Servicios de Internet

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Arquitectura de Redes, Sistemas y Servicios
3º Ingeniería de Telecomunicación

Temario

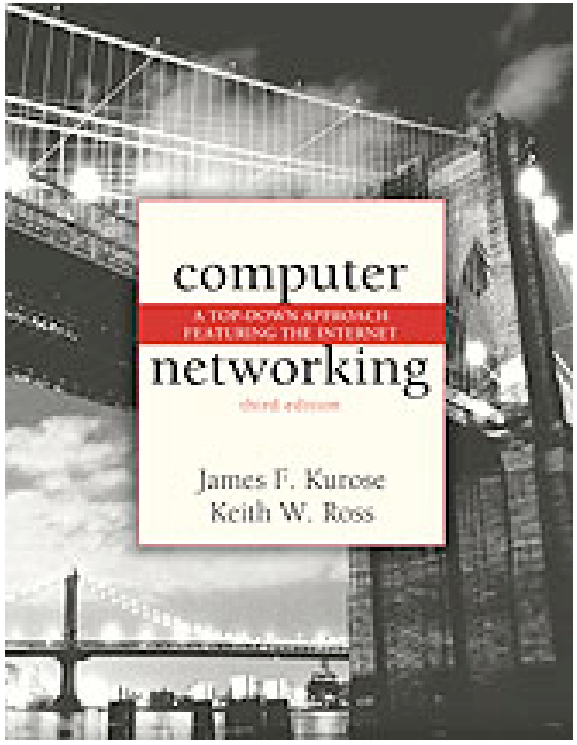
1. Introducción
2. Arquitecturas, protocolos y estándares
3. Conmutación de paquetes
4. Conmutación de circuitos
5. Tecnologías
6. Control de acceso al medio en redes de área local
7. Servicios de Internet

Temario

1. Introducción
2. Arquitecturas, protocolos y estándares
3. Conmutación de paquetes
4. Conmutación de circuitos
5. Tecnologías
6. Control de acceso al medio en redes de área local
7. Servicios de Internet
   - **La Web**
   - **E-Mail**.
   - FTP. Telnet
   - Otros
   - Desarrollo de clientes y servidores

Material

Del Capitulo 2 de
Kurose & Ross,
"**Computer Networking a top-down approach featuring the Internet**"
Addison Wesley



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

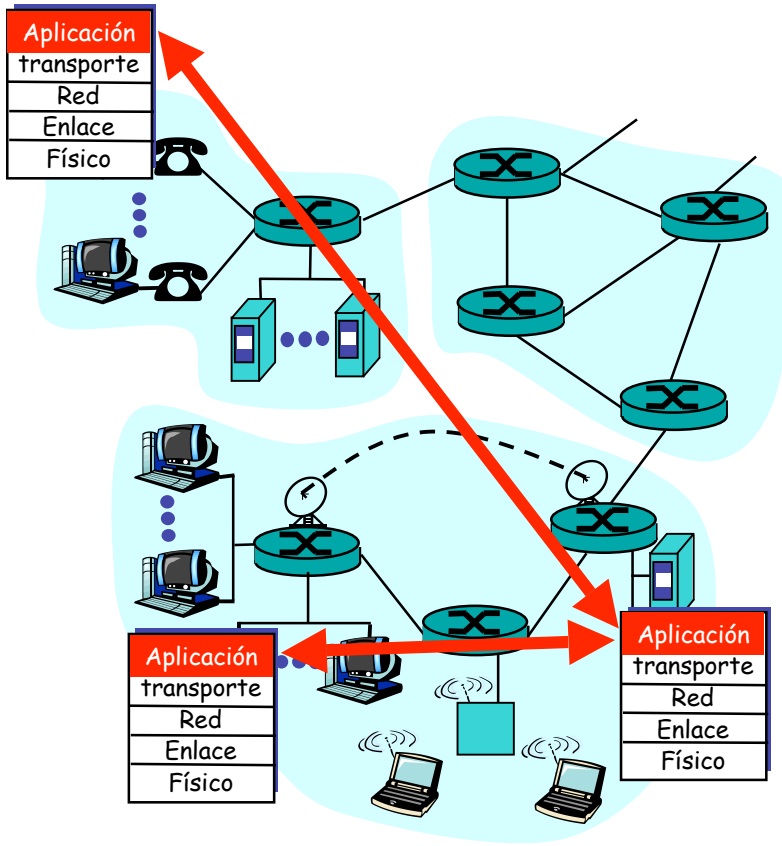
Algunas aplicaciones en red

- 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

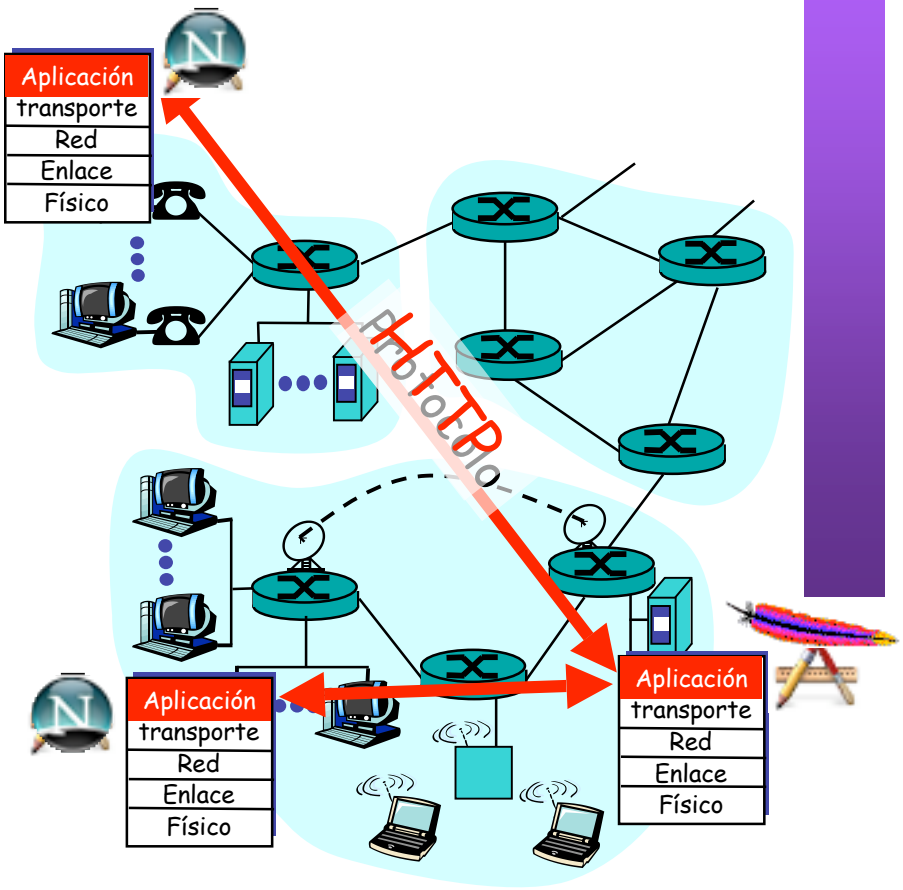
Las aplicaciones

- 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

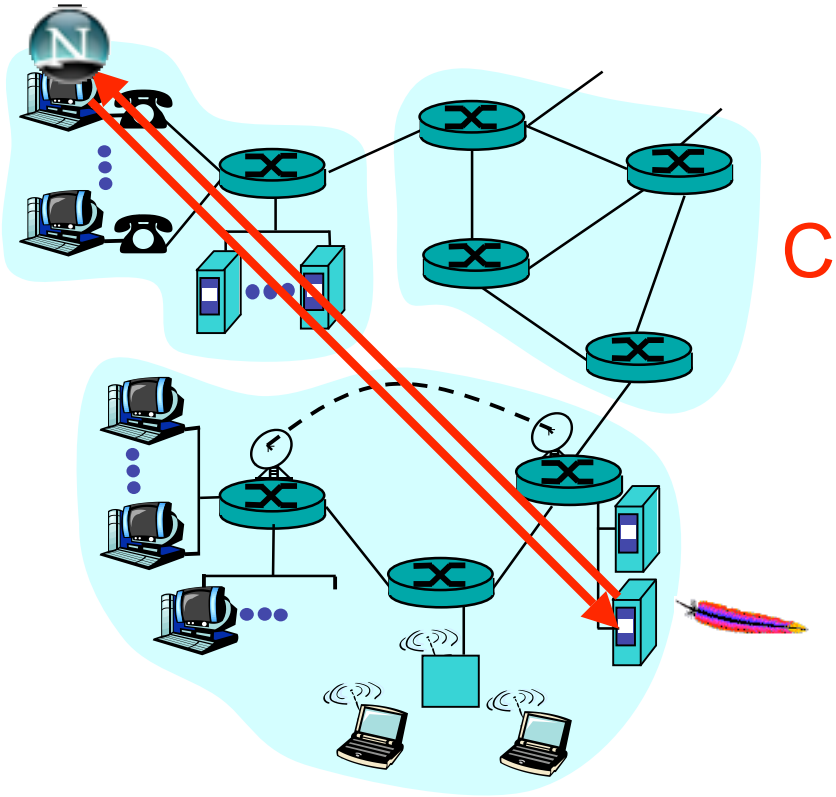
- Los Protocolos de aplicación son una parte de las aplicaciones de red (... …)
- Definen:
  - Tipos de mensajes
  - Sintaxis/formato de mensajes
  - Significado del contenido
  - Reglas de funcionamiento
- Ejemplo: La Web
  - Navegador, Servidor Web (…)
  - HTTP (…)
- Muchos protocolos son estándares abiertos (en RFCs)



Paradigmas

- Cliente-servidor
- Peer-to-peer (P2P)
- Híbrido de cliente-servidor y P2P

Arquitectura cliente-servidor



Servidor:

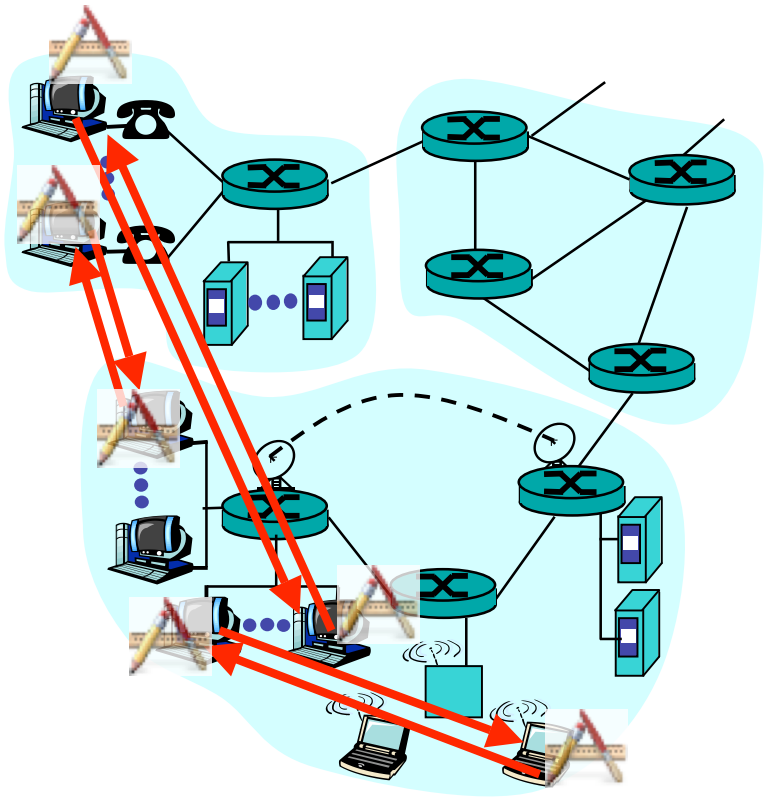
- Comienza a ejecutarse primero (…)
- Espera a ser contactado
- Host siempre disponible
- Dirección permanente

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

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

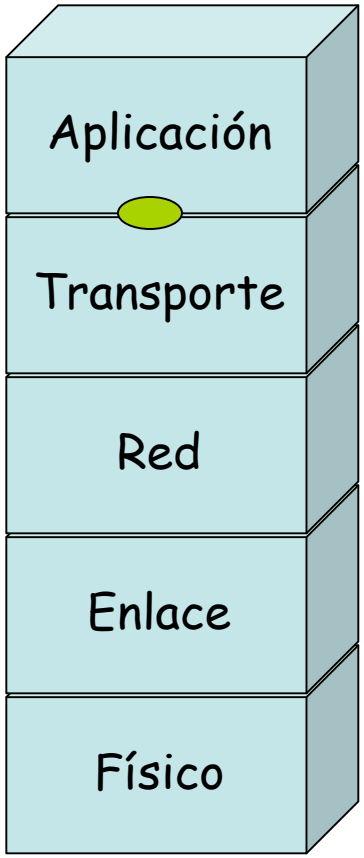
Híbrido de cliente-servidor y P2P

- Napster
  - 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 puede ser P2P
  - Transferencia de ficheros 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

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 apps

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

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

Servicio: Web

Web y HTTP

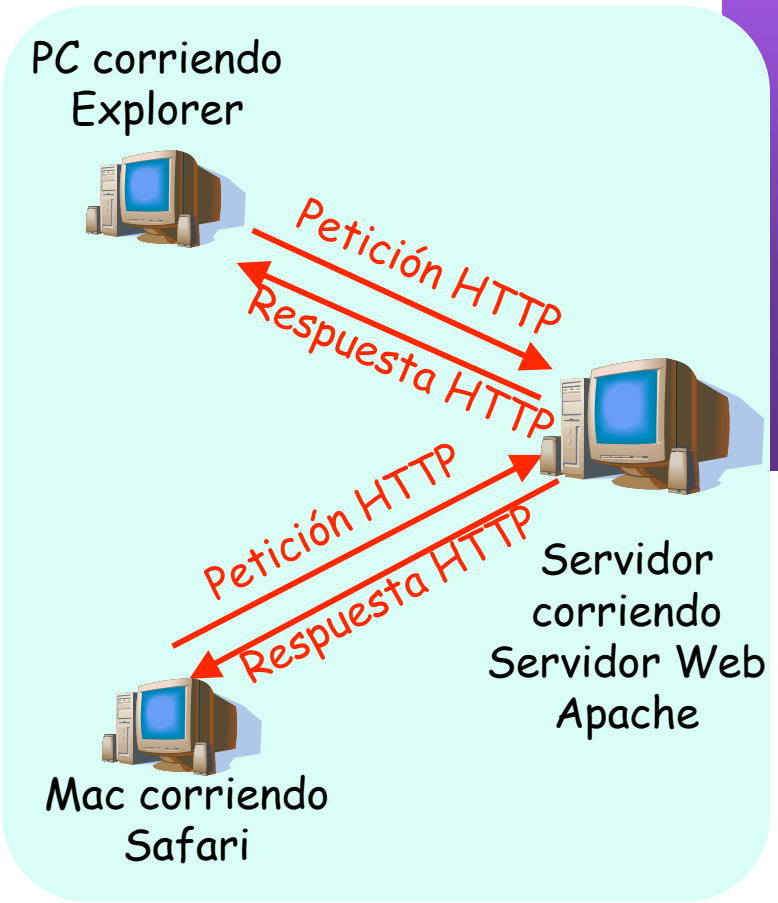
Términos

- Una Página Web está compuesta por objetos
- Un objeto puede ser un fichero HTML, una imagen JPEG, un applet JAVA, un fichero de sonido, etc
- 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 (navegador) 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

- Emplea TCP
- *Well known port*: 80
- Acciones típicas:
  - Cliente conecta con servidor
  - Solicita un objeto mediante su URI
  - Servidor envía el objeto y cierra la conexión
- HTTP es **sin estado**
- El servidor no mantiene ninguna información de peticiones anteriores del cliente
- Los protocolos sin estado son más simples

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

(contiene texto y 1 referencia a una imagen JPEG)

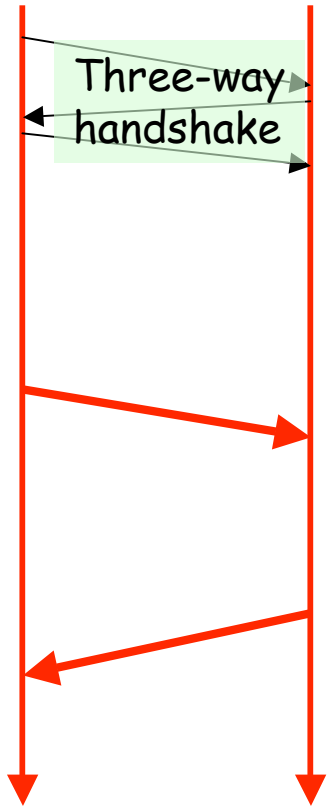
Supongamos que el usuario solicita el URL:

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

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

2: El cliente HTTP envía un mensaje de petición

El mensaje indica que el cliente quiere el objeto /~daniel/index.html



1b: El servidor acepta la conexión, notificando al cliente

3: El servidor HTTP recibe el mensaje de petición

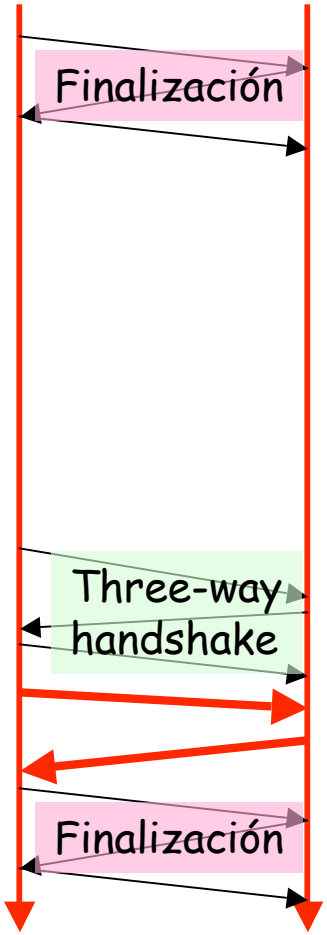
Forma un mensaje de respuesta que contiene el objeto solicitado y lo envía a través de su socket

HTTP no persistente

5: El cliente HTTP recibe el mensaje de respuesta que contiene el fichero HTML

Lo muestra y al interpretarlo encuentra la referencia a un objeto JPEG

6: Los pasos 1-5 se repiten para el objeto JPEG



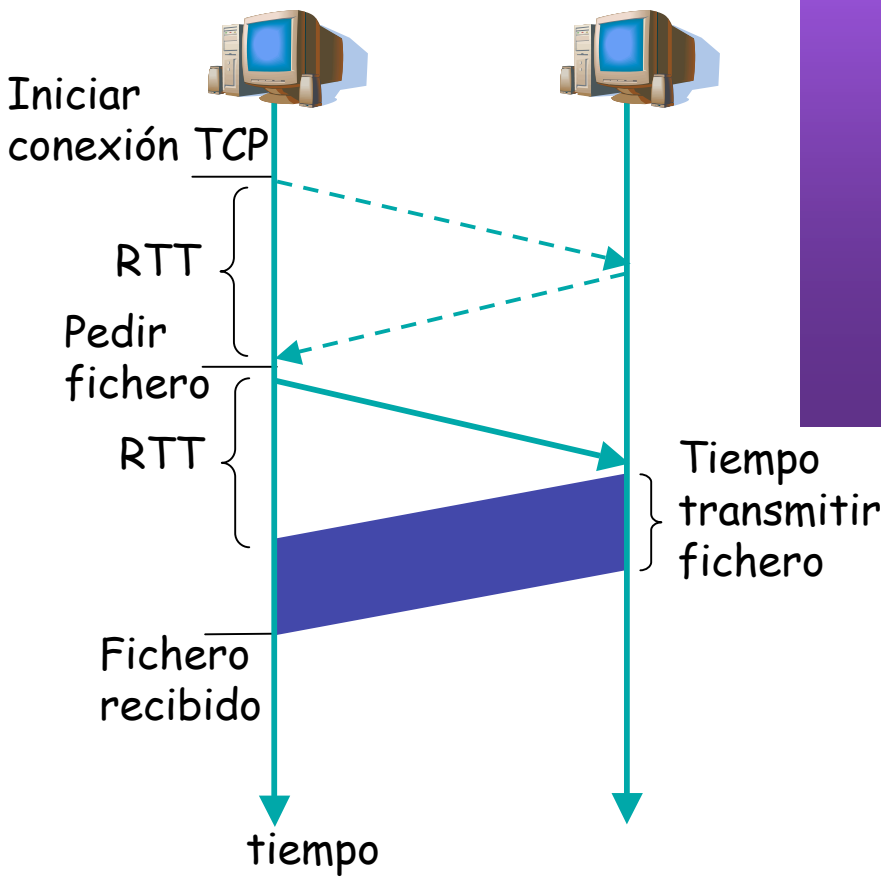
4: El servidor HTTP clierra la conexión TCP

Modelo del tiempo de respuesta

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
- Mejor caso, ignorando mecanismos de TCP

total = 2xRTT + tiempo_transmisión



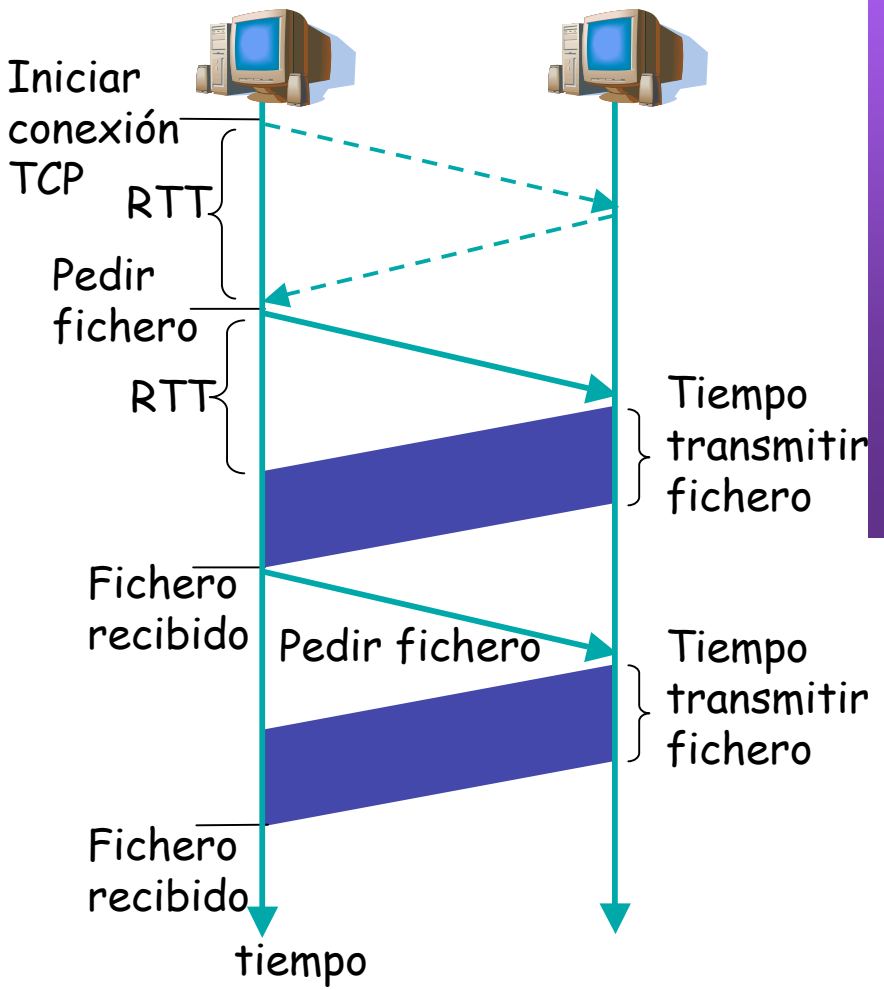
HTTP persistente

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



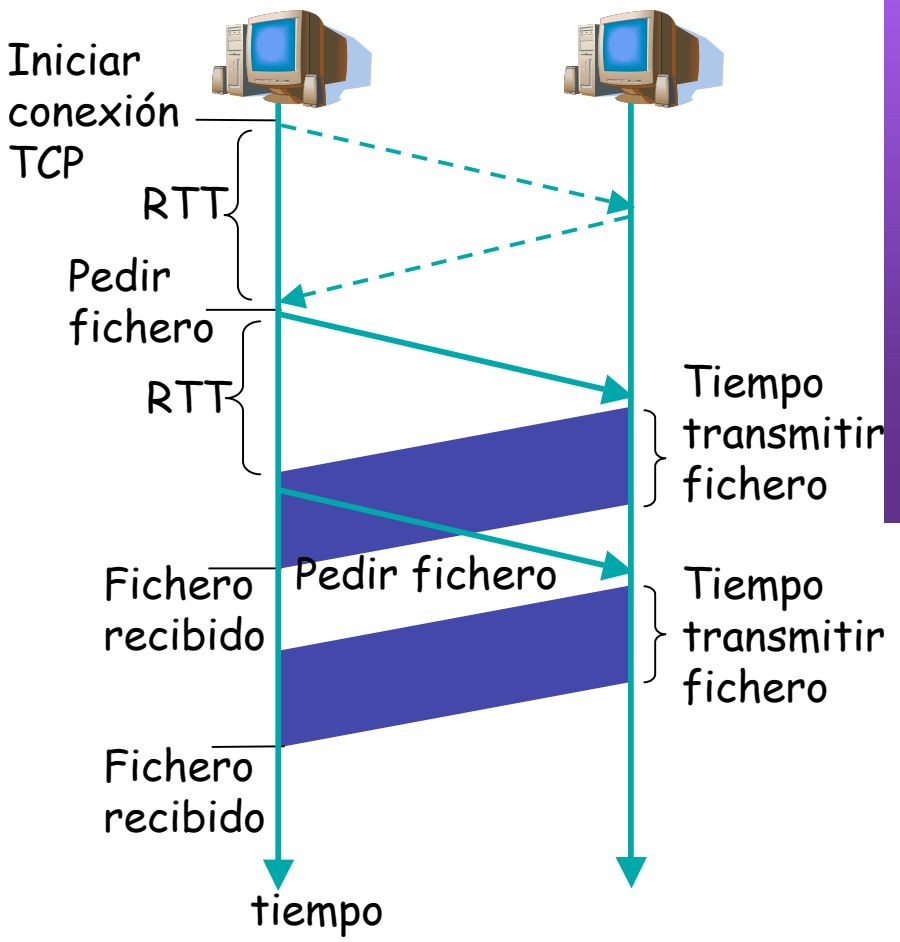
HTTP persistente

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)

línea de petición (comandos GET, POST, HEAD)

líneas de cabecera

Retorno del carro, fín de linea indica fin del mensaje

```
GET /~daniel/index.html HTTP/1.1
Host: www.tlm.unavarra.es
User-agent: Mozilla/4.0
Connection: close
Accept-language:es
```

HTTP response message

línea de estado (código de estado frase de estado)

cabecera

datos, ej., fichero HTML solicitado

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/2.0.47 (Unix)
Last-Modified: Mon, 22 Jun 1998 ...
Content-Length: 6821
Content-Type: text/html

datos datos datos datos datos...
```

Probando HTTP desde el cliente

1. Conexión con su servidor Web favorito:

```
nc www.unavarra.es 80
```

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

2. Escribir una petición GET de HTTP:

```
GET / HTTP/1.1
Host: www.unavarra.es
```

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

3. Vea el mensaje de respuesta del servidor

Ejemplo de HTTP

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en0, link-type EN10MB (Ethernet), capture size 96 bytes
410.891 IP 130.206.169.159.49459 > 66.249.87.104.80: S 2471:2471(0)
410.947 IP 66.249.87.104.80 > 130.206.169.159.49459: S 5231:5231(0) ack 2472
410.947 IP 130.206.169.159.49459 > 66.249.87.104.80: . ack 5232
410.948 IP 130.206.169.159.49459 > 66.249.87.104.80: P 2472:2825(353) ack 5232
411.004 IP 66.249.87.104.80 > 130.206.169.159.49459: . ack 2825
411.005 IP 66.249.87.104.80 > 130.206.169.159.49459: . ack 2825
411.022 IP 66.249.87.104.80 > 130.206.169.159.49459: P 5232:5622(390) ack 2825
411.024 IP 130.206.169.159.49459 > 66.249.87.104.80: F 2825:2825(0) ack 5622
411.080 IP 66.249.87.104.80 > 130.206.169.159.49459: F 5622:5622(0) ack 2826
411.181 IP 130.206.169.159.49460 > 66.249.87.104.80: S 2436:2436(0)
411.237 IP 66.249.87.104.80 > 130.206.169.159.49460: S 2618:2618(0) ack 2437
411.237 IP 130.206.169.159.49460 > 66.249.87.104.80: . ack 2619
411.237 IP 130.206.169.159.49460 > 66.249.87.104.80: P 2437:2812(375) ack 2619
411.293 IP 66.249.87.104.80 > 130.206.169.159.49460: . ack 2812
411.294 IP 66.249.87.104.80 > 130.206.169.159.49460: . ack 2812
411.320 IP 66.249.87.104.80 > 130.206.169.159.49460: P 4049:4482(433) ack 2812
411.320 IP 130.206.169.159.49460 > 66.249.87.104.80: . ack 2619
411.321 IP 66.249.87.104.80 > 130.206.169.159.49460: . 2619:4049(1430) ack 2812
411.321 IP 130.206.169.159.49460 > 66.249.87.104.80: . ack 4482
412.085 IP 66.249.87.104.80 > 130.206.169.159.49459: F 5622:5622(0) ack 2826
412.085 IP 130.206.169.159.49459 > 66.249.87.104.80: . ack 5623
```

Servicio: e-mail

Contenido

- Arquitectura
- SMTP
- Formato de los mensajes
- Protocolos de acceso al mail

Contenido

- **Arquitectura**
- SMTP
- Formato de los mensajes
- Protocolos de acceso al mail

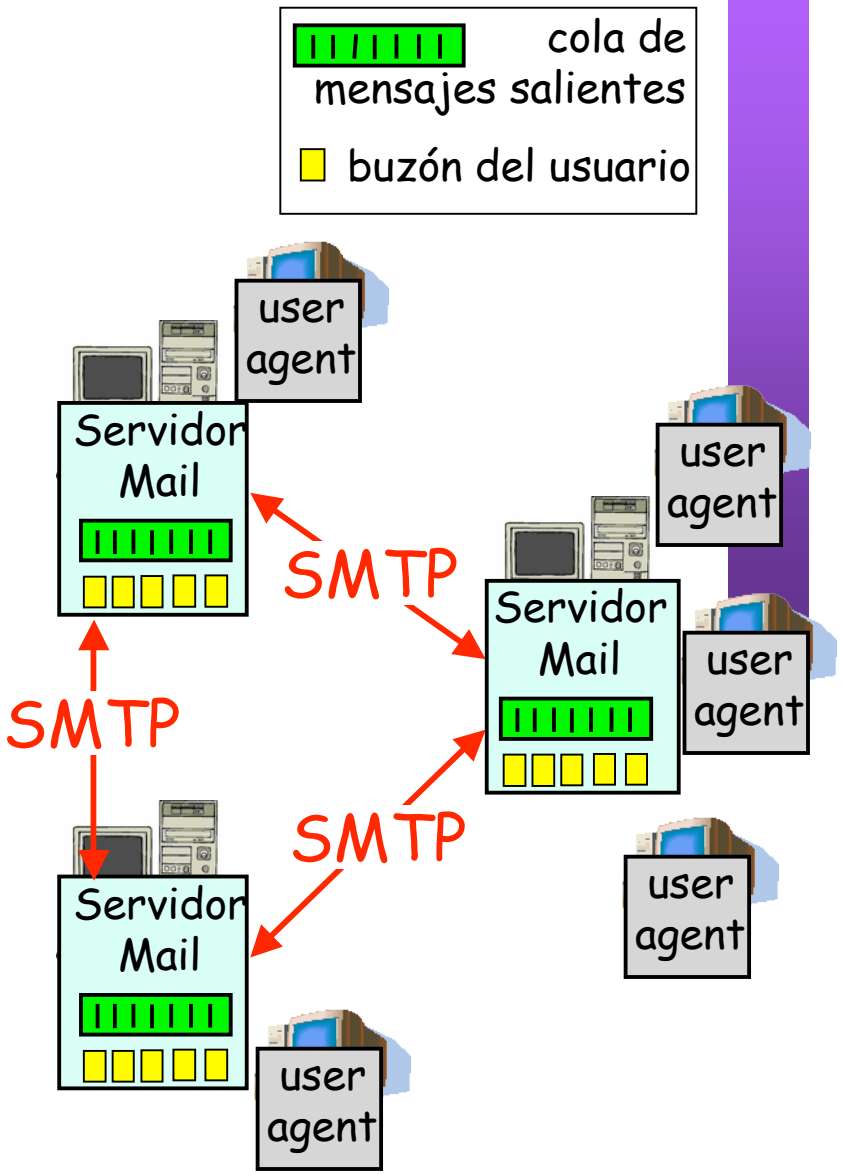
Electronic Mail

Tres elementos principales:

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

User Agent

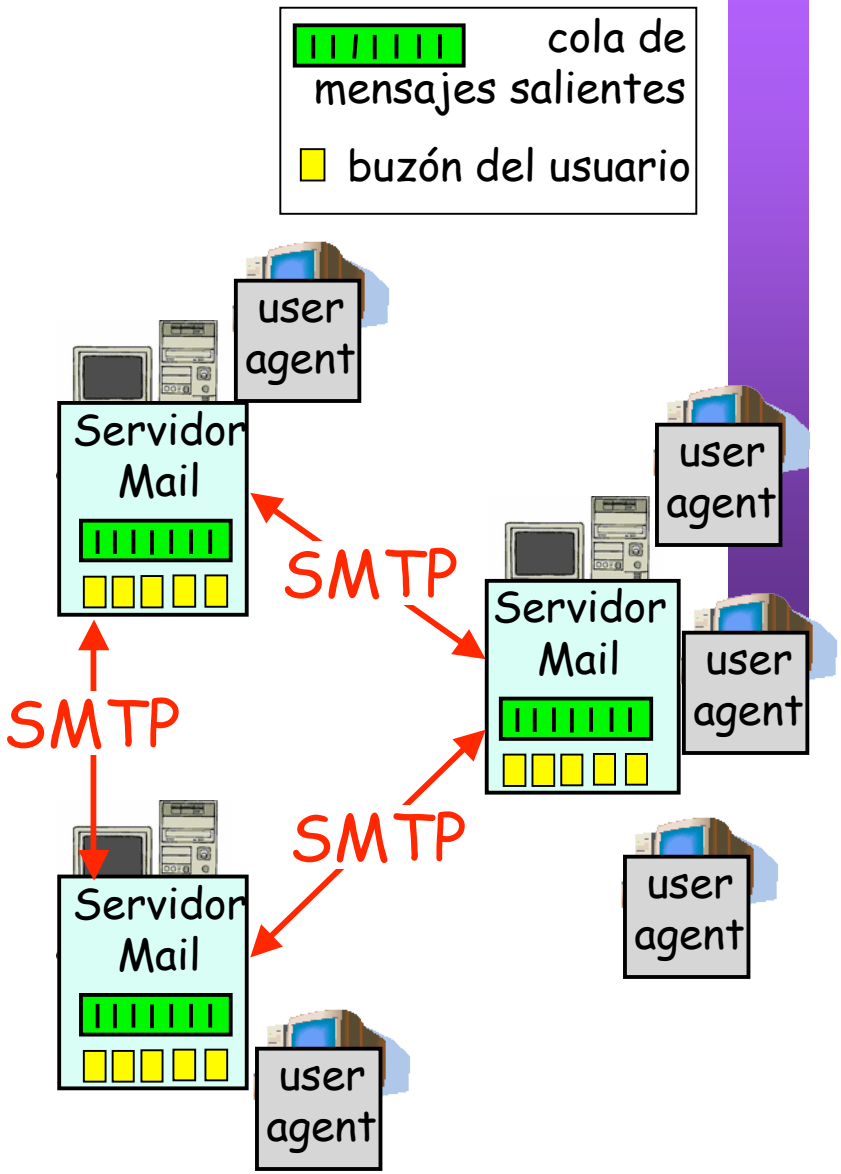
- alias “programa de correo”
- Componer, editar, leer mensajes de correo
- ej., Eudora, Outlook, elm, Netscape Messenger
- Mensajes salientes y entrantes en el servidor



E-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



Contenido

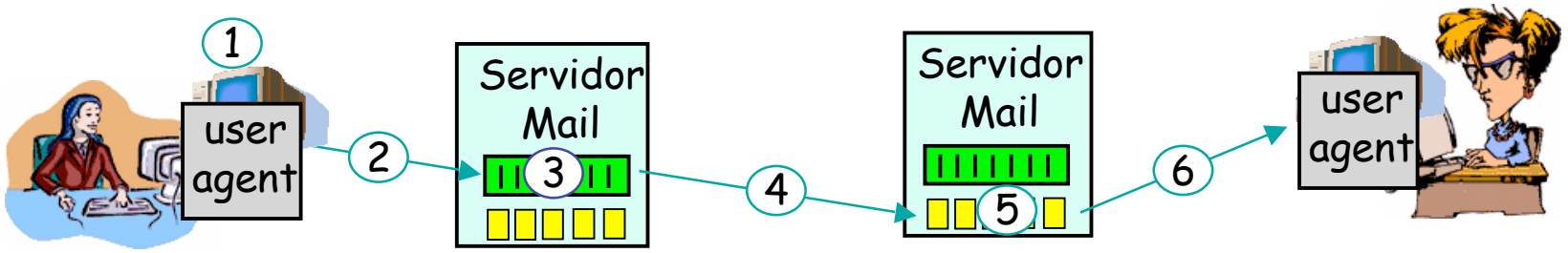
- Arquitectura
- **SMTP**
- Formato de los mensajes
- Protocolos de acceso al mail

E-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 la transferencia
  - handshaking (el saludo)
  - transferencia de mensajes
  - cierre
- Interacción mediante comandos y respuestas
  - comandos: texto ASCII
  - respuestas: código de estado y frase de estado (texto ASCII)
- Los mensajes deben estar en **ASCII** de 7 bits

Ejemplo: Usuario 1 envía mensaje a Usuario 2

1) Usuario 1 emplea un UA para crear el mensaje para usuario2@micasa.com
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 Usuario 2
4) Envía el mensaje de Usuario 1 empleando SMTP sobre esa conexión TCP
5) El servidor de mail de Usuario 2 coloca el mensaje en su buzón
6) Usuario 2 lanza su UA para leer el mensaje (volveremos a esta parte)



Ejemplo de SMTP

[Conexión del cliente con el servidor (puerto 25)]

- 220 unavarra.es ESMTP Sendmail 8.9.3/8.9.1 (IRIS 3.0); Fri, 29 Apr 2005 14:00:19 +0200 (MET DST)
- **HELO daniel.tlm.unavarra.es**
- 250 unavarra.es Hello s169m159.unavarra.es [130.206.169.159], pleased to meet you
- **MAIL FROM: <daniel.morato@unavarra.es>**
- 250 <daniel.morato@unavarra.es>... Sender ok
- **RCPT TO: danielmorato@yahoo.com**
- 250 danielmorato@yahoo.com... Recipient ok
- **DATA**
- 354 Enter mail, end with "." on a line by itself
- **Hola**
- **Aqui, saludandome a mi mismo**
- **.**
- 250 OAA24057 Message accepted for delivery
- **QUIT**
- 221 unavarra.es closing connection

[Cierre de la conexión TCP]

Probando SMTP

- **`nc servername 25`**
- ó
- **`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
- Si el servidor de SMTP es el programa `sendmail` incluso ofrece ayuda con el comando HELP

Algo más sobre SMTP

- Conexiones persistentes
- Requiere que el mensaje (cabecera y contenido) esté en ASCII de 7 bits
- El servidor de SMTP emplea **`CRLF.CRLF`** para reconocer el final del mensaje

Comparación con HTTP:

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

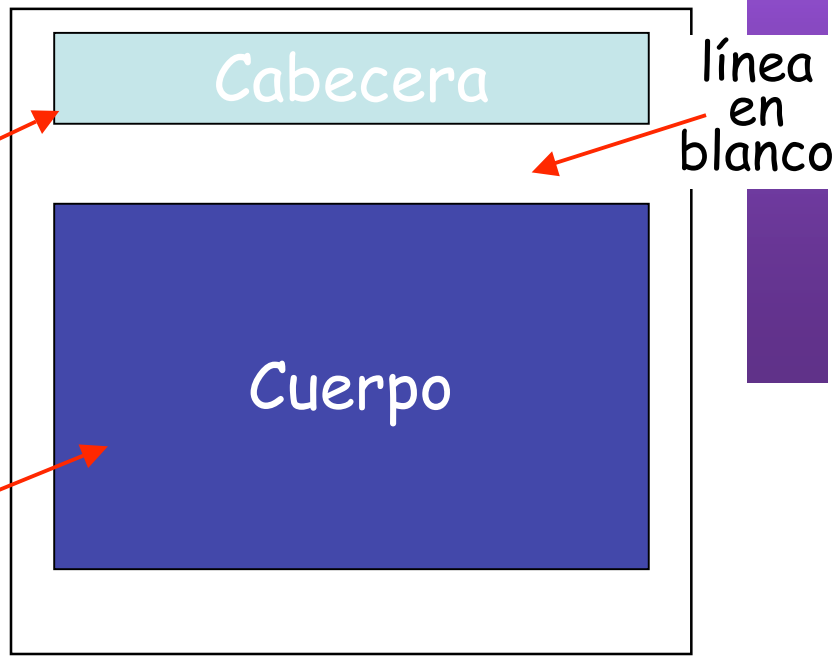
Contenido

- Arquitectura
- SMTP
- **Formato de los mensajes**
- Protocolos de acceso al mail

Formato del mensaje de email

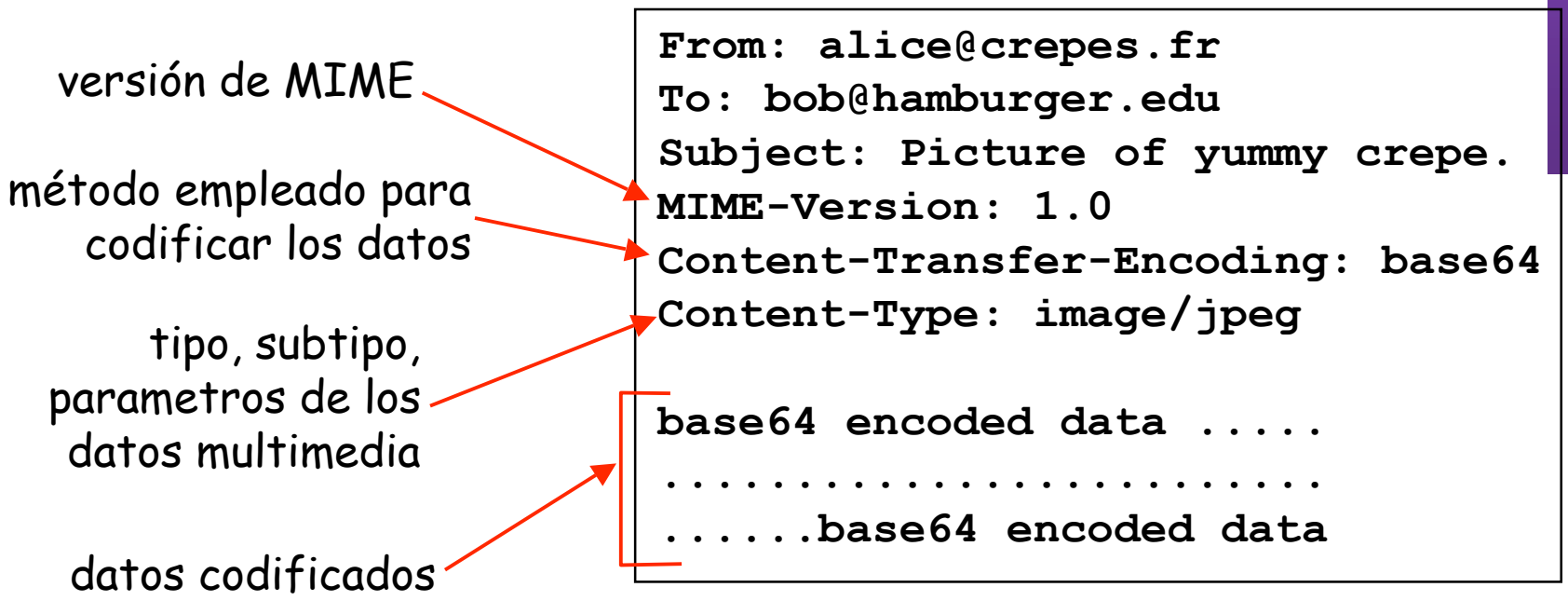
- SMTP: protocolo para intercambiar mensajes de email (RFC 2821)
- RFC 822: estándar para el formato del mensaje:
- Líneas de cabecera, ej.,
  - `To:`
  - `From:`
  - `Subject:`

  Diferentes de los comandos de SMTP
- Cuerpo
  - el “mensaje”, solo caracteres ASCII



Formato del mensaje: Multimedia Extensions

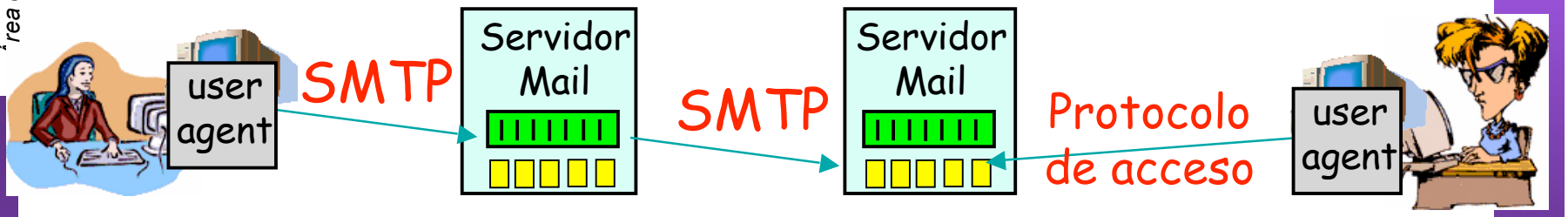
- MIME: MultImedia Mail Extension, RFC 2045, 2056
- Permite mandar contenido que no sea texto ASCII
- Líneas adicionales en la cabecera del mensaje para declarar el tipo del contenido



Contenido

- Arquitectura
- SMTP
- Formato de los mensajes
- **Protocolos de acceso al mail**

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 2821]
    - Autorización (agente ↔ servidor) y descarga
  - IMAP: Internet Message Access Protocol [RFC 3501]
    - Más funcionalidades (más complejo)
    - Manipulación de mensajes almacenados en el servidor
  - HTTP: Hotmail , Yahoo! Mail, etc.

Protocolo POP3

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`**

+OK Qpopper (version 4.0.5) at si starting.
user daniel.morato
+OK
pass hungry
+OK daniel.morato has 412 visible messages (0 hidden) in 35020509 octets.
list
1 498
2 912
.
retr 1
<contenido mensaje 1>
.
dele 1
retr 2
<contenido mensaje 2>
.
dele 2
quit
+OK POP3 at si 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
- Puerto 110

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