



**REDES DE ORDENADORES**  
*Área de Ingeniería Telemática*

# Nivel de Aplicación: WWW

## Tema 5.- Nivel de aplicación en Internet

*Área de Ingeniería Telemática*  
<http://www.tlm.unavarra.es>

*Redes de Ordenadores*  
*Ingeniero Técnico de Telecomunicación Especialidad en Sonido e Imagen, 3º curso*



# Temario

- 1.- Introducción
- 2.- Nivel de enlace en LANs
- 3.- Interconexión de redes IP
- 4.- Nivel de transporte en Internet
- 5.- Nivel de aplicación en Internet



# Temario

- 1.- Introducción
- 2.- Nivel de enlace en LANs
- 3.- Interconexión de redes IP
- 4.- Nivel de transporte en Internet
- 5.- Nivel de aplicación en Internet**
  - Nivel de aplicación: WWW
  - FTP. Telnet. Ejemplos
  - E-mail
  - DNS. P2P. Mensajería...



# Tema 6: 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



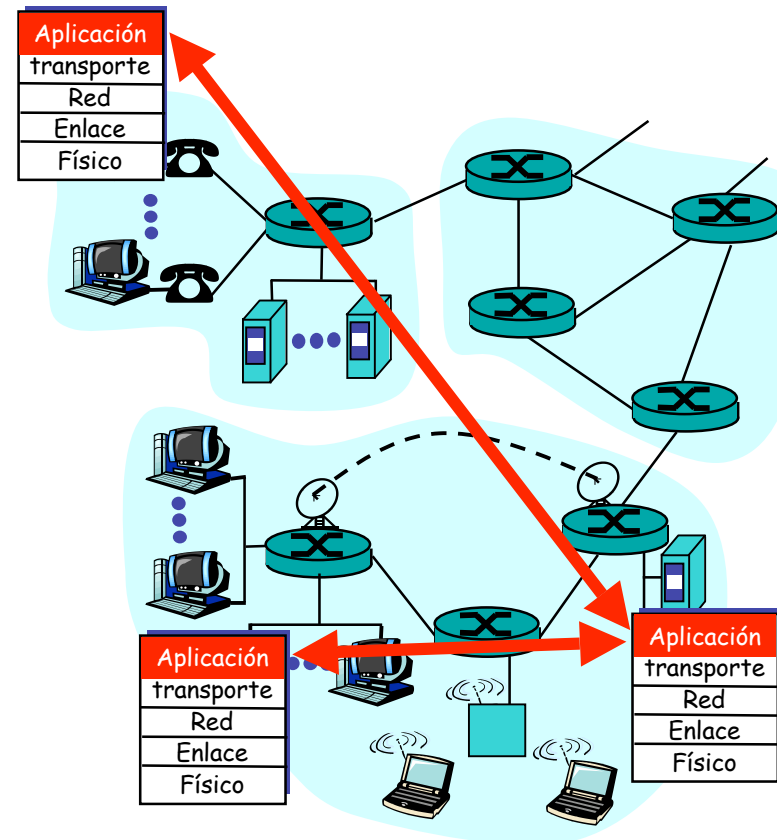
# Algunas aplicaciones en red

- 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

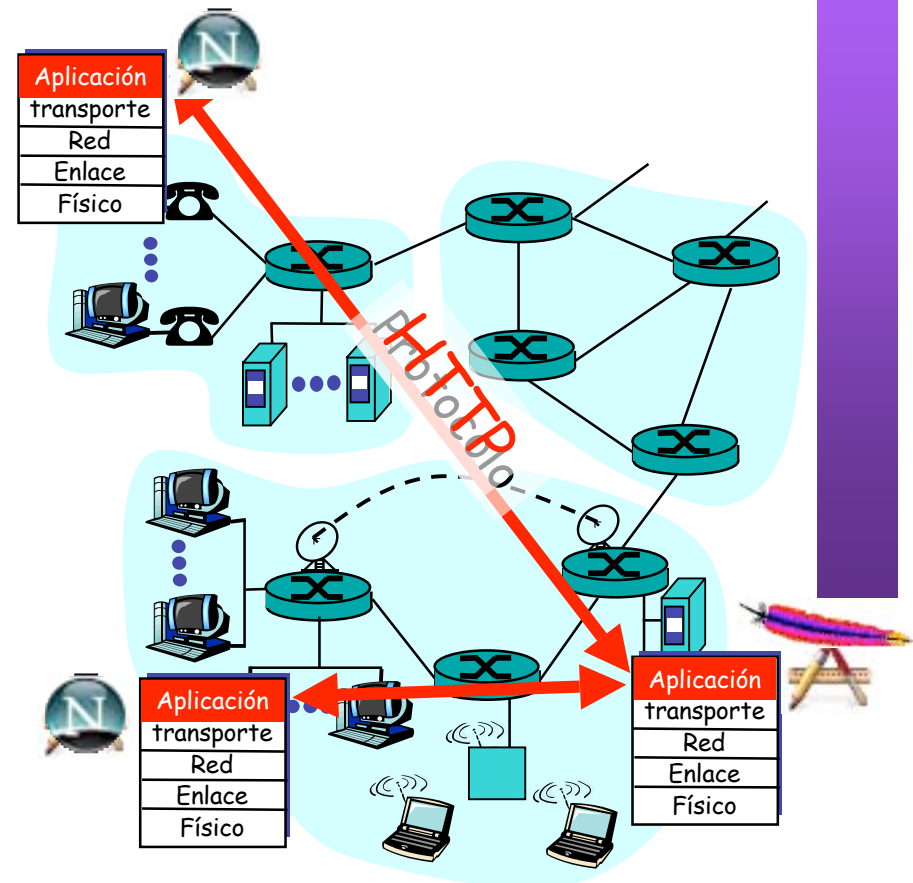
- Son software
- Diferentes máquinas y Sistemas Operativos
- Quienes se comunican son procesos
- IPC: Inter Process Communication
- Nos interesan procesos ejecutándose en diferentes máquinas
- Se comunican a través de una red
- Intercambian mensajes
- Emplean Protocolos de nivel de aplicación (...)





# 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)





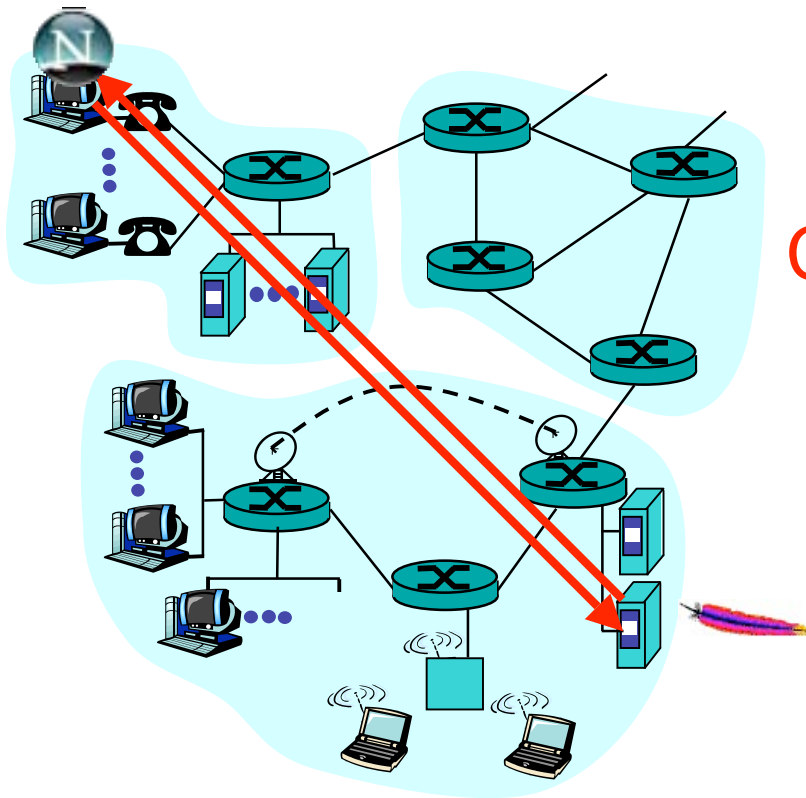
# Paradigmas

- 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:

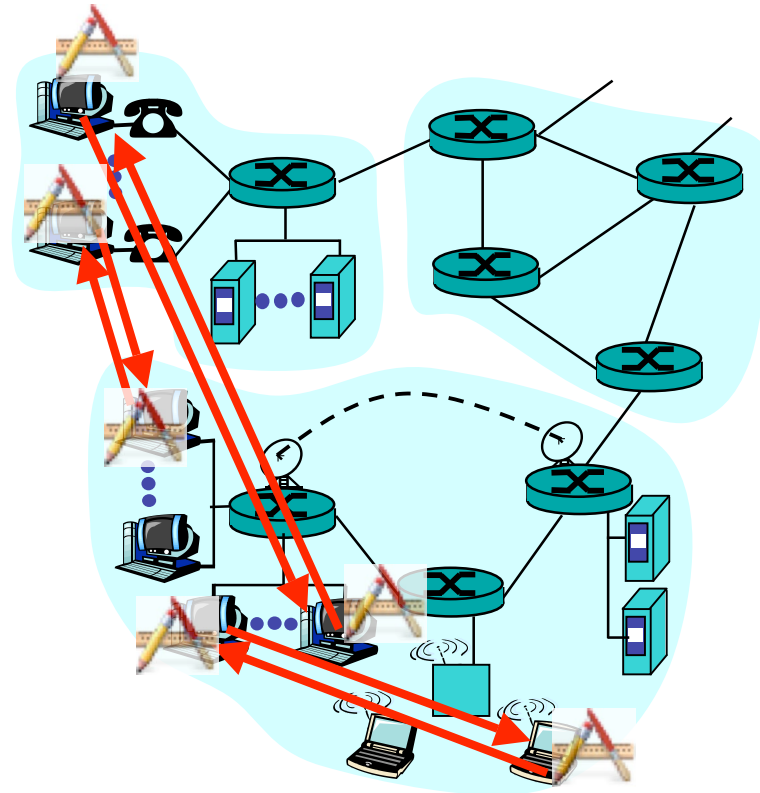
- Lanzado más tarde por el usuario (...)
- **Inicia la comunicación con un servidor (...)**
- No con clientes
- Termina cuando el usuario deja de usarlo
- Puede no tener siempre la misma dirección

# 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





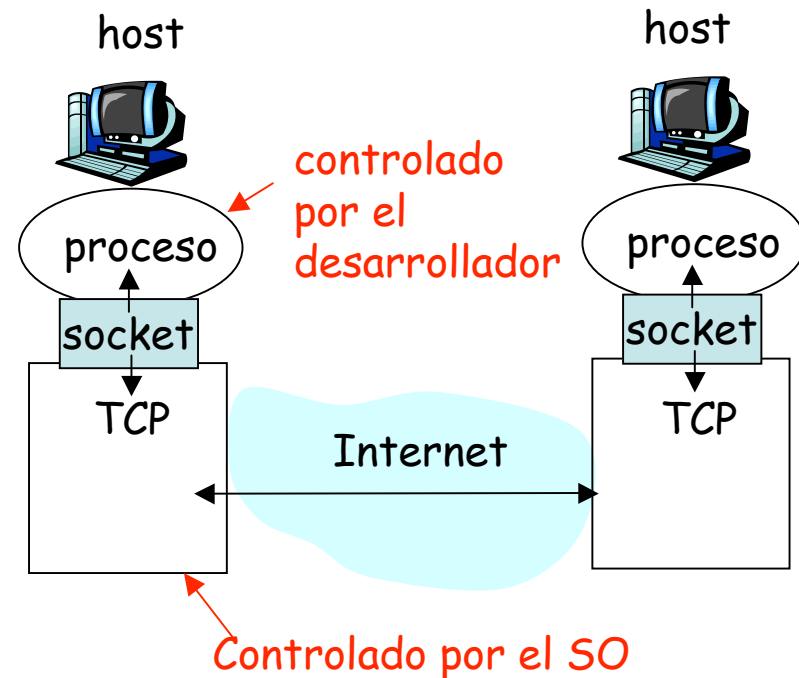
# Híbrido de cliente-servidor y P2P

- 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 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

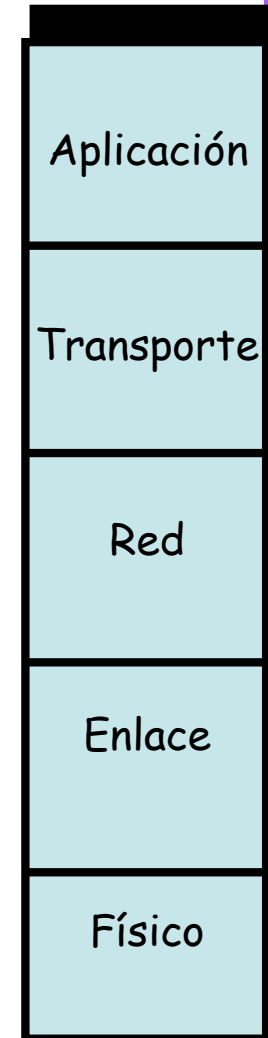
- Los procesos envían y reciben mensajes a través de un socket
- Delega en el nivel de transporte para que haga llegar los mensajes al otro socket
- Acceso a través de un API
- Puede escoger el protocolo de transporte
- Puede configurar algunos parámetros del mismo
- No controla cómo se comporta





# 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 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



# Tema 5: 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**



# 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://www.tlm.unavarra.es/~daniel/index.html`

host path

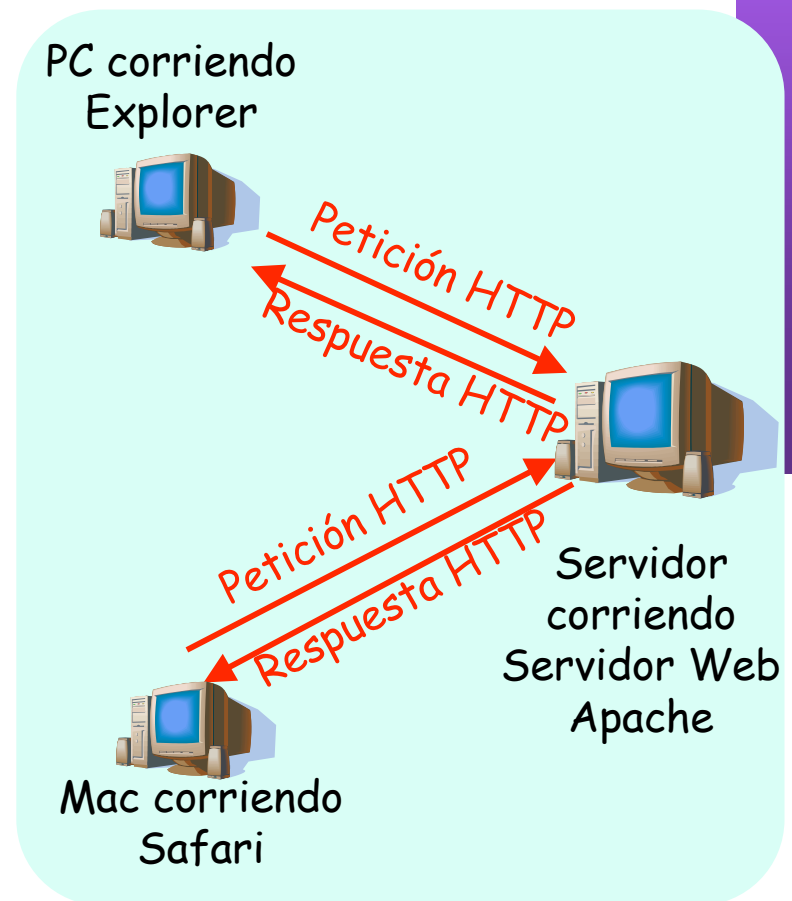




# HTTP

## 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

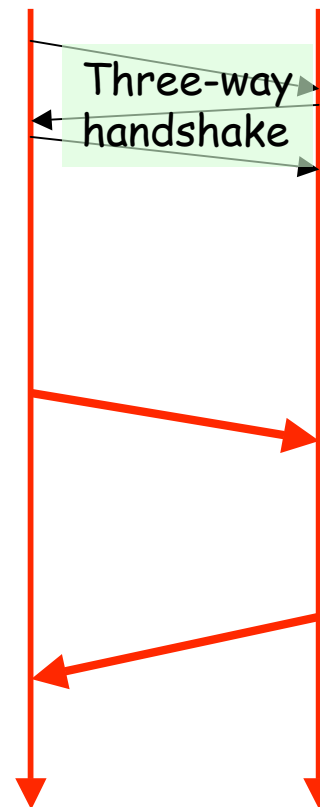
(contiene texto y  
1 referencia a  
una imagen JPEG)

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

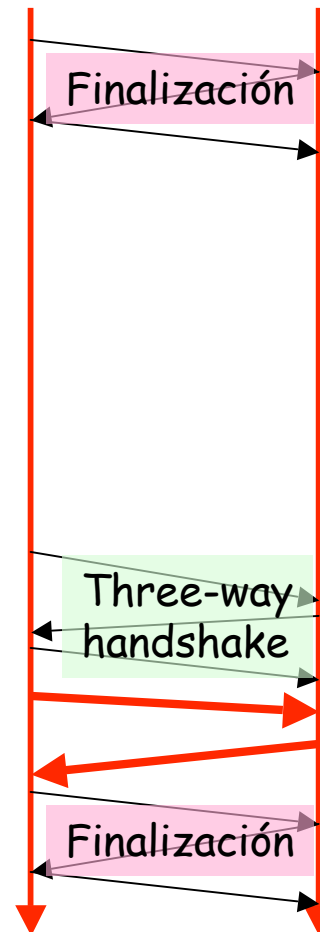


# HTTP no persistente

**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



Nivel de aplicación: WWW

**4:** El servidor HTTP cierra la conexión TCP

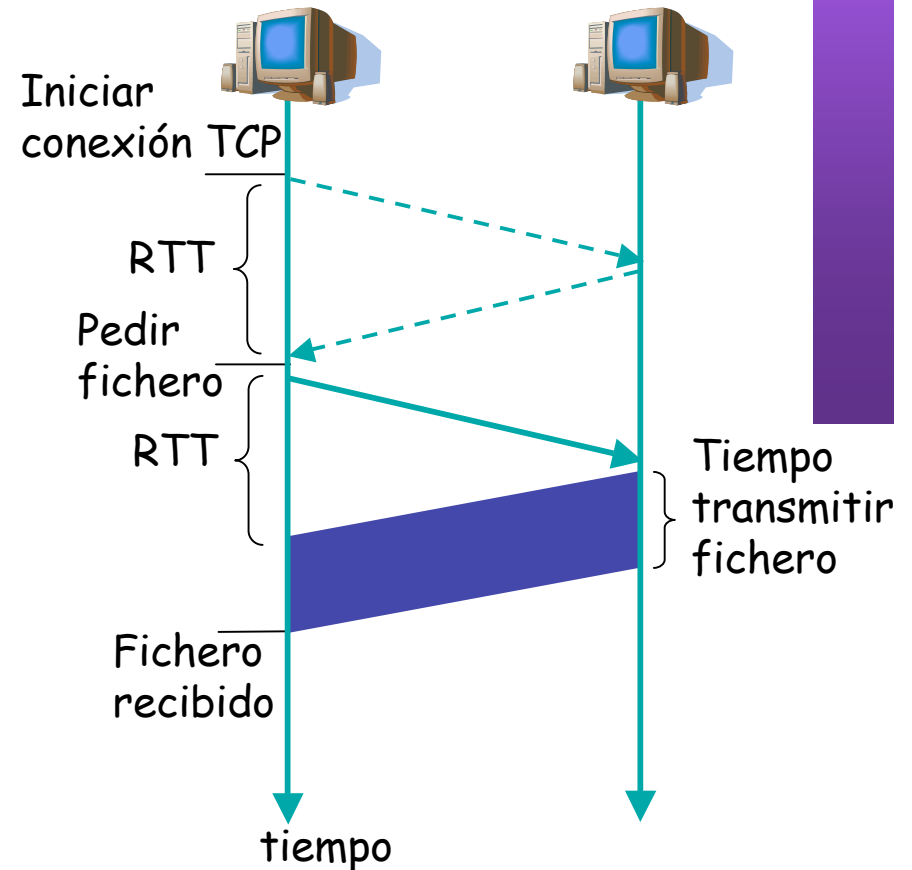


# Modelo del tiempo de respuesta

## 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

$$\text{total} = 2x\text{RTT} + \text{tiempo\_transmisión}$$





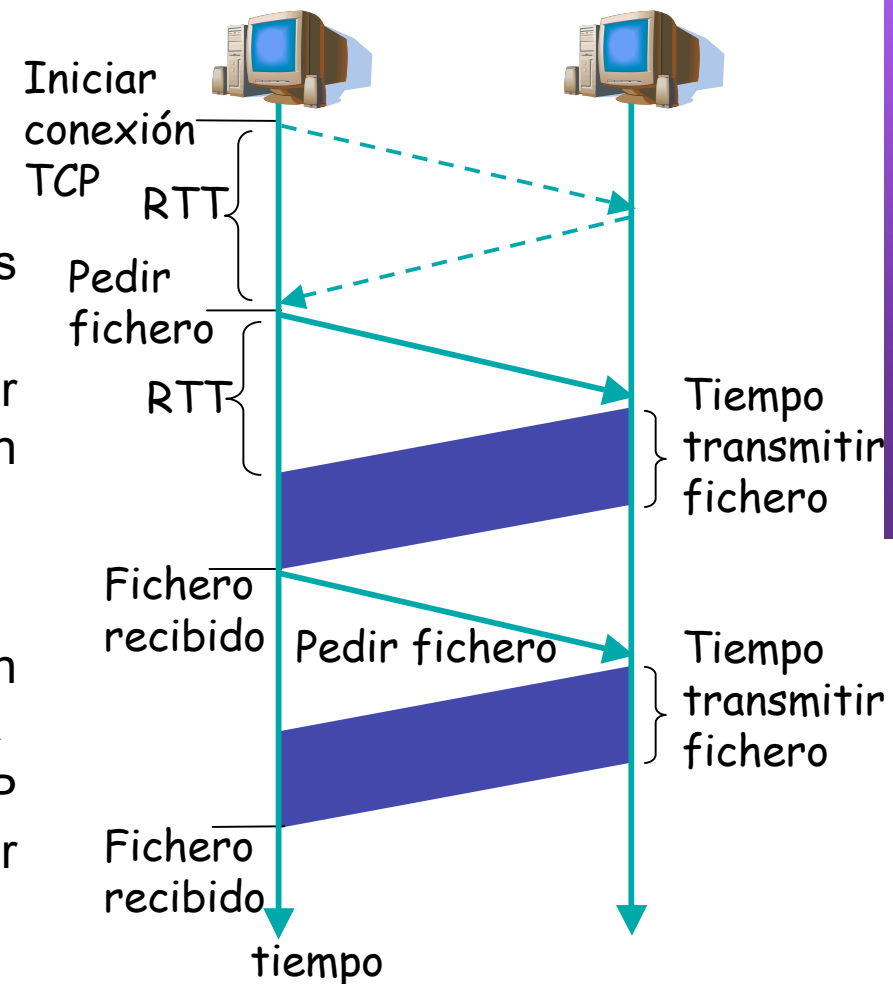
# HTTP persistente

## HTTP no persistente:

- Requiere 2 RTTs por objeto
- 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





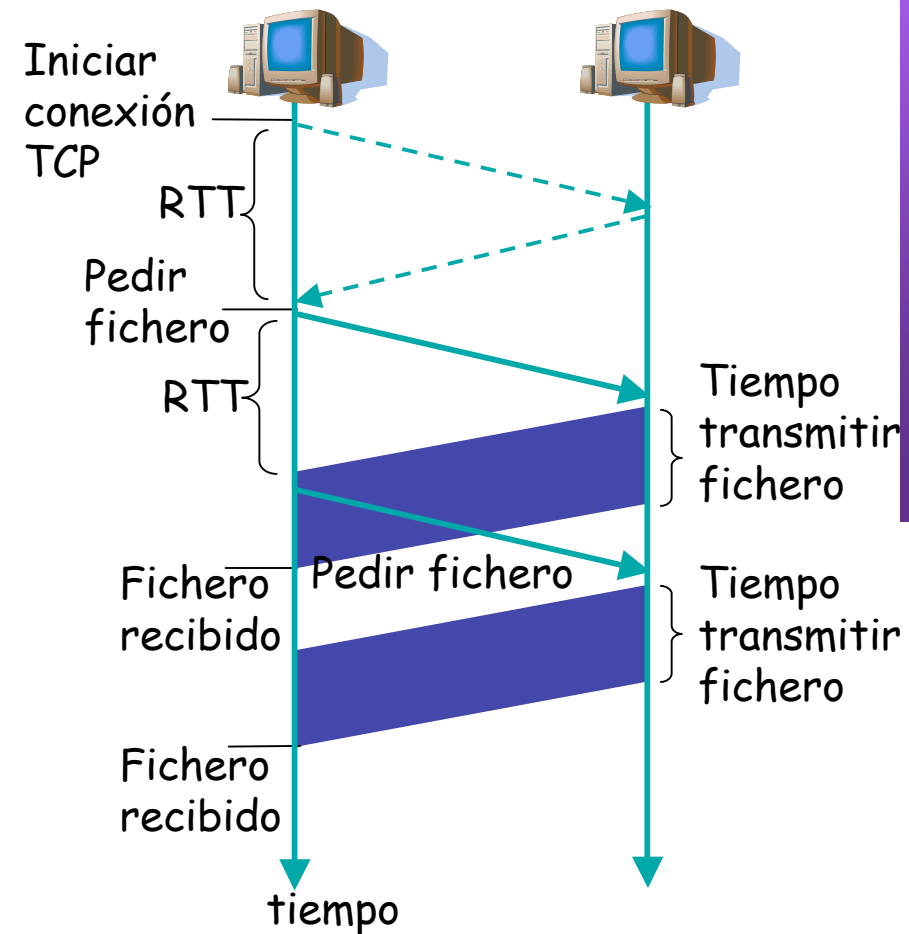
# HTTP persistente

## 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 línea  
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)

HTTP/1.1 200 OK

cabecera

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, ej.,  
fichero HTML  
solicitado

datos datos datos datos datos...



# Probando HTTP desde el cliente

1. 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

2. Escribir una petición GET de HTTP:

```
GET /~daniel/ 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

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
```



# Temario

- 1.- Introducción
- 2.- Nivel de enlace en LANs
- 3.- Interconexión de redes IP
- 4.- Nivel de transporte en Internet
- 5.- Nivel de aplicación en Internet**
  - Nivel de aplicación: WWW
  - FTP. Ejemplos
  - E-mail
  - DNS. P2P. Mensajería...



# Próximas clases

## *Problemas*

### *FTP. Telnet. Ejemplos*

- Lecturas
  - [Kurose05] 2.3
  - [Forouzan03] 19.1
  - 5 páginas

## *Problemas*