Redes de Computadores

Grado en Ingeniería Informática

Departamento de
Tecnología Electrónica

Contenido de la asignatura

Tema 1: Redes de Computadores e Internet

Tema 2: Capa de Aplicación

Tema 3: Capa de Transporte

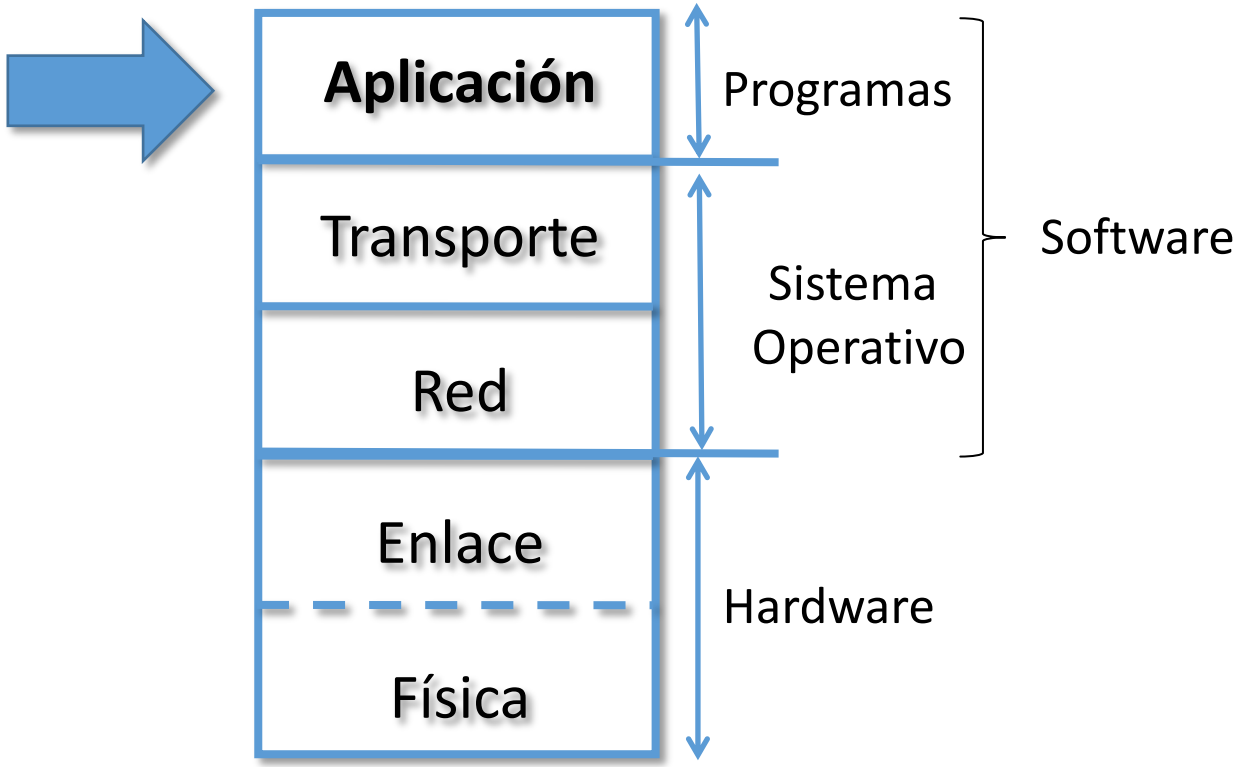
Tema 4: Capa de Red

Tema 5: Capa de Enlace de Datos

Redes de Computadores
Tema 2

La Capa de Aplicación

Departamento de
Tecnología Electrónica


Aplicación
Transporte
Red
Enlace
Física
Programas
Sistema Operativo
Software
Hardware

Tema 2: La Capa de Aplicación

Objetivos

- Conocer qué es la capa de aplicación del modelo TCP/IP y el modelo OSI
- Conocer algunos protocolos básicos de esta capa
- Acercarnos a la programación de la interfaz de acceso al servicio de transporte

Contenido

1. Principios de las aplicaciones en red
2. DNS
3. Web y HTTP
4. Programación de la interfaz de acceso al servicio de transporte

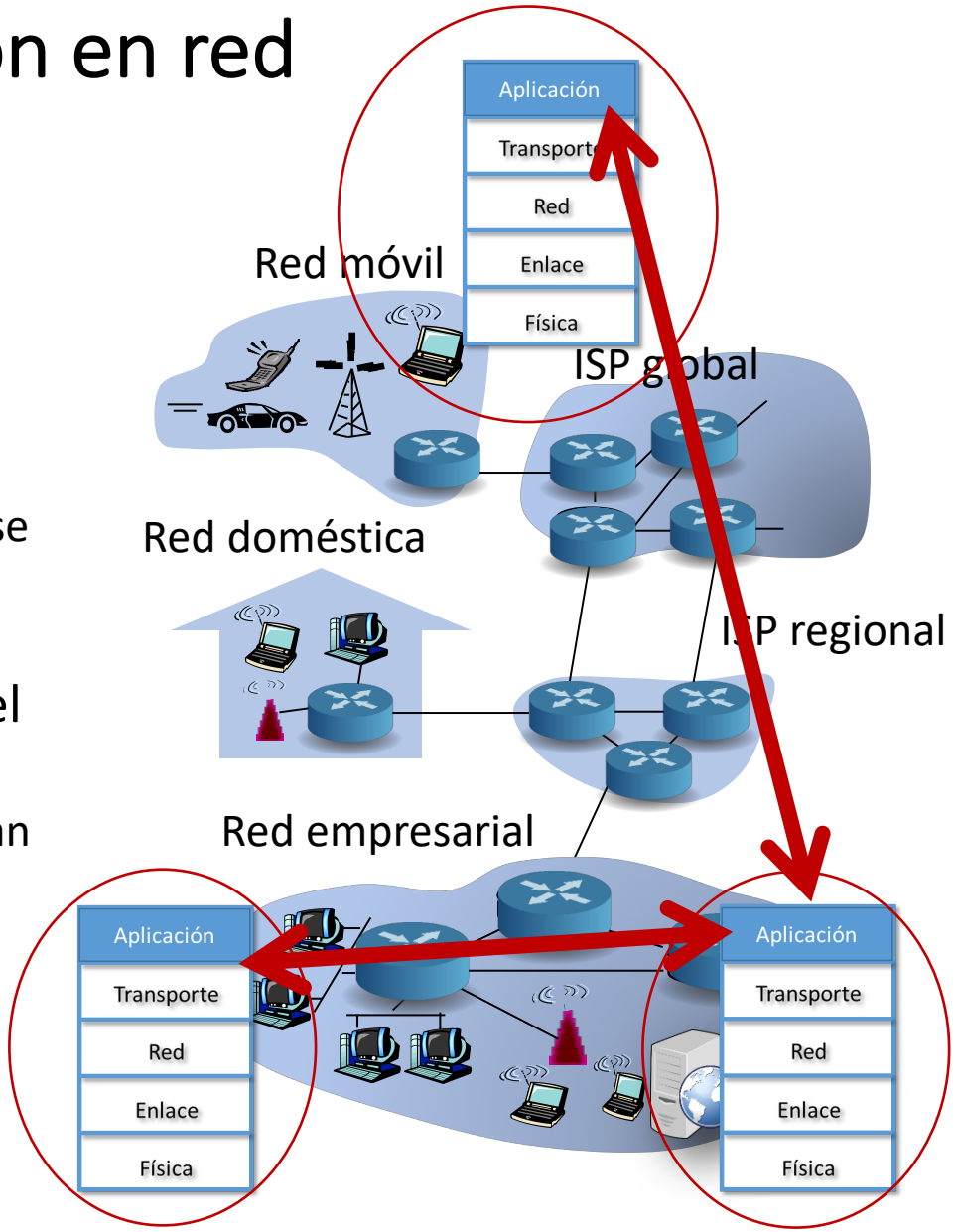
Algunas aplicaciones en red…

Algunas aplicaciones en red…

- e-mail
- web
- Mensajería instantánea
- Acceso remoto
- Compartición de archivos P2P
- Juegos online multiusuario
- Streaming de video (ej. YouTube)
- Voz sobre IP (VoIP)
- Videoconferencia
- Computación en la nube (cloud)
- Etc.

Creando una aplicación en red

- Desarrollar programas que
  - Se ejecuten en (diferentes) sistemas finales
  - Se comuniquen a través de la red
  - Ej: software de un servidor web que se comunica con el software navegador web
- No hay que hacer programas para el núcleo de la red
  - Los dispositivos del núcleo no ejecutan las aplicaciones de los usuarios
  - Al hacerlo en los sistemas finales se acelera el tiempo de desarrollo y propagación de la aplicación



Tema 2: La Capa de Aplicación

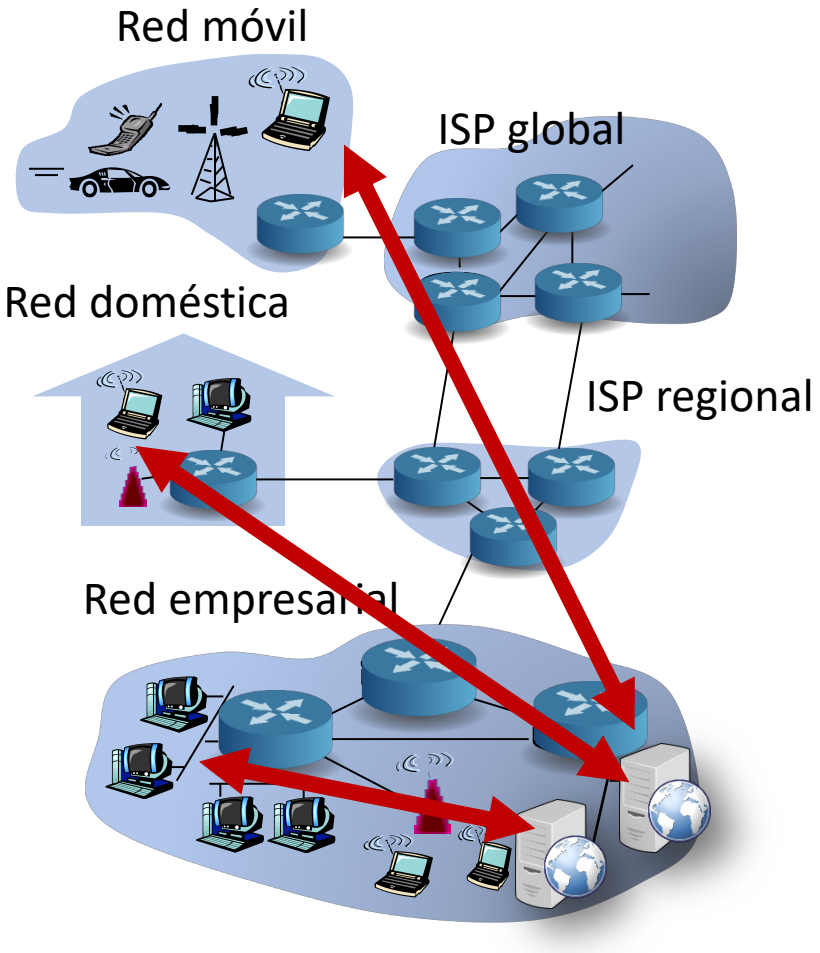
Objetivos

- Conocer qué es la capa de aplicación y aspectos de implementación de protocolos de aplicación en red
  - modelos de servicio del nivel de transporte
  - paradigma cliente-servidor
  - paradigma P2P
- Conocer algunos protocolos básicos de esta capa, como ejemplos
- Acercarnos a la programación de la interfaz de acceso al servicio de transporte

Contenido

1. **Principios de las aplicaciones en red**
2. DNS
3. Web y HTTP
4. Programación de la interfaz de acceso al servicio de transporte

Principios de las aplicaciones en red
Arquitectura Cliente-Servidor



Servidor:

- Equipo siempre-ON
- Dirección IP fija
- Granjas de servidores por escalabilidad

Clientes:

- Se comunican con el servidor
- De manera intermitente
- Con IPs dinámicas o fijas
- No se comunican directamente entre ellos

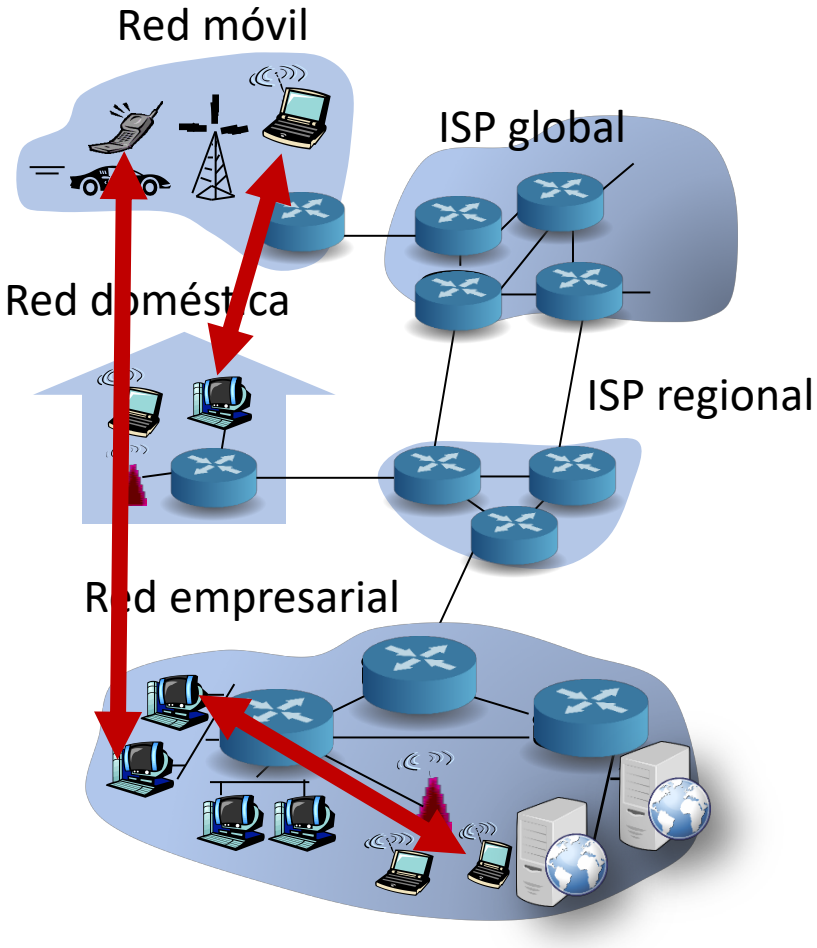
Dirección IP: identifica de forma única a los equipos (sistemas finales, routers,…) conectados a una red TCP/IP. La asigna el ISP de forma estática (fija) o dinámica (variable). Más en tema 4…

Principios de las aplicaciones en red

Arquitectura P2P

- El servidor no está siempre-ON
- Los sistemas finales se comunican entre sí de manera arbitraria
- Los peers se comunican de manera intermitente y con direcciones IP distintas en cada ocasión

Muy escalable pero difícil de gestionar



Dirección IP: identifica de forma única a los equipos (sistemas finales, routers,…) conectados a una red TCP/IP. La asigna el ISP de forma estática (fija) o dinámica (variable). Más en tema 4…

Principios de las aplicaciones en red

Ejemplos de arquitectura híbrida cliente-servidor + P2P

Skype

- Aplicación voz-sobre-IP arquitectura P2P
- Servidor centralizado: encontrar dirección IP del interlocutor remoto
- Conexión cliente-cliente: directa (sin pasar por el servidor)

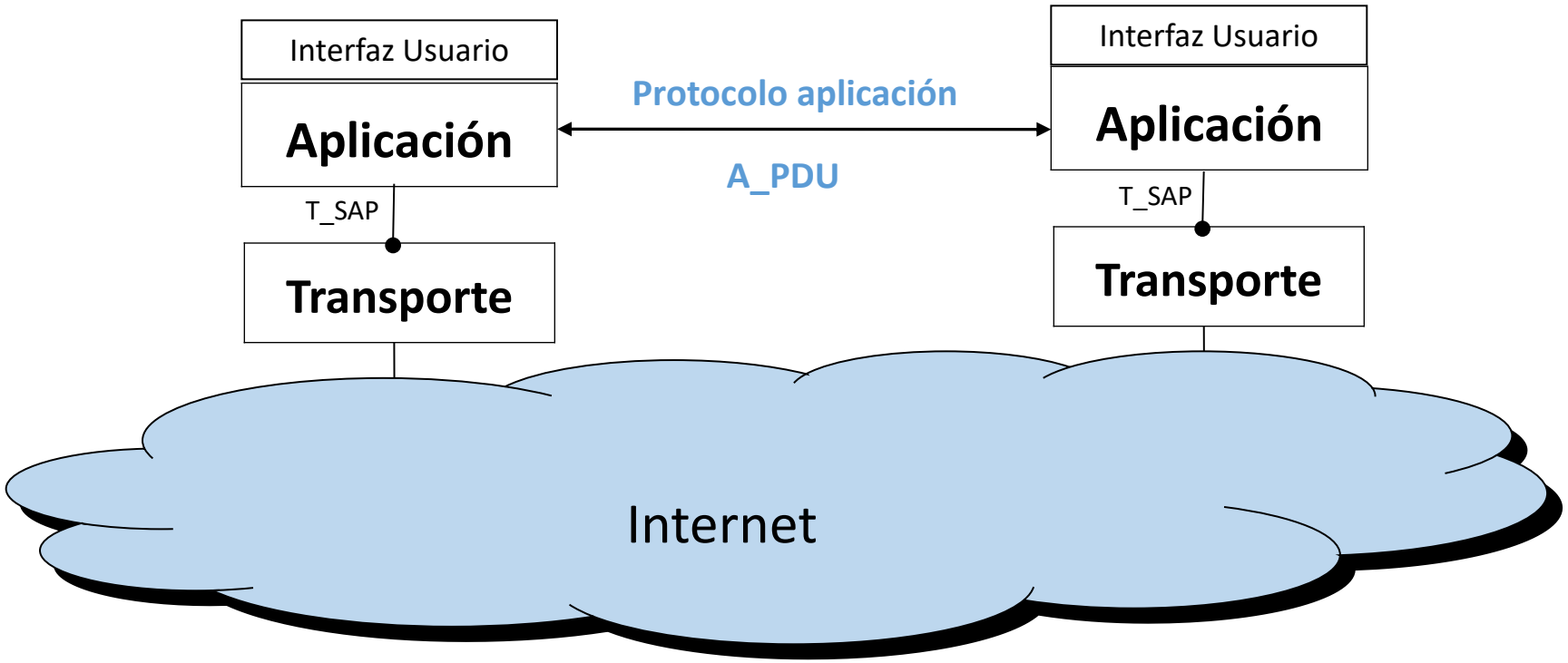
Mensajería instantánea

- La charla entre dos usuarios es P2P
- Servidor centralizado: detecta presencia y localización de los clientes
  - Los usuarios registran su IP con el servidor central al conectarse
  - Los usuarios dialogan con el servidor central en busca de la IP de su contacto

Dirección IP: identifica de forma única a los equipos (sistemas finales, routers,…) conectados a una red TCP/IP. La asigna el ISP de forma estática (fija) o dinámica (variable). Más en tema 4…

Principios de las aplicaciones en red

¿Cómo se implementa la capa de aplicación?



Navegadores Web, p.e: Chrome, Firefox, Microsoft Edge, Safari, Opera, Tor…

Principios de las aplicaciones en red

El protocolo de nivel de aplicación define…

- Tipo de mensaje a intercambiar,
  - P.e. petición o respuesta
- Sintaxis del mensaje
  - Número de campos y delimitación entre ellos
- Semántica del mensaje
  - Significado de los campos
- Reglas de cómo y cuándo los procesos envían y responden a los mensajes

Protocolos de dominio público:

- Definidos en RFCs
- Permiten la inter-operatibilidad
- Ej: HTTP, SMTP

Protocolos propietarios:

- Ej: Skype, Whatsapp

Principios de las aplicaciones en red

Comunicación entre procesos

Proceso: programa que se ejecuta en un equipo (en nuestro caso implementa un determinado protocolo de aplicación).

- En un mismo equipo, dos procesos se comunican usando comunicación entre-procesos (la proporciona el SO).
- Procesos en equipos diferentes se comunican intercambiando mensajes (PDU) usando los servicios de comunicación (en general los proporciona SO)

Proceso cliente: proceso que inicia la comunicación

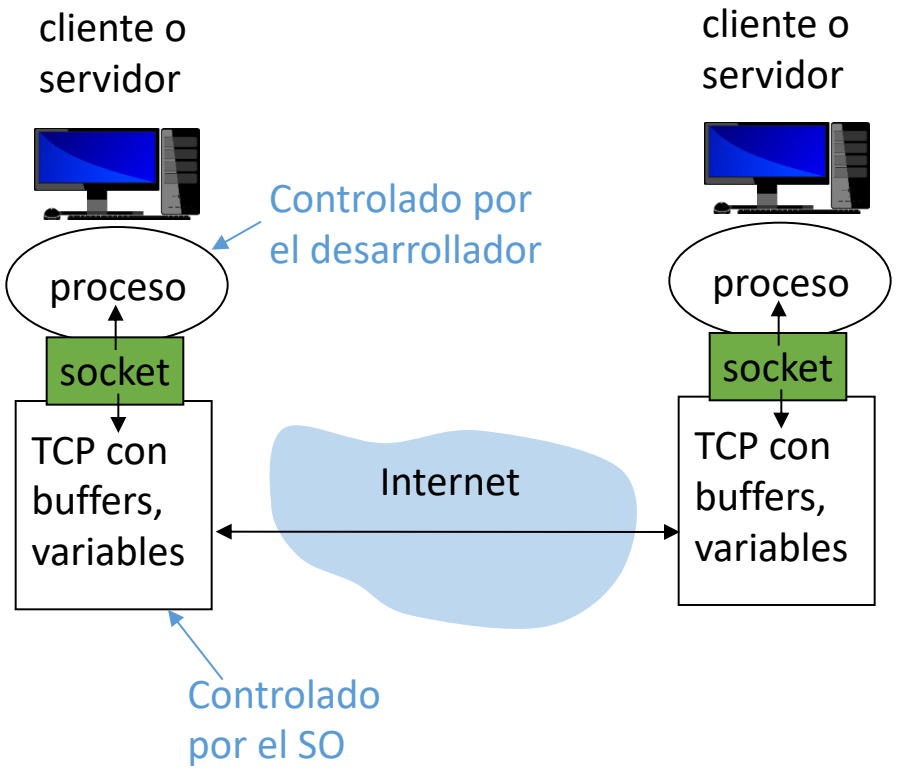
Proceso servidor: proceso que espera a ser contactado

Nota: las aplicaciones P2P combinan ambos procesos, cliente y servidor

Principios de las aplicaciones en red

Sockets (SAP)

- Un proceso envía/recibe mensajes a/de su socket
- Analogía con una puerta:
  - El proceso emisor envía el mensaje a través de la puerta de salida
  - El proceso emisor confía en la infraestructura de transporte que hay detrás de la puerta, encargada de llevar el mensaje hasta la puerta del receptor
- API: (1) elección del servicio de transporte ; (2) posibilidad de fijar parámetros



Principios de las aplicaciones en red
¿Cómo se identifica el socket?

- Para enviar una carta a alguien es necesario saber su dirección para que llegue al buzón de su casa.
  - Cada sistema final tiene un dirección IP única de 32 bits.

> **Nota**
>
> A las direcciones IP se les asocia un nombre, que es el que se utiliza para identificar a los equipos.
>
> Por ejemplo, www.dte.us.es = 150.214.141.196
>
> Más sobre nombres en el apartado siguiente…

¿Es suficiente con la dirección para hacer que llegue la carta a un amigo? No, varias personas pueden estar viviendo en el misma casa.

- Varios protocolos de aplicación pueden estar ejecutándose en un sistema final.
  - Navegador, lector de correo, Skype, …

Principios de las aplicaciones en red
¿Cómo se identifica el socket?

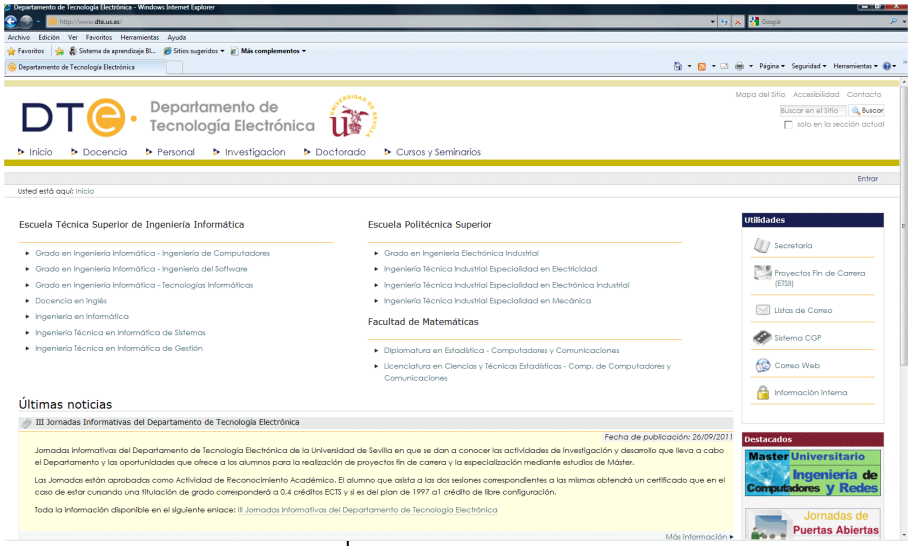
- Cada protocolo de aplicación se identifica por un número de puerto.
- El número de puerto usado para identificar al proceso cliente y servidor en general no coinciden.
- Ej. de número de puerto:
  - Servidor HTTP: 80
  - Servidor HTTPS: 443
  - Servidor Email: 25
  - Servidor DNS: 53

Tipos de puertos:
0 <= P < 1024 son puertos bien conocidos.
1024 <= P < 49152 son puertos de usuario o registrados.
49152 < P <= 65535 son puertos dinámicos, privados o efímeros.

- La ICANN (Internet Corporation for Assigned Names and Numbers), se encarga del registro de los puertos de protocolos de aplicación públicos (http://www.iana.org/assignments/port-numbers)
  - Existen diferentes tipos de puertos.
- Un socket queda identificado por:
  - Dirección IP.
  - Número de puerto.

Principios de las aplicaciones en red

Ejemplo



Servidor web DTE

150.214.141.196, 80

Dir IP cliente, puerto

Transporte

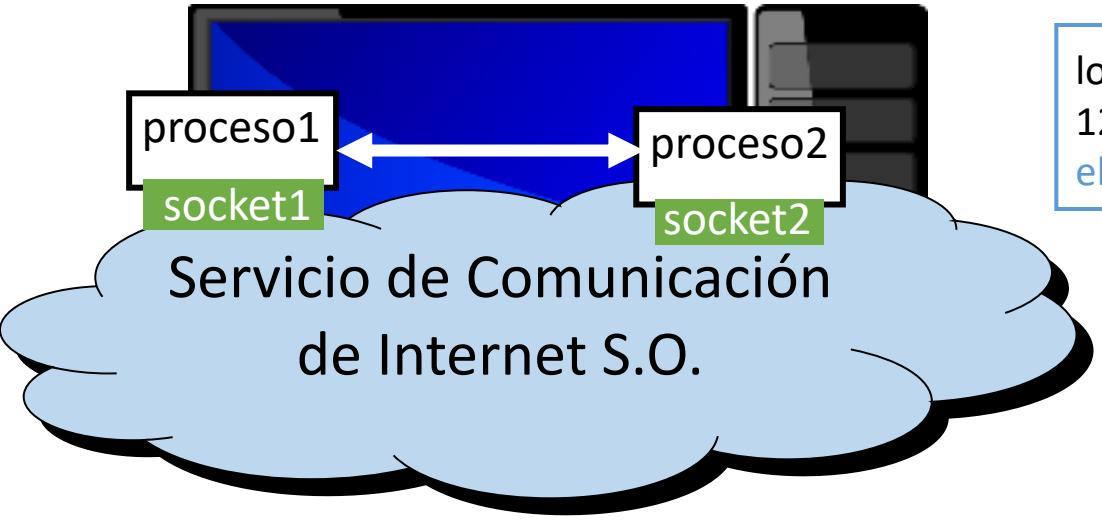
Transporte

Internet

Principios de las aplicaciones en red

Localhost: Conectando 2 procesos del mismo sistema final

- localhost: es un “nombre especial” que está asociado a una dirección IP especial que sirve para identificar al propio sistema final.
- Permite probar aplicaciones en red en un único sistema final sin necesidad de estar conectado a una red.
- En general permite comunicar procesos en un mismo sistema final usando los servicios de comunicaciones de Internet.



Nota

localhost suele tener asociado la IP 127.0.0.1, aunque puede ser otra. Más en el tema 4…

P: ¿Por qué el Servicio de Comunicación del sistema final es capaz de distinguir a cada proceso?

Principios de las aplicaciones en red
¿Qué servicios de transporte necesito?

Pérdida de datos

- Algunas aplicaciones toleran algo de pérdida (ej: audio, video)
- Otras requieren 100% de fiabilidad (ej: login, transferencia de archivos)

Temporización

- Algunas aplicaciones precisan de retardos cortos para ser 'efectivas' (ej: telefonía por Internet, juegos interactivos)

Tasa de transferencia

- Algunas requieren una tasa mínima para funcionar adecuadamente (ej: multimedia)
- Otras, conocidas como “aplicaciones elásticas”, hacen uso de la tasa disponible en cada momento

Seguridad

- Encriptación, integridad de los datos, …

Principios de las aplicaciones en red

Requisitos de algunas aplicaciones comunes

Aplicación	Pérdida de datos	Tasa de transferencia	Sensible al tiempo
Transferencia/descarga de archivos	Sin pérdidas	Elástica	No
Correo electrónico	Sin pérdidas	Elástica	No
Documentos web	Sin pérdidas	Elástica (pocos kbps)	No
Telefonía por Internet/ Videoconferencia	Tolerante a las pérdidas	Audio: unos pocos kbps - 1 Mbps Vídeo: 10 kbps - 5 Mbps	Sí; décimas de seg
Flujos de audio/vídeo almacenado	Tolerante a las pérdidas	Audio: unos pocos kbps - 1 Mbps Vídeo: 10 kbps - 5 Mbps	Sí; unos pocos seg
Juegos interactivos	Tolerante a las pérdidas	Unos pocos kbps – 10 kbps	Sí; décimas de seg
Mensajería para teléfono inteligente	Sin pérdidas	Elástica	Sí y no

Principios de las aplicaciones en red
Servicios de los protocolos de Internet

Servicio TCP

- **Orientado a conexión**: requiere acuerdo previo entre los procesos cliente y servidor antes de iniciar la transferencia
- **Transporte fiable** entre procesos emisor y receptor
- **Control de flujo**: emisor no saturará al receptor
- **Control de congestión**: uso equitativo del ancho de banda
- **No provee**: temporización, garantizar un ancho de banda, seguridad

Servicio UDP

- Transporte ligero, no orientado a conexión y no confiable entre procesos emisor y receptor
- **No provee**: acuerdo previo entre procesos, fiabilidad, control de flujo, control de congestión, temporización, ancho de banda garantizado, ni seguridad.

P: ¿Qué utilidad tiene UDP?

Principios de las aplicaciones en red

Ejemplos: Protocolos de aplicación y transporte

Aplicación	Protocolo de la capa de Aplicación	Protocolo de transporte subyacente
Correo electrónico	SMTP [RFC 5321]	TCP
Acceso remoto a terminal	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2116]	TCP
Transferencia de archivos	FTP [RFC 959]	TCP
Flujos multimedia	HTTP (p.e. YouTube)	TCP
Telefonía por Internet	SIP [RFC 3261], STP[RFC 3550] o propietario (p.e. Skype)	UDP o TCP

Tema 2: La Capa de Aplicación

Objetivos

- Conocer qué es la capa de aplicación del modelo TCP/IP y el modelo OSI
- Conocer algunos protocolos básicos de esta capa
- Acercarnos a la programación de la interfaz de acceso al servicio de transporte

Contenido

1. Principios de las aplicaciones en red
2. **DNS**
3. Web y HTTP
4. Programación de la interfaz de acceso al servicio de transporte

DNS: Domain Name System

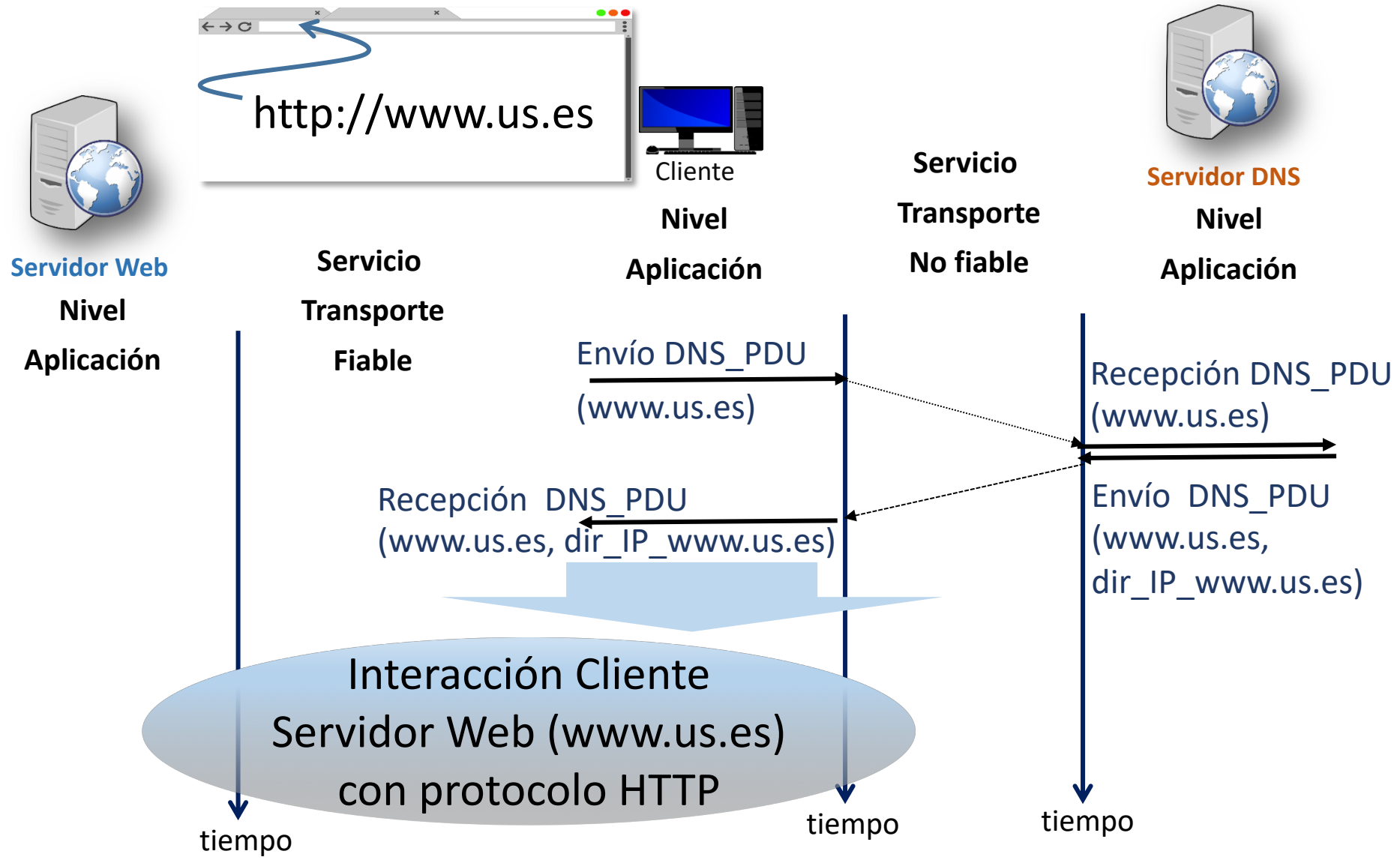
- Las personas tienen muchos IDs: DNI, nombre, nº seguridad social...
- Equipos y routers de Internet:
  - direcciones IP (32 bit) – sirven para direccionar datagramas
  - "nombre", ej: www.google.com – usado por humanos

P: ¿cómo mapeamos entre direcciones IP y nombres y viceversa?

Domain Name System:

- **Base de datos distribuida** implementada con una jerarquía de servidores de nombres
- **Protocolo de nivel de aplicación**: equipos y servidores de nombres se comunican para resolver nombres (traducción de direcciones y de nombres)
- Característica fundamental de Internet, ¡implementada en el nivel de aplicación!

DNS: Funcionamiento simplificado



DNS: Funciones, escalabilidad, transporte

Funciones del servicio DNS

- Traducción de nombre a IP (directa)
- Traducción de IP a nombre (inversa)
- Permite asociar uno o más ”alias” a un nombre
- Permite asignar servidores de correo electrónico a un dominio
- Distribución de carga en servidores web
  - Asociando un conjunto de IPs a un único nombre

¿Por qué no centralizar el servicio DNS?

- Único punto de falla
- Volumen de tráfico
- Distancia a la base de datos
- Mantenimiento
- en definitiva... ¡no sería escalable!

Usa UDP como protocolo de transporte (sin fiabilidad)

- El cliente envía mensajes al puerto 53 del servidor a través del socket.

DNS: Memoria caché

- Una vez que un cliente DNS aprende una traducción, ésta se guarda en una memoria caché.
- Antes de consultar a un servidor DNS se consulta la caché y si se encuentra lo que se busca, no se consulta al servidor DNS.
- Las traducciones más habituales suelen estar ya en caché y esto evita tráfico DNS.
- Las entradas de la caché “caducan” tras un tiempo determinado (timeout).
- Los servidores DNS también tienen memoria caché, pues también actúan como clientes DNS dentro de la jerarquía DNS, preguntando a otros servidores DNS cuando lo necesitan.

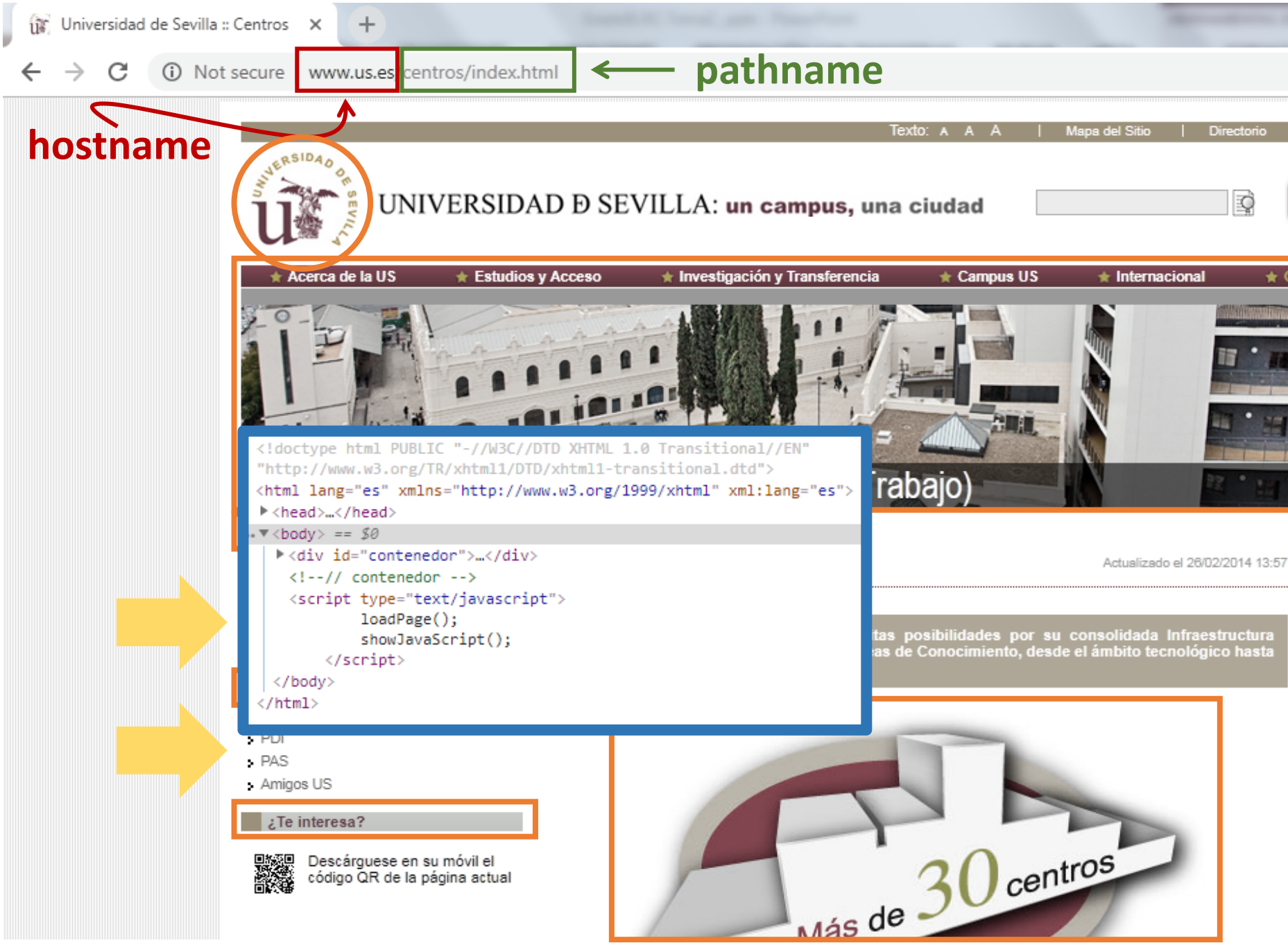
Tema 2: La Capa de Aplicación

Objetivos

- Conocer qué es la capa de aplicación del modelo TCP/IP y el modelo OSI
- Conocer algunos protocolos básicos de esta capa
- Acercarnos a la programación de la interfaz de acceso al servicio de transporte

Contenido

1. Principios de las aplicaciones en red
2. DNS
3. **Web y HTTP**
4. Programación de la interfaz de acceso al servicio de transporte


Universidad de Sevilla :: Centros
Not secure
www.us.es
centros/index.html
pathname
hostname
Texto: A A A | Mapa del Sitio | Directorio
UNIVERSIDAD Ð SEVILLA: un campus, una ciudad
Acerca de la US
Estudios y Acceso
Investigación y Transferencia
Campus US
Internacional
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="es" xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
<head>…</head>
<body> == $0
…
<!--// contenedor -->
<script type="text/javascript">
loadPage();
showJavaScript();
</script>
</body>
</html>
Actualizado el 26/02/2014 13:57
PDI
PAS
Amigos US
¿Te interesa?
Descárguese en su móvil el código QR de la página actual
Más de 30 centros

Web y HTTP

Páginas web y objetos referenciados

- Una página web consiste en un fichero base en HTML que contiene una serie de objetos referenciados.
- Los objetos pueden ser: fichero HTML, imagen JPEG, applet Java, fichero audio, script en JavaScript, hoja de estilos CSS, …
- El fichero base en HTML es, en sí, un objeto más.
- El fichero base no contiene los objetos referenciados, lo que contiene son referencias a los objetos.
- Cada objeto es direccionable a través de su URL
- Ejemplo de URL (Uniform Resource Locator):

hostname
(nombre del servidor
donde está el objeto)

pathname
(nombre de la ruta
al objeto en el servidor)

Web y HTTP

Formato del lenguaje de marcado HTML

Sirve para elaborar las páginas web, desde 1991 (versión actual: HTML5). Se usan elementos con etiquetas entre <>. Cada elemento suele tener 4 campos: una etiq. inicio (<html>) y una etiq. cierre (</html>), unos atributos (en la de inicio) y un contenido (entre ambas). Lo interpreta el cliente.

`<!DOCTYPE html…>`	define inicio documento (opcional)
`<html>página</html>`	define inicio/fin documento
`<head>cabecera</head>`	el contenido de la cabecera (información "no visible" al usuario, como título, estilos, metainformación, etc…)
`<body>cuerpo</body>`	define el cuerpo, contiene:
`<h1> a <h6>`	encabezados
`<table>tabla</table>`	crea una tabla filas/columnas
`<a href="URL">enlace</a>`	define un hipervínculo; al hacer click en "enlace" se solicita la página del URL.
`<img src="URL"/>`	imagen referenciada, el navegador la carga a continuación desde URL para visualizarla.

Nota

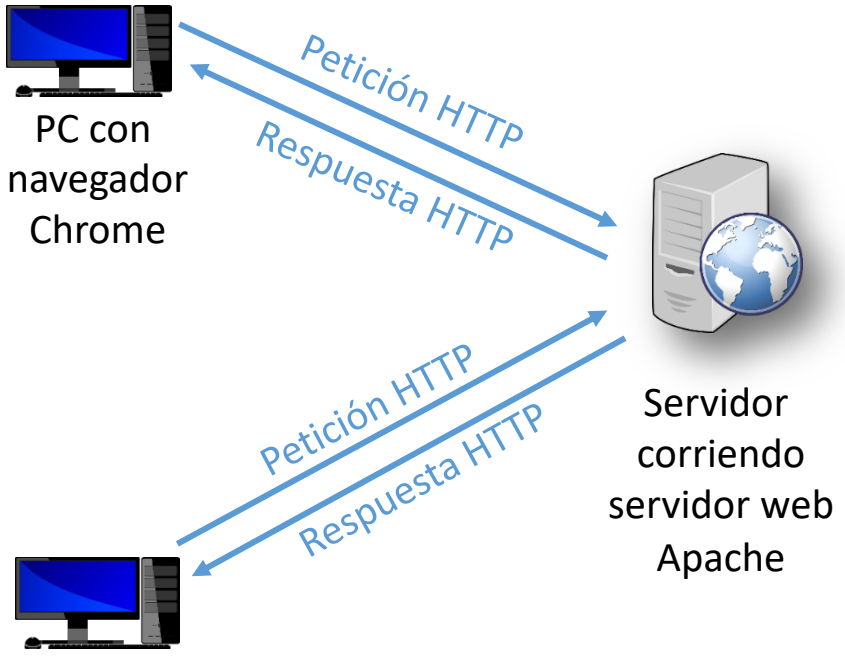
Se puede ver el código HTML de una página en el navegador, botón derecho → ver código fuente

Web y HTTP

HTTP de un vistazo

HTTP: HyperText Transfer Protocol

- Protocolo de nivel de aplicación para la web.
- Modelo cliente/servidor
  - **Cliente**: navegador que pide, recibe y muestra los objetos web.
  - **Servidor**: proceso que envía los objetos pedidos por los clientes.
- Usa transporte orientado a la conexión fiable (**TCP**).
- Es "sin estado"
  - El servidor no guarda información acerca de las peticiones anteriores de los clientes
- Tipos de conexiones HTTP:
  - No Persistentes
  - Persistentes



Petición HTTP:

- Enviada por el cliente
- Transporta información necesaria (HTTP_PCI) para solicitar un objeto del servidor (HTTP_UD)
- Se compone de caracteres ASCII (texto inteligible)

Respuesta HTTP:

- Enviada por el servidor
- Transporta si procede el objeto (HTTP_UD) solicitado por el cliente además de información de control (HTTP_PCI)

Web y HTTP

Tipo de conexiones HTTP

HTTP no persistente

Como máximo se envía un objeto por cada conexión TCP.

HTTP persistente

Se pueden enviar múltiples objetos por una misma conexión TCP entre cliente y servidor.

Web y HTTP

HTTP no persistente

Supongamos que un usuario introduce esta URL:

http://www.us.es/centros/index.html ← (contiene texto y referencias a 6 objetos)

1a. La aplicación cliente HTTP solicita establecer una conexión TCP con el proceso servidor en el equipo www.us.es al puerto 80

1b. La aplicación servidora HTTP en el equipo www.us.es, que estaba a la espera de conexiones TCP en el puerto 80, acepta esta conexión, notificándoselo al cliente.

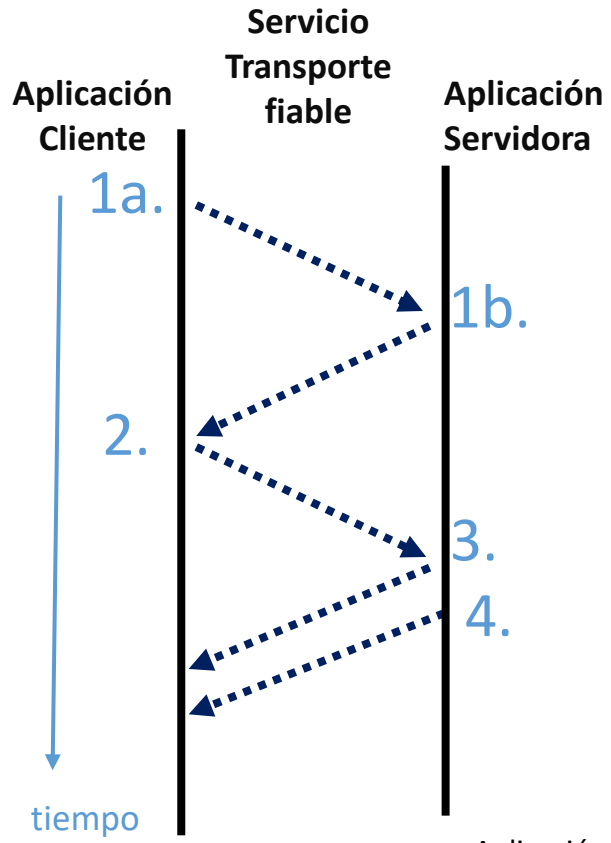
2. El cliente HTTP envía un **mensaje de petición** (que contiene la URL) en la conexión TCP establecida. El mensaje indica que el cliente quiere el objeto /centros/index.html

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

4. El servidor HTTP solicita cierre de la conexión TCP (también lo ha podido hacer el cliente).

5. El cliente HTTP recibe el mensaje de respuesta, conteniendo el fichero HTML, muestra el contenido y lo analiza encontrando 6 referencias a otros objetos.

6. Los pasos 1-5 **se repiten para cada una de los 6 objetos** (4 imágenes y 2 scripts JavaScript) con URLs distintas.



Web y HTTP

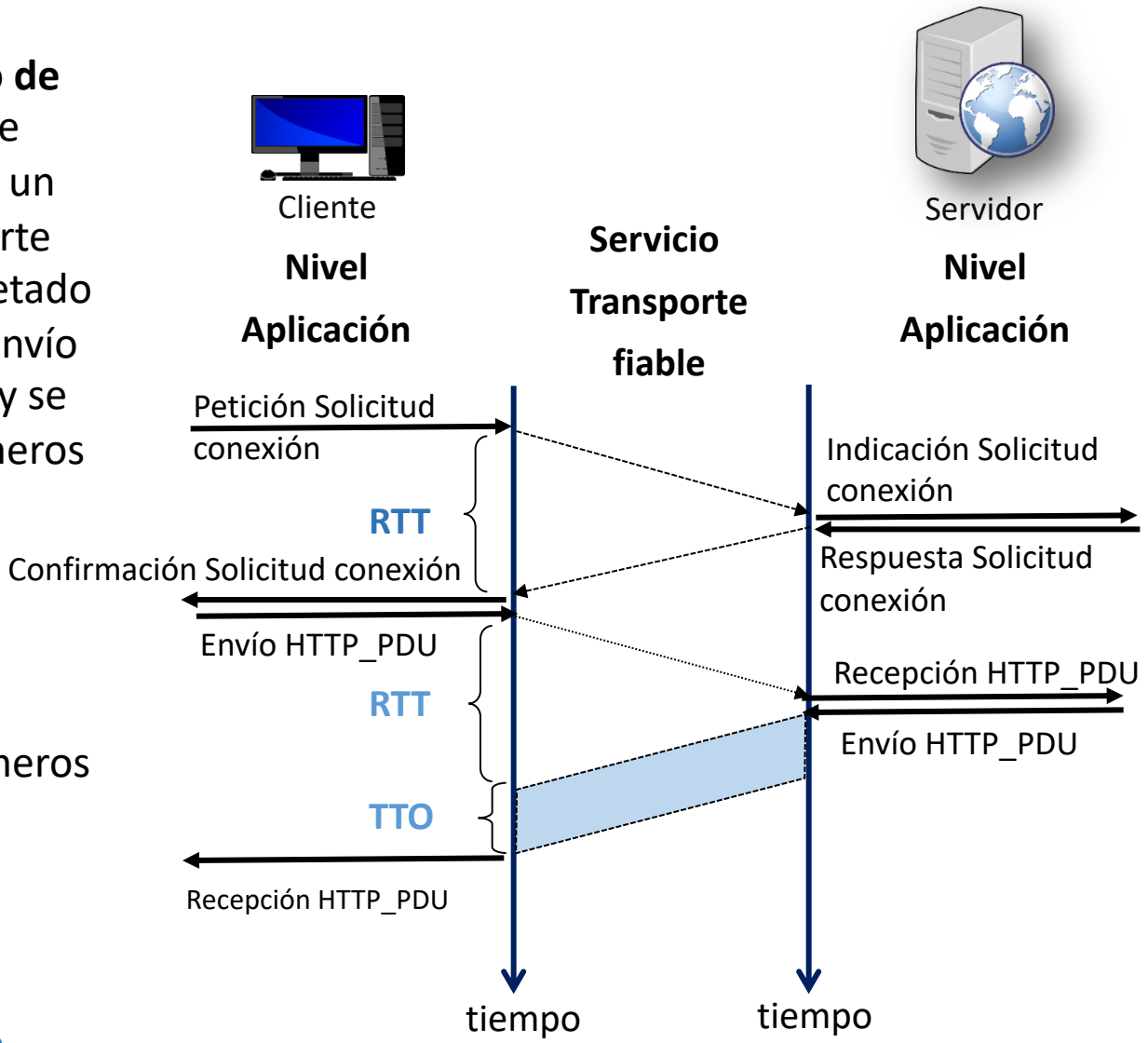
HTTP No-persistente: Tiempo de respuesta

RTT (*Round-Trip Time*, tiempo de ida y vuelta): el tiempo que tarda desde que se solicita un servicio al nivel de transporte hasta que este está completado o desde que se solicita el envío del "mensaje de petición" y se empiezan a recibir los primeros bytes del "mensaje de respuesta".

Tiempo de respuesta (TR):

- 1 RTT, inicio conexión.
- 1 RTT, petición HTTP y primeros bytes de respuesta HTTP.
- Tiempo transmisión bytes objeto solicitado (TTO).

TR= 2RTT + TTO



Web y HTTP

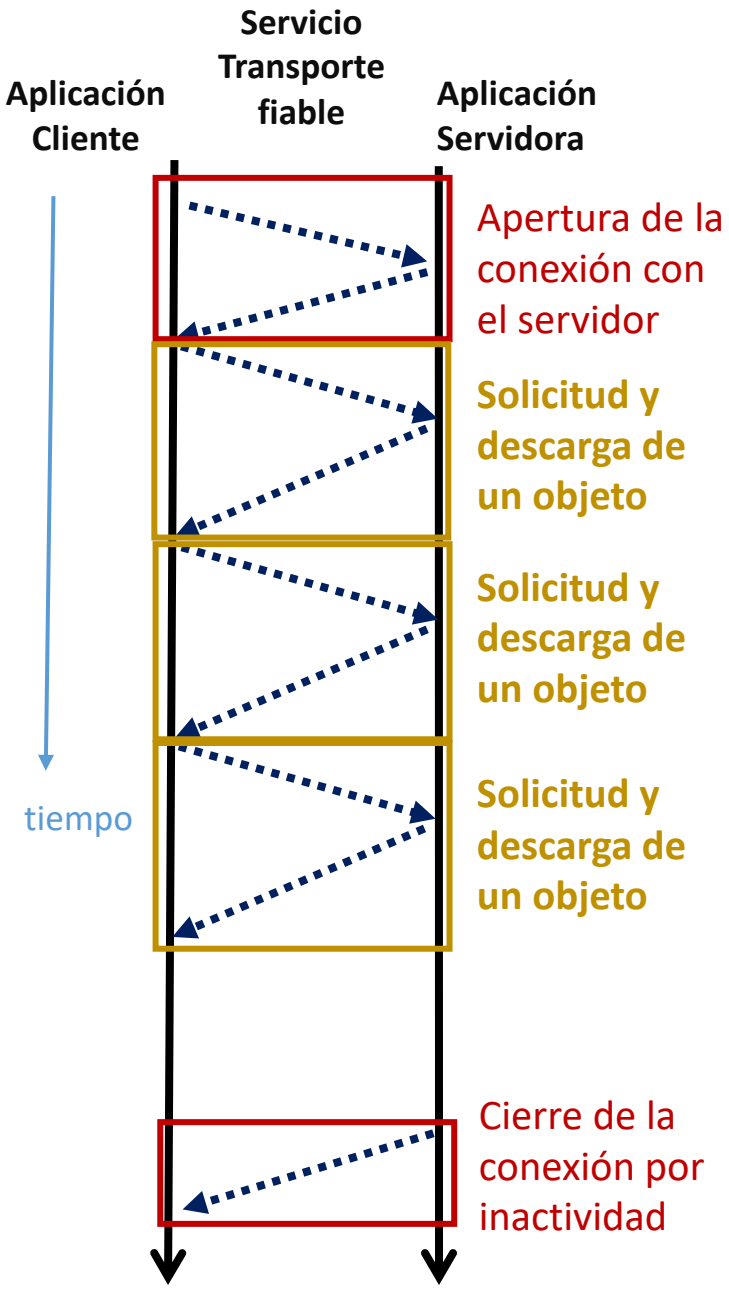
HTTP Persistente

- El servidor mantiene la conexión abierta tras enviar la respuesta.
- Los siguientes mensajes HTTP entre el mismo cliente y el servidor se envían por la conexión abierta.
- El cliente envía una nueva petición cuando acaba de recibir el objeto anterior.
- Cada objeto referenciado tarda sólo 1 RTT (más lo que tarde en transmitirse dicho objeto).

Nota

Conexiones HTTP en paralelo:

Los navegadores a menudo abren varias conexiones TCP en paralelo para obtener los objetos referenciados más rápidamente, los cuales se piden de forma simultánea por cada conexión (sean estas persistentes o no).



Web y HTTP
Mensajes HTTP (o HTTP_PDU)

Hay 2 tipos de mensajes:

- **Petición HTTP**:
  - Enviada por el cliente
  - Transporta información necesaria (HTTP_PCI) para solicitar un objeto del servidor (HTTP_UD)
  - Se compone de caracteres ASCII (texto inteligible)
- **Respuesta HTTP**:
  - Enviada por el servidor
  - Transporta si procede el objeto (HTTP_UD) solicitado por el cliente además de información de control (HTTP_PCI)

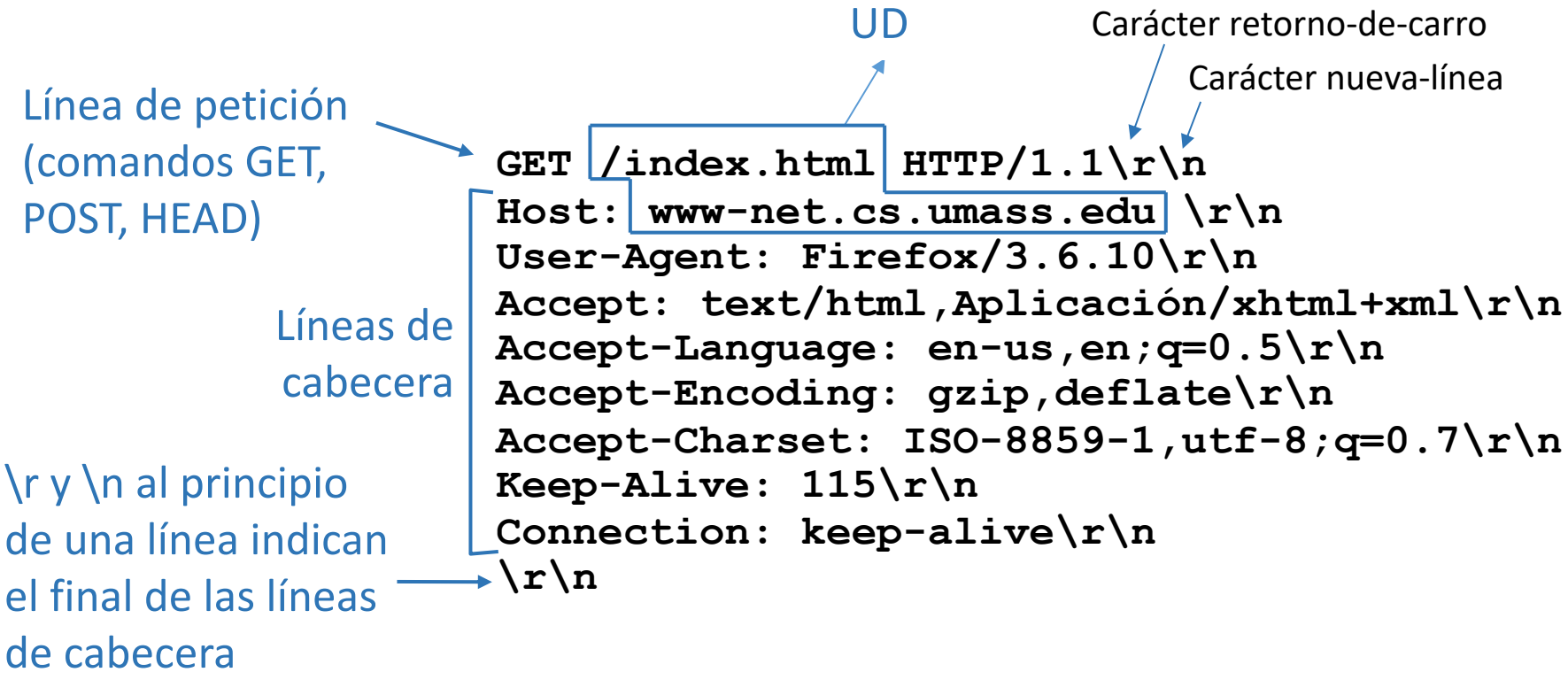
Web y HTTP

Mensaje de Petición HTTP

Nota

<CR>: Carriage-Return : \r

<LF>: Line-Feed: \n

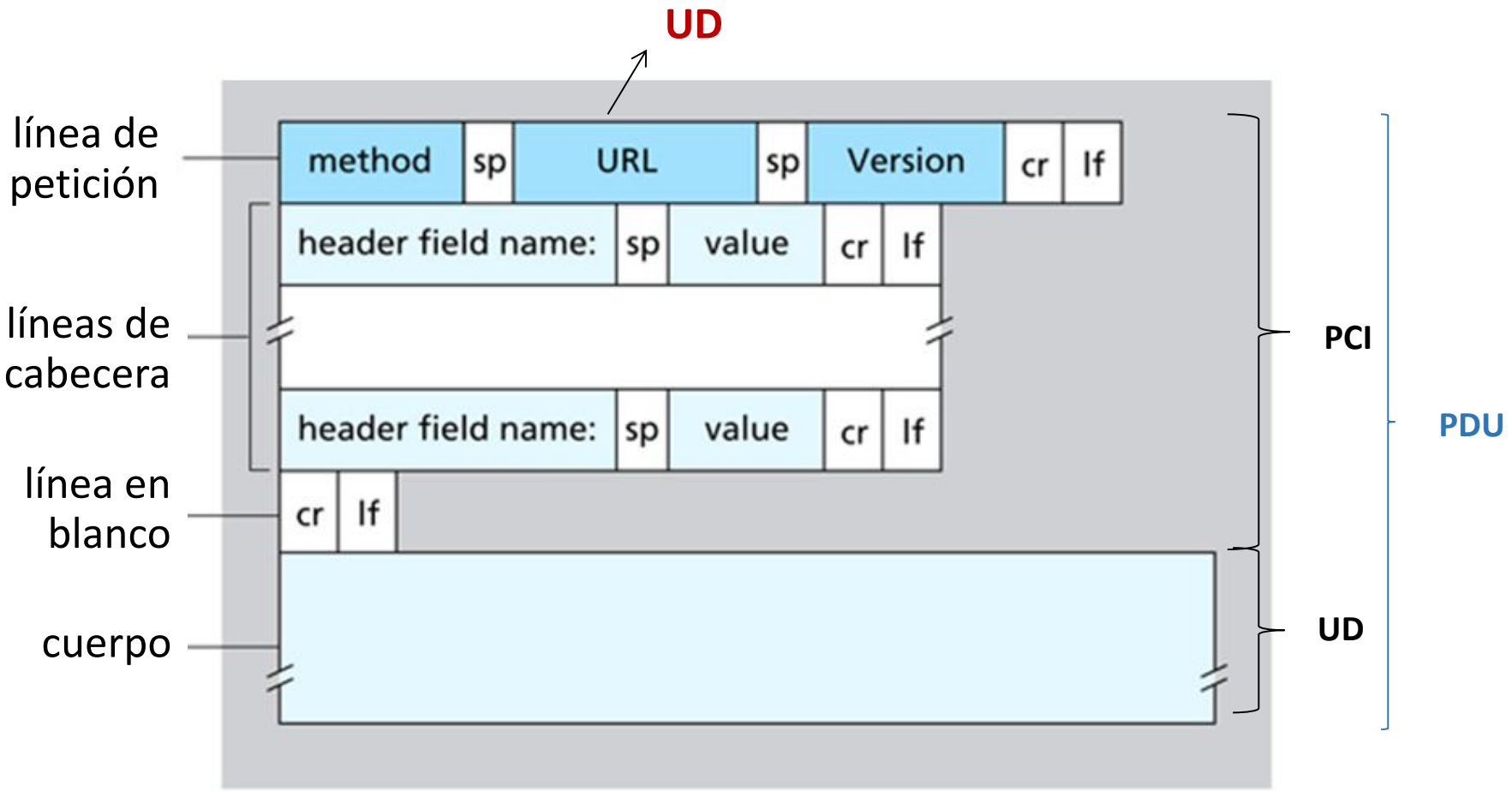


Web y HTTP
Ejemplos de líneas de cabecera del cliente

- Host: **hostname (nombre servidor web). Obligatorio en HTTP 1.1**
- User-Agent: versión_del_navegador
- Accept-**xxx**: lista_de_preferencias_para_**xxx**
- Connection: keep-alive
  - Con “keep-alive” el cliente solicita que la conexión sea persistente.
  - Con ”close” solicitaría conexión no persistente.

Web y HTTP

Mensaje de Petición HTTP: formato general



Method (HTTP 1.1.): GET, POST, HEAD, PUT, DELETE

Web y HTTP

Tipos de métodos

HTTP/1.0 (RFC-1945)

- GET
- POST
- HEAD
  - Idéntico al GET, salvo que no se incluye el objeto en el cuerpo de la respuesta (sólo las cabeceras correspondientes)

HTTP/1.1 (RFC-2616)

- GET
- POST
- HEAD
- PUT
- DELETE

Web y HTTP

Mensaje de Respuesta HTTP

línea de estado (**protocolo**, **código estado**, **frase estado**)

líneas de cabecera

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data data data data data data
data data data data data data data data data data
data data data data data data data data data data
data data data data data ...
```

PCI

UD

CUERPO
ej: archivo HTML pedido

Códigos de estado habituales: 200 OK; 301 Moved Permanently; 400 Bad Request; 404 Not Found; 505 HTTP Version Not Supported

Web y HTTP
Ejemplos de líneas de cabecera del servidor

- Date: fecha (en el que se envía el mensaje)
- Last-Modified: fecha (en la que el objeto se modificó por última vez)
- Server: versión_del_servidor
- Content-Type: tipo_del_objeto (HTML, imagen, …)
- Content-Length: tamaño_del_cuerpo (en bytes)
- Connection: keep-alive
    - Con “keep-alive” el servidor confirma al cliente que esa conexión será persistente. Con “close” es lo contrario.
- Keep-Alive: timeout=ttt, max=nnn
    - El servidor cerrará la conexión persistente tras ttt segundos de inactividad o tras solicitarse nnn objetos por dicha conexión.

Web y HTTP

Ejemplos de códigos de estado en las respuestas del servidor

- 200 OK

  Petición exitosa, el objeto solicitado va a continuación...

- 301 Moved Permanently

  El objeto pedido se ha movido permanentemente. Se especifica el URL de la nueva ubicación usando la línea de cabecera “Location:”

- 400 Bad Request

  Mensaje de petición no entendido por el servidor

- 404 Not Found

  El documento solicitado no se encuentra en el servidor

- 505 HTTP Version Not Supported

  El servidor no soporta esa versión del protocolo HTTP

Web y HTTP

Cookies: manteniendo “el estado”

Muchos servicios web usan cookies. Tienen **cuatro componentes**:

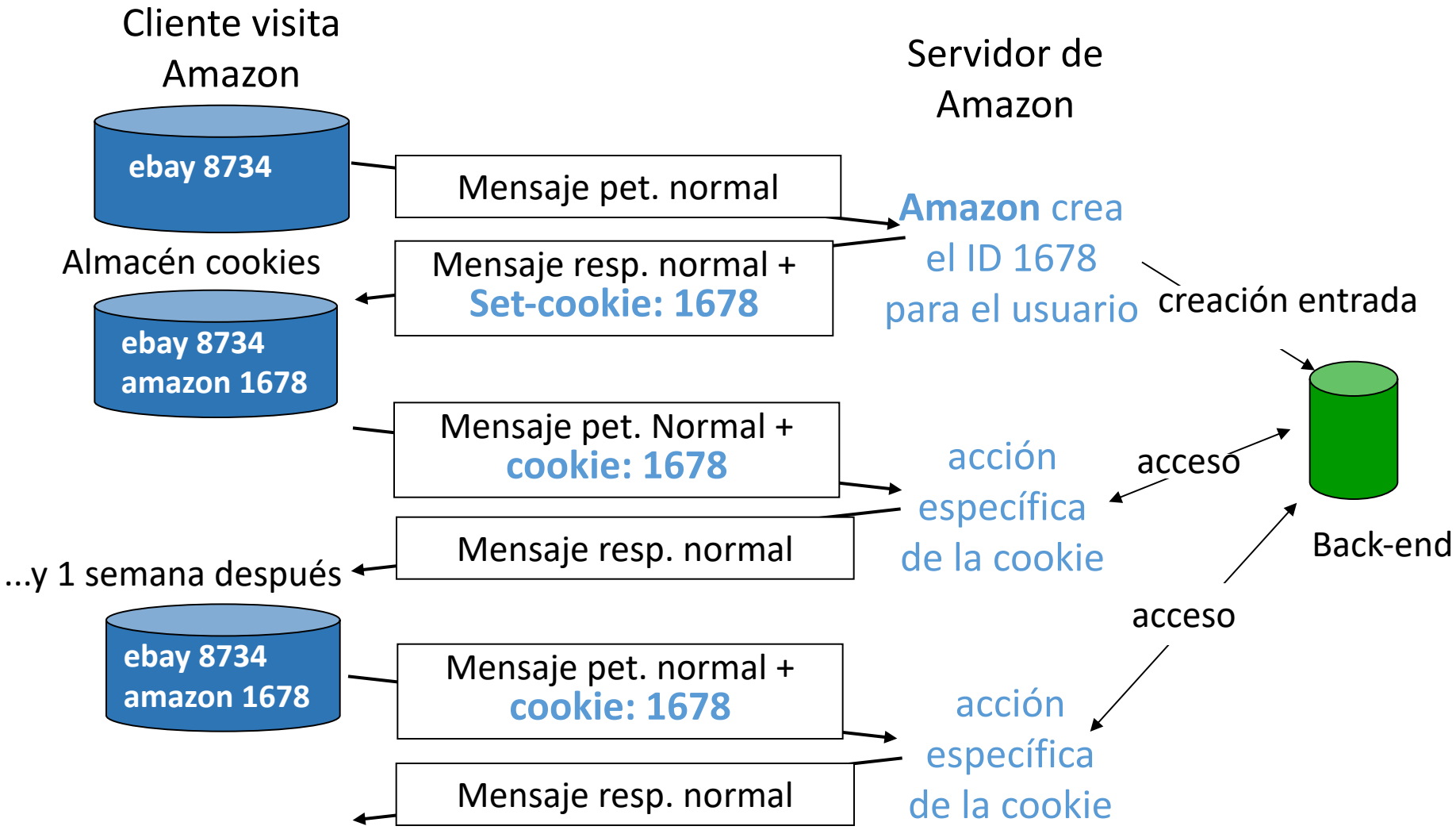
1) cabecera Set-cookie: en el mensaje de respuesta
2) cabecera Cookie: en el mensaje de petición
3) archivo de cookies almacenado por el equipo del usuario y gestionado por el navegador
4) base de datos back-end en el servidor web

Ejemplo de uso:

- Pedro siempre accede a Internet desde su PC
- Visita una tienda online (ej: Amazon) por primera vez
- Al llegar las peticiones, el servidor crea:
  - Un ID único
  - Una entrada para esa ID en la base de datos back-end

Web y HTTP

Cookies: Manteniendo el estado. Ejemplo



Web y HTTP
Cookies: discusión

Posibles aplicaciones:

- autorización
- carritos de la compra
- recomendaciones
- mantenimiento de sesión de usuario (ej: webmail)

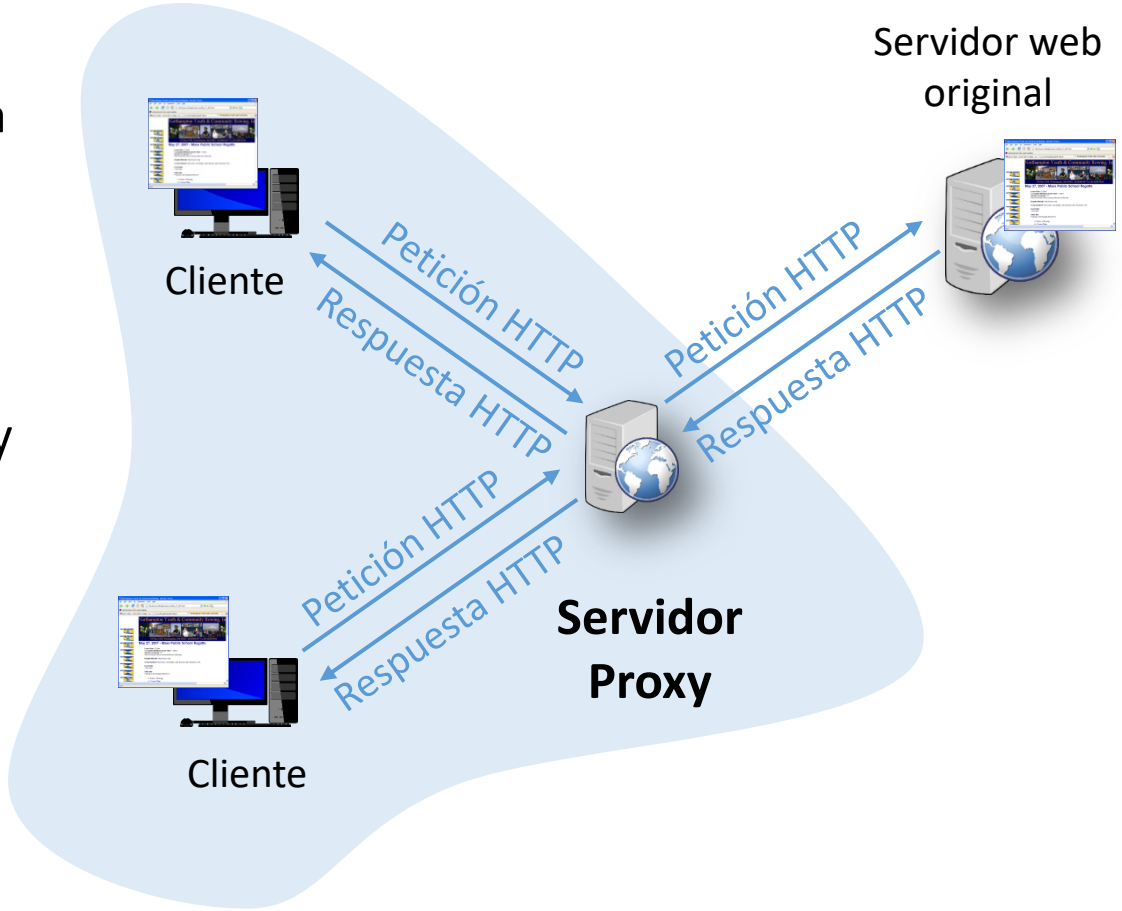
Cookies y la privacidad:

- las cookies permiten a los sitios conocer mucho sobre ti
- puedes estar dando información personal a esas páginas: emails, nombres, etc...

Web y HTTP
Servidor Proxy (Caché de la Web)

Objetivo del Proxy: satisfacer la petición del cliente sin involucrar al servidor web original.

- El Proxy tiene una memoria caché con páginas web.
- El navegador se configura para usar el Proxy.
- Entonces se envían todas las peticiones HTTP al Proxy
  - Si objeto está en la caché, se devuelve el objeto al cliente.
  - Si no, el proxy solicita el objeto al servidor original, lo devuelve al cliente y lo guarda en la caché



Web y HTTP
Más acerca del Proxy

- El caché actúa como cliente (del servidor original) y como servidor (del cliente)
- Normalmente se instalan en los ISP (universidades, compañías, ISPs residenciales)
- Hay que tener en cuenta que no solo el Proxy tiene una caché de páginas. **El navegador del cliente también tiene una memoria caché.**

¿Por qué es interesante?

- Reduce el tiempo de respuesta de la petición del cliente
- Reduce el tráfico de enlace de datos de una institución
- Permite a proveedores “pequeños” entregar de forma eficiente los contenidos (algo que también permite P2P)

Web y HTTP
GET Condicional

Objetivo

Que el servidor web no envíe el objeto si en la la memoria caché hay una versión actualizada del mismo

- El Proxy **(o un navegador con caché)**: especifica la fecha de la copia cacheada en la petición HTTP

  If-modified-since: <date>

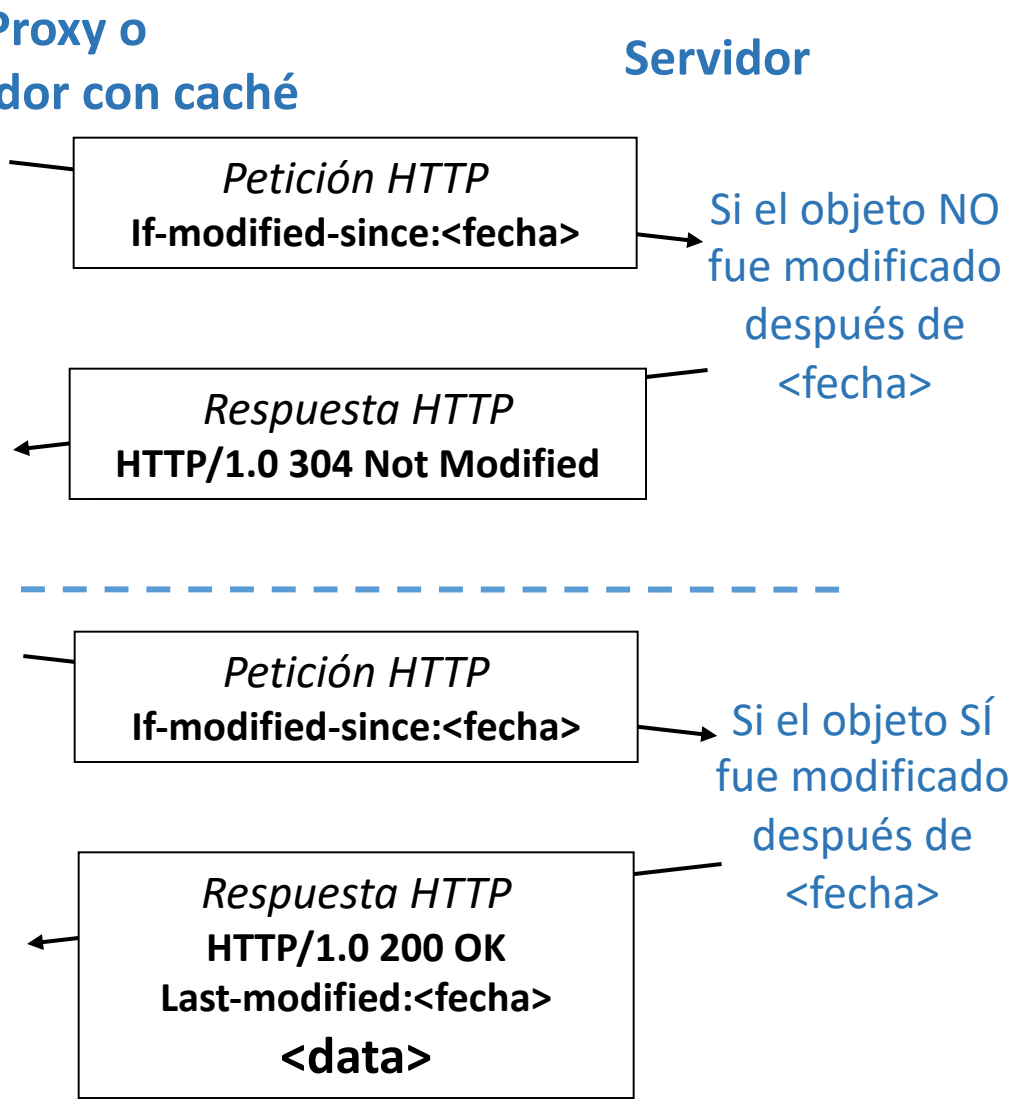
- Servidor: en la respuesta van cabeceras y…

a) no va ningún objeto si la copia no se ha modificado...

HTTP/1.0 304 Not Modified

b) o bien se envía el objeto si está modificado, junto con la fecha de modificación:

Last-modified:<fecha>



Tema 2: La Capa de Aplicación

Objetivos

- Conocer qué es la capa de aplicación del modelo TCP/IP y el modelo OSI
- Conocer algunos protocolos básicos de esta capa
- Acercarnos a la programación de la interfaz de acceso al servicio de transporte

Contenido

1. Principios de las aplicaciones en red
2. DNS
3. Web y HTTP
4. **Programación de la interfaz de acceso al servicio de transporte**

Programación de Sockets

Objetivo

- Aprender cómo se programa una aplicación cliente/servidor que se comunique usando sockets

Socket API

- Se introdujo en BSD4.1 UNIX, 1981
- Los sockets se crean, usan y liberan de forma explícita por las aplicaciones
- Paradigma cliente/servidor
- Dos tipos de servicios:
  - No fiable, orientado a datagramas (UDP)
  - Fiable, orientado a flujo de bytes (TCP)

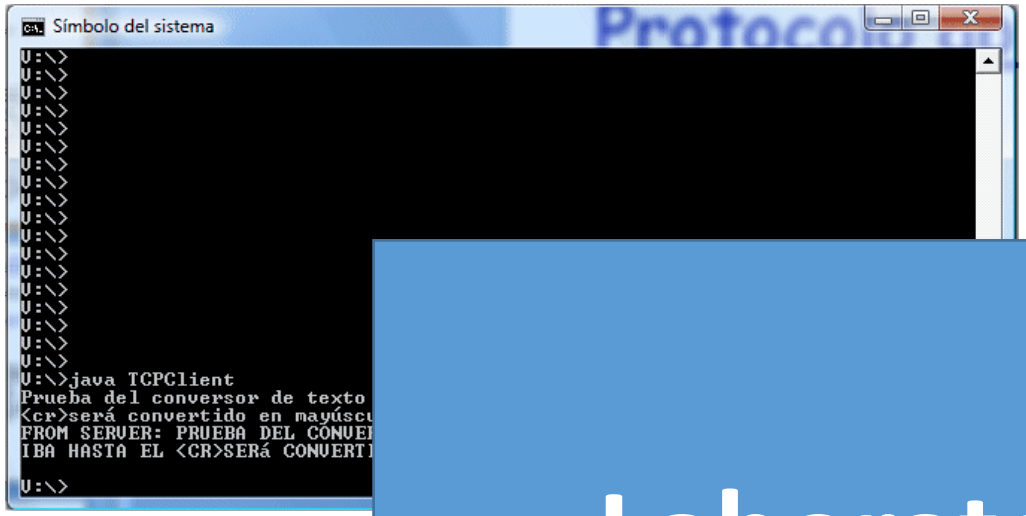
¿Qué es un socket?

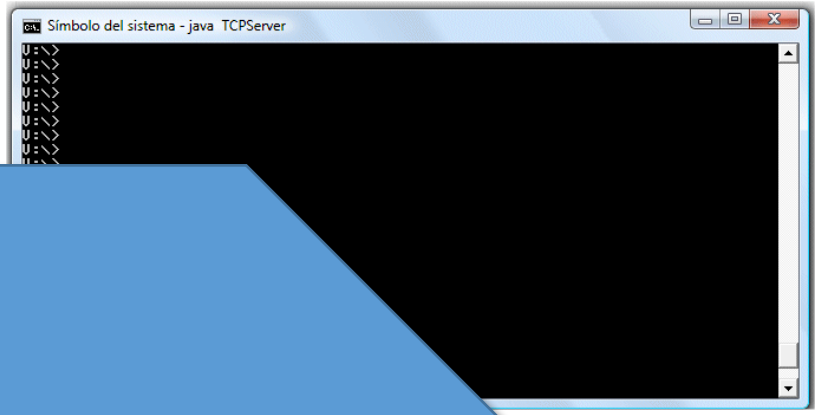
- Una interfaz del equipo local, creada por una aplicación y controlada por el SO (una "puerta") por la que el proceso de aplicación puede tanto enviar/recibir mensajes a/desde otros procesos remotos (o incluso locales).

Protocolo de aplicación

Cliente

Servidor





Dir IP cliente, pue

servidor, puerto

Laboratorio

Internet

Tema 2: La Capa de Aplicación

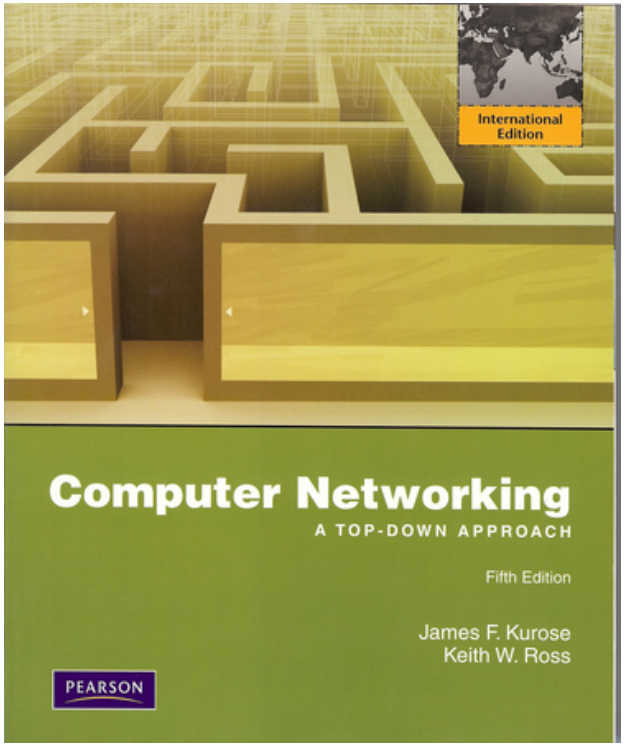
Objetivos

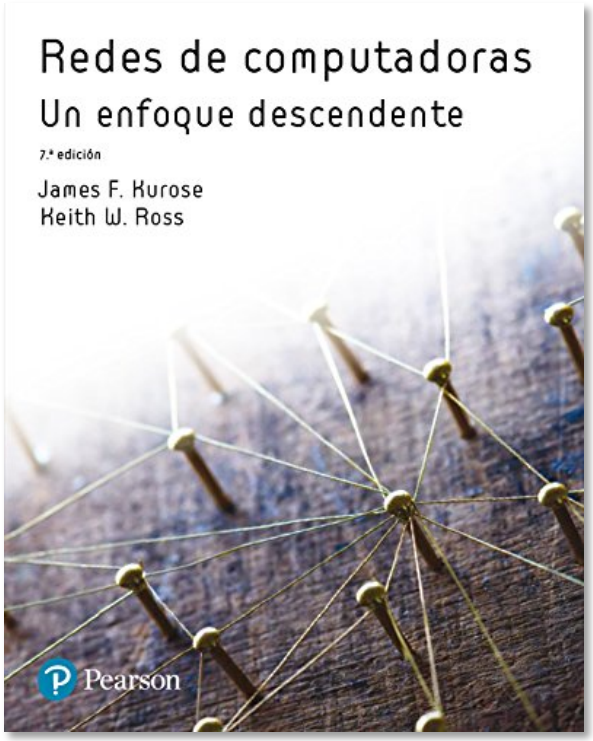
- Conocer qué es la capa de aplicación del modelo TCP/IP y el modelo OSI
- Conocer algunos protocolos básicos de esta capa
- Acercarnos a la programación de la interfaz de acceso al servicio de transporte

Contenido

1. Principios de las aplicaciones en red
2. DNS
3. Web y HTTP
4. Programación de la interfaz de acceso al servicio de transporte

Contenidos





Estas transparencias han sido elaboradas a partir de material con copyright que Pearson pone a disposición del profesorado, a partir del libro:

Jim Kurose, Keith Ross (2010). Computer Networking: A Top Down Approach, 5th edition, Ed. Pearson.

Algunas actualizaciones pertenecen a la última edición:

Jim Kurose, Keith Ross (2017). Redes de Computadoras: Un enfoque descendente, 7ª edición, Ed. Pearson.

Redes de Computadores
Tema 2

La Capa de Aplicación

EJERCICIOS

DTE
Departamento de
Tecnología Electrónica

Ejercicio 1: ¿Verdadero o Falso?

a) Un usuario solicita una página web que consta de texto y 3 referencias a imágenes. Para obtener esa página, el cliente envía un mensaje de solicitud y recibe cuatro mensajes de respuesta.

b) Dos páginas web diferentes (www.mit.edu/research.html y www.mit.edu/students.html) se pueden enviar a través de la misma conexión persistente.

c) Con las conexiones no persistentes entre un navegador y un servidor de origen, un único segmento TCP puede transportar dos mensajes de solicitud HTTP distintos.

d) La línea de cabecera “Date:” del mensaje de respuesta HTTP indica cuándo el objeto fue modificado por última vez.

e) Los mensajes de respuesta HTTP nunca incluyen un cuerpo de mensaje vacío.

Ejercicio 2: Aplicación-Transporte

Un cliente HTTP desea recuperar un documento web que se encuentra en una URL dada. Inicialmente, la dirección IP del servidor HTTP es desconocida.

¿Qué protocolos de la capa de aplicación y de la capa de transporte, además de HTTP, son necesarios en este escenario?

Ejercicio 3: Cabeceras Cliente HTTP

La siguiente cadena ASCII ha sido capturada cuando el navegador enviaba un mensaje GET HTTP.

NOTA: La anchura de las líneas del recuadro es 60 caracteres

```
GET /cs453/index.html HTTP/1.1←↓Host: gaia.cs.umass.edu←↓Use
r-Agent: Mozilla/5.0 (Windows;U; Windows NT 5.1; en-US; rv:1
.7.2) Gecko/20040804 Netscape/7.2 (ax)←↓Accept: ext/xml, app
lication/xml, application/xhtml+xml, text/html;q=0.9, text/p
lain;q=0.8, image/png, */*;q=0.5←↓Accept-Language: en-us,en;
q=0.5←↓Accept-Encoding: zip,deflate←↓Accept-Charset: ISO-885
9-1,utf-8;q=0.7,*;q=0.7←↓Connection: keep-alive←↓←↓
```

NOTA: ← es un retorno de carro y ↓ es un fin de línea.

Responda a las siguientes cuestiones, indicando en que parte del mensaje GET HTTP se encuentra la respuesta a la cuestión:

a) ¿Cuál es la URL del documento solicitado?

b) ¿Qué versión de HTTP se está ejecutando en el navegador?

c) ¿Solicita el navegador una conexión persistente o no?

d) ¿Cuál es la dirección IP del host que corre el navegador?

e) ¿Qué tipo de navegador envía el mensaje? ¿Por qué es necesario indicar el tipo de navegador en el mensaje?

f) ¿Cuántos bytes ocupa la HTTP_PDU enviada por el cliente?

g) ¿Cuántos bytes de HTTP_UD transporta?

Ejercicio 4: Cabeceras Servidor HTTP

La siguiente cadena muestra la respuesta devuelta por el servidor web al mensaje del problema anterior.

NOTA: La anchura de las líneas del recuadro es 60 caracteres

```
HTTP/1.1 200 OK←↓Date: Tue, 07 Mar 2008 12:39:45 GMT←↓Server
: Apache/2.0.52 (Fedora)←↓Last-modified: Sat, 10 Dec 2005 18
:27:46 GMT←↓ETag: "526c3-f22-a88a4c80"←↓Accept-Ranges: bytes
←↓Content-Length: 3874←↓Keep-Alive: timeout=15, max=100←↓Con
nection: keep-alive←↓Content-Type: text/html; charset=ISO-88
59-1←↓←↓<!doctype html public "-//w3c//dtd html 4.0 transiti
onal//en">←↓<html>←↓<head>←↓<meta name="GENERATOR" content="
Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape]">←↓<title>←↓
</head>←↓...Aquí seguiría el resto del documento HTML...
```

NOTA: ← es un retorno de carro y ↓ es un fin de línea.

Responda a las siguientes cuestiones, indicando en que parte del mensaje respuesta HTTP se encuentra la respuesta a la cuestión:

a) ¿Ha encontrado el servidor el documento? ¿En qué momento se suministra la respuesta con el doc.?

b) ¿Cuándo fue modificado por última vez el documento?

c) ¿Cuántos bytes contiene el documento devuelto?

d) ¿Cuáles son los primeros 5 bytes del documento devuelto?

e) ¿Ha acordado el servidor emplear una conexión persistente? (si es que sí diga el tiempo máximo de inactividad que se permite)

f) ¿Cuántos bytes de HTTP_UD transporta?

g) ¿Cuántos bytes ocupa la HTTP_PDU enviada por el servidor?

Ejercicio 5: Tiempo de transferencia (I)

Suponga que en su navegador hace clic en un vínculo a una página web. La dirección IP correspondiente al URL asociado no está almacenada en la caché de su host local, por lo que es necesario realizar una búsqueda DNS.

Suponga que el tiempo de ida y vuelta (RTT) de la consulta al servidor DNS es RTT_{DNS}

Suponga también que la página web asociada con el vínculo es un pequeño fichero HTML (lo que supone un tiempo de transmisión despreciable) y que no contiene referencias a otros objetos.

Sea RTT_0 el tiempo RTT entre el host local y el servidor web.

¿Cuánto tiempo transcurre desde que el cliente hace clic en el vínculo hasta que recibe el objeto?

Ejercicio 6: Tiempo de transferencia (II)

Continuando con el Problema 5, suponga que el archivo base HTML hace referencia a 8 objetos muy pequeños que se encuentran en el mismo servidor.

Despreciando los tiempos de transmisión, para cargar la página web completa, **¿cuánto tiempo transcurre si se utiliza…**

a) …HTTP no persistente sin conexiones TCP en paralelo?

b) …HTTP no persistente con 5 conexiones en paralelo?

c) …1 única conexión HTTP persistente?