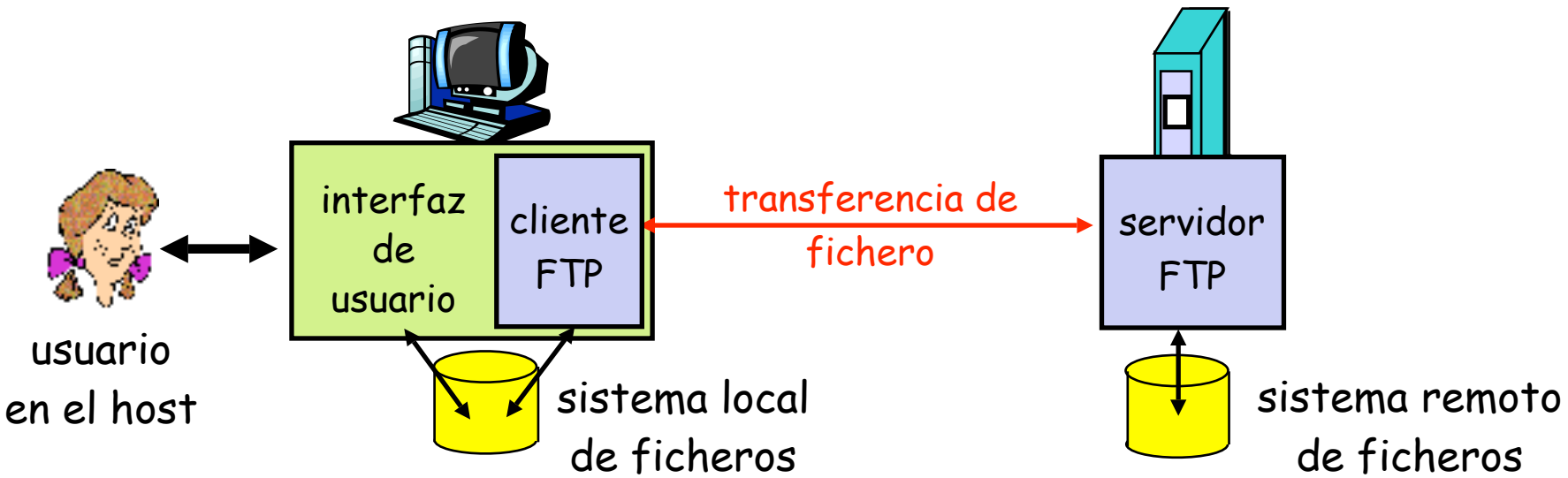
```
$ telnet tlm22.net.tlm.unavarra.es
Trying 10.1.1.22...
Connected to tlm22.net.tlm.unavarra.es.
Escape character is '^]'.
Fedora Core release 2 (Tettnang)
Kernel 2.6.5-1.358 on an i686
login: mikel
Password:
Last login: Fri May  6 20:33:54 from bender.net.tlm.unavarra.es
[mikel@tlm22 mikel]$ ls -l
total 106132
drwxr-xr-x   2 mikel staff     4096 Mar  5 00:17 a
drwxr-xr-x  15 mikel staff     4096 Oct 25  2004 apache
-rw-r--r--   1 mikel staff 41685513 Nov 12  2004 borrar
drwxr-xr-x   4 mikel staff     4096 Dec  2  2004 demo-italia
-rw-------   1 mikel staff   116627 Dec  2  2004 demo-italia.tar.gz
drwxr-xr-x   3 mikel staff     4096 Jun 27 09:43 Desktop
drwx------   4 mikel staff     4096 May 27 15:38 evolution
drwxr-xr-x   2 mikel staff     4096 Oct  6  2004 prueba_c
-rw-r--r--   1 mikel staff   209211 May  6 10:24 prueba.ps
drwxr-xr-x   2 mikel staff     4096 May 11 21:34 py23
[mikel@tlm22 mikel]$
```

Servicios de Internet

Objetivos:

- Aprender con el ejemplo: Funcionamiento de protocolos de nivel de aplicación
  - **Web y HTTP** ✓
  - **DNS** ✓
  - **SMTP/POP3** ✓
  - **Telnet** ✓
  - **FTP**
  - **P2P**

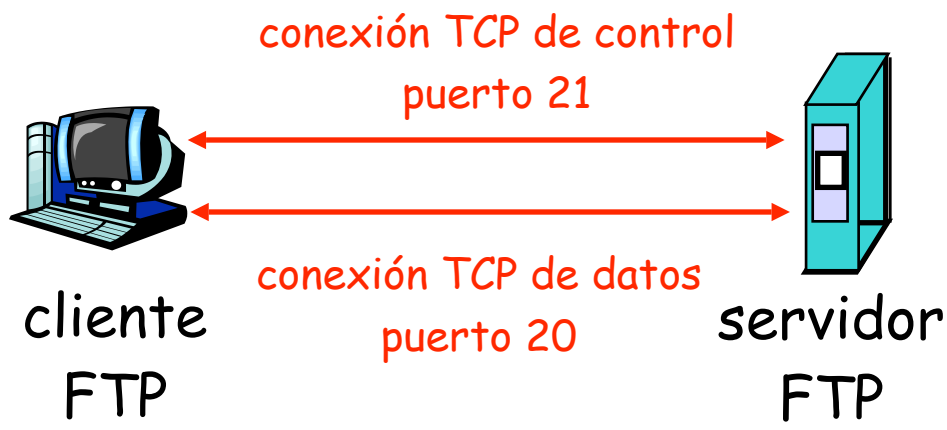
FTP: File Transfer Protocol



- Transferencia de fichero hacia/desde host remoto
- modelo cliente-servidor
  - cliente: extremo que inicia la transferencia (bien sea desde o hacia el extremo remoto)
  - servidor: host remoto
- FTP: RFC 959
- Servidor FTP: TCP puerto 21

FTP: conexiones de datos y control

- El cliente FTP contacta con el servidor en el puerto 21 empleando TCP
- El cliente se autentifica a través de esta conexión de control
- El cliente puede explorar los directorios remotos enviando comandos por la conexión de control
- Cuando el servidor recibe un comando para una transferencia de fichero abre una conexión TCP con el cliente
- Tras transferir el fichero cierra esa conexión de datos



- El servidor abre una segunda conexión TCP para transferir el fichero
- Conexión de control "out of band"
- El servidor FTP mantiene el "estado": directorio actual, autentificación

Comandos y respuestas FTP

Comandos de ejemplo:

Enviados como texto ASCII por el canal de control

- **`USER username`**
- **`PASS password`**
- **`LIST`** devuelve una lista de los ficheros en el directorio actual
- **`RETR filename`** Obtiene el fichero
- **`STOR filename`** Almacena el fichero en el host remoto

Códigos de respuesta:

Código de estado y frase (como en HTTP)

- **`331 Username OK, password required`**
- **`125 data connection already open; transfer starting`**
- **`425 Can't open data connection`**
- **`452 Error writing file`**

Ejemplo de FTP

```
$ ftp tlm22.net.tlm.unavarra.es
Connected to tlm22.net.tlm.unavarra.es (10.1.1.22).
220 (vsFTPd 1.2.1)
Name (tlm22.net.tlm.unavarra.es:mikel): mikel
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (10,1,1,22,215,255)
150 Here comes the directory listing.
drwxr-xr-x    3 1003     1000         4096 Jun 27 07:43 Desktop
-rw-r--r--    1 1003     1000       209211 May 06 08:24 prueba.ps
drwxr-xr-x    2 1003     1000         4096 Oct 06  2004 prueba_c
drwxr-xr-x    2 1003     1000         4096 May 11 19:34 py23
-rw-r--r--    1 1003     1000     20083157 May 11 19:34 py23.tgz
226 Directory send OK.
ftp> get py23.tgz
local: py23.tgz remote: py23.tgz
227 Entering Passive Mode (10,1,1,22,95,165)
150 Opening BINARY mode data connection for py23.tgz (20083157
bytes).
226 File send OK.
20083157 bytes received in 1.86 secs (1.1e+04 Kbytes/sec)
ftp>
```

Servicios de Internet

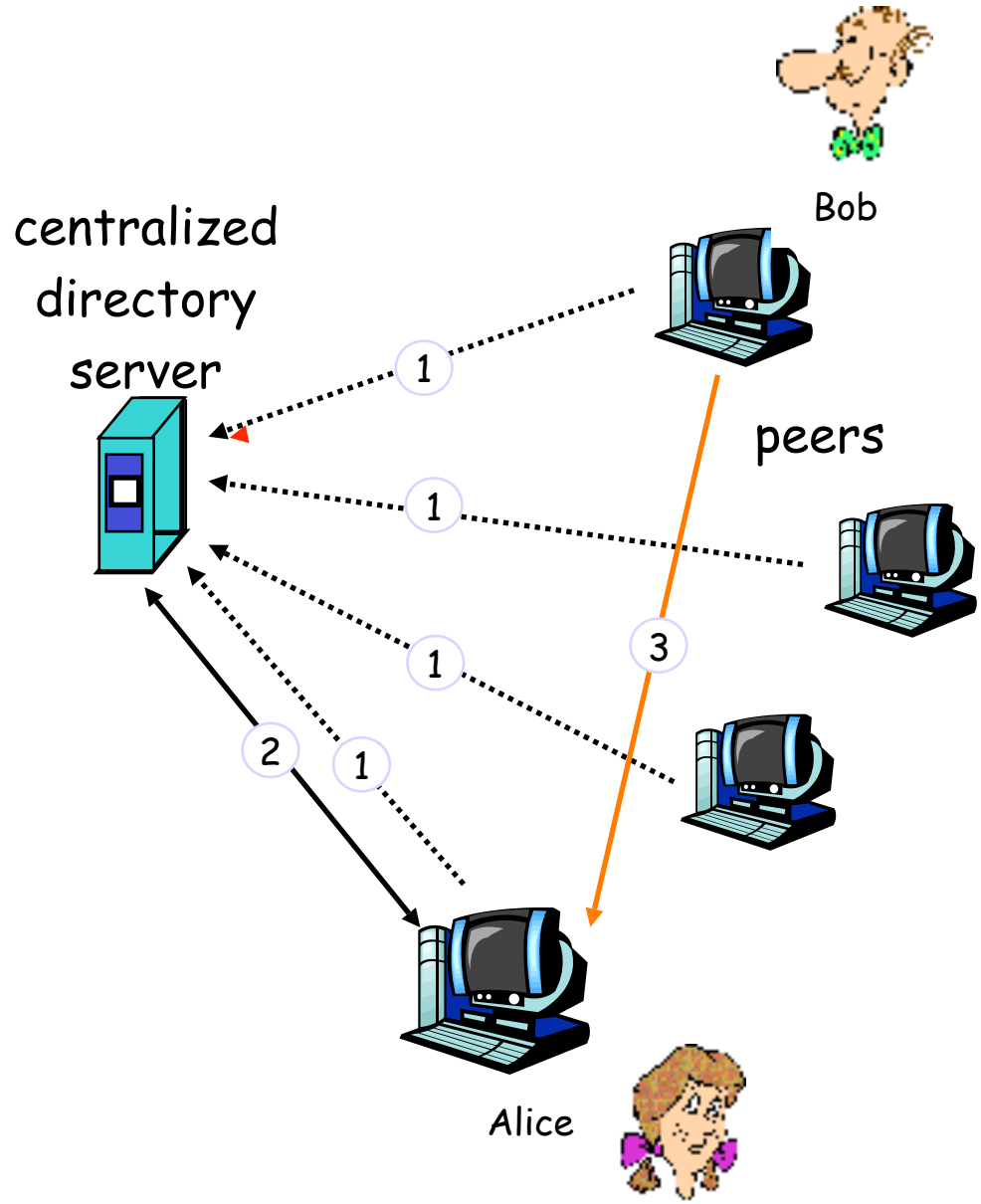
Objetivos:

- Aprender con el ejemplo: Funcionamiento de protocolos de nivel de aplicación
  - **Web y HTTP** ✓
  - **DNS** ✓
  - **SMTP/POP3** ✓
  - **Telnet** ✓
  - **FTP** ✓
  - **P2P**

P2P: directorio centralizado

Diseño original de "Napster"

1) Cuando un peer se conecta, informa al servidor central:
   - Dirección IP
   - contenido
2) Alice hace una búsqueda de "Hey Jude"
3) Alice pide el fichero a Bob



Ventajas e inconvenientes

Ventajas

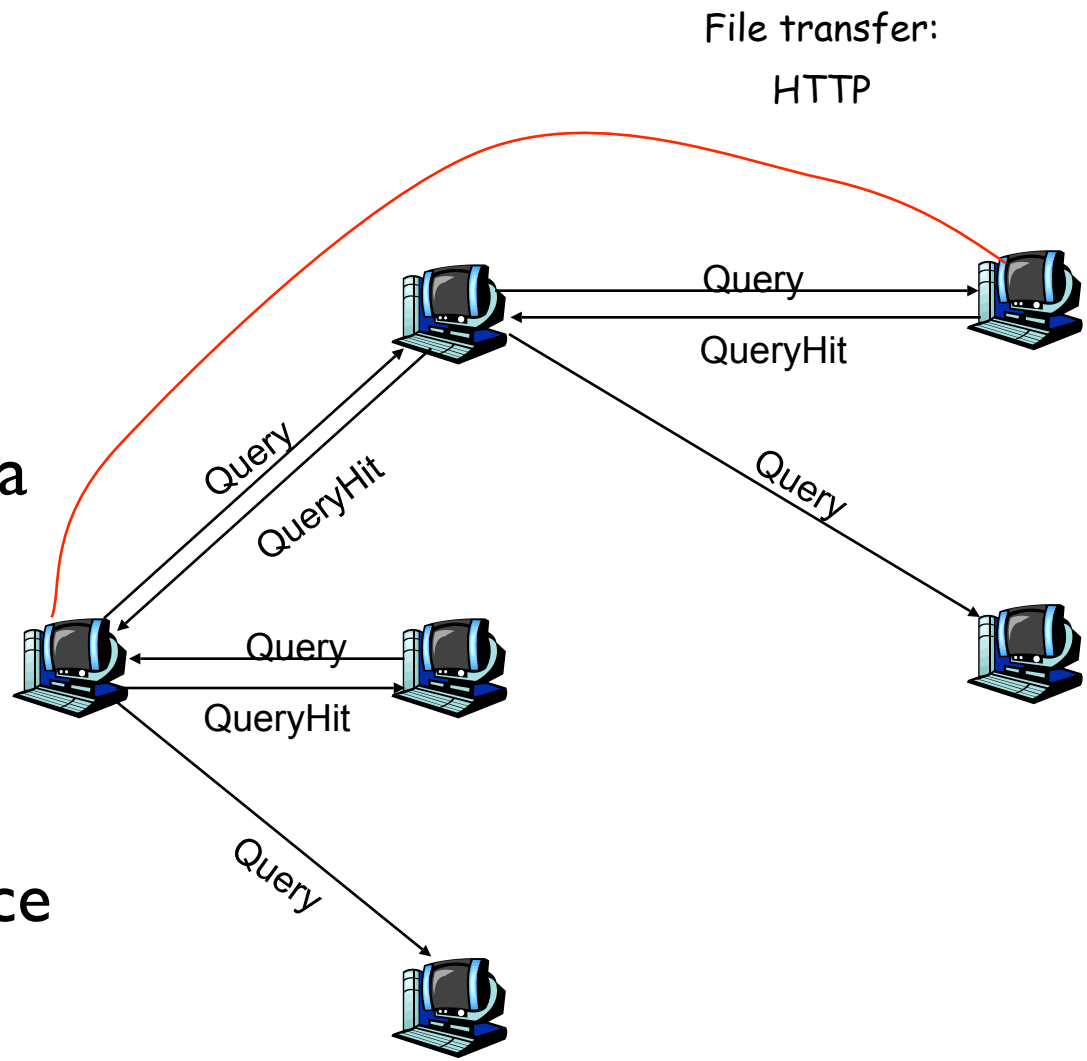
- Todos los peers son servidores
- Altamente escalable

Inconvenientes

- Un punto de fallo central
- Impone un límite de prestaciones
- Infracción de copyrights!

Gnutella

- Completamente distribuido
- Dominio público
- Overlay network
  - Grafo
  - Cada conexión un enlace
- Petición de búsqueda enviada sobre las conexiones TCP
- peers reenvían la petición
- Respuesta enviada por el camino inverso
- Escalabilidad: limitar el alcance de la inundación



Conclusiones

- El nivel de aplicación en Internet está formado por los protocolos de nivel de los diferentes servicios
- Filosofías para organizar las aplicaciones cliente-servidor, P2P
- Los protocolos de aplicación se construyen sobre sockets que permiten acceder a los protocolos de transporte del SO
- Servicios que ofrecen TCP/UDP a las aplicaciones

Conclusiones

- Protocolos de nivel de aplicación
  - **Web y HTTP** ✓
  - **DNS** ✓
  - **SMTP/POP3** ✓
  - **Telnet** ✓
  - **FTP** ✓
  - **P2P** ✓
- Todos se basan en el uso de sockets que proporcionan dos servicios:
  - fiable orientado a conexión
  - no fiable no orientado a conexión
- Próxima clase: API de Sockets