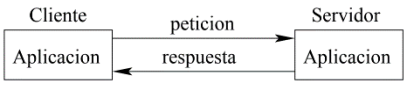
Tema 2: Capa de Aplicación

INTRODUCCIÓN



- La CAPA DE APLICACIÓN se ocupa de la **comunicación entre procesos**.
  - La comunicación de 2 procesos, tanto en el mismo host como en dos hosts diferentes, se realiza enviando **mensajes**.

PROTOCOLOS DE COMUNICACIÓN

- Son necesarios para la **comprensión de los mensajes** entre los procesos.
- Facilitan la programación de las funciones de **envío** y de **recepción**.

Deben especificar:

- **Tipo de mensajes** que se intercambian (petición, respuesta, etc.).
- **Reglas** que especifican cuándo y cómo se envían los mensajes.
- **Sintaxis** del mensaje (los campos que tiene).
- **Semántica** de cada campo (su significado).

TCP	HTTP (web) SMTP, POP3 (correo) FTP (ficheros)
UDP	DNS (traducciones)

SERVICIOS QUE NECESITA LA APLICACIÓN DE RED

- Los protocolos de la capa de aplicación usan por debajo los protocolos básicos de la **capa de transporte**.
  - TCP/IP → servicio fiable orientado a conexión.
  - UDP → servicio no fiable más rápido no orientado a conexión.
- **Ancho de banda**.
- **Temporización**.

AGENTE DE USUARIO

- El AGENTE DE USUARIO (AU) es la **interfaz entre el usuario y la aplicación**, es decir, el programa gráfico que le permite al usuario trabajar cómodamente con la aplicación (navegador, gestor de correo, etc.).

HTTP: PROTOCOLO DE TRANSFERENCIA DE HIPERTEXTO

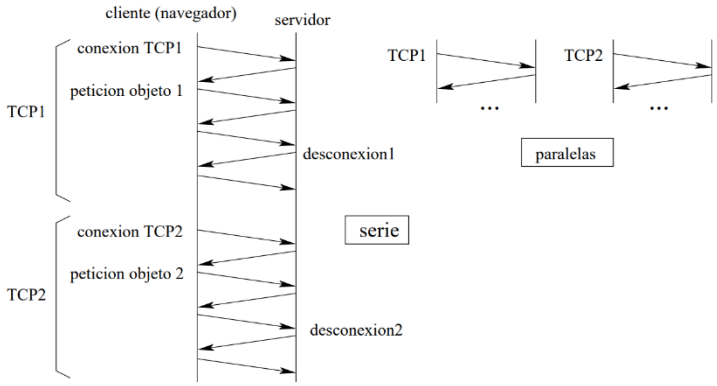
- El HTTP define la comunicación entre un **servidor web** y un **cliente web**.
  - Usa conexiones **TCP**.
    - Usa por defecto el puerto 80.
  - Es un protocolo **sin estado** → una petición se atiende sin tener en cuenta las anteriores. El servidor no almacena información sobre los clientes.
    - A veces usa mecanismos como **cookies**.

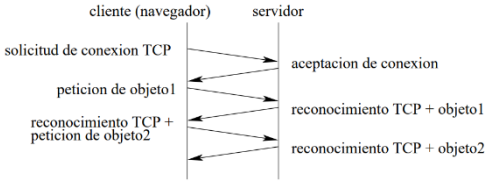
CONCEPTOS BÁSICOS

- OBJETO → archivo direccionable por un URL (documento base, imágenes, animaciones flash, etc.).
- PÁGINA WEB (o documento) → formada por varios objetos.
- SERVIDOR WEB → alberga objetos.
- NAVEGADOR → AU para la web.
- Cuando un navegador solicita una página web, el primer objeto que el servidor devuelve es el **documento base**. No solicitará más objetos hasta que no reciba este.

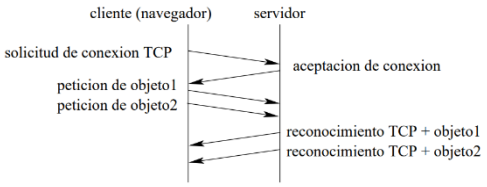
TIPOS DE CONEXIONES HTTP

- **No persistentes** (HTTP 1.0) → se usa una conexión TCP distinta para transmitir cada objeto.
  - **Serie** → se espera a que acabe la conexión TCP previa antes de comenzar la siguiente.
  - **Paralelo** → se inician varias conexiones TCP a la vez.
- **Persistentes** → se pueden transferir varios objetos (e incluso varias páginas) con la misma conexión TCP.
  - **Sin entubamiento** → el cliente pide un nuevo objeto cuando el previo ha sido recibido.
  - **Con entubamiento** (HTTP 1.1) → el cliente puede hacer peticiones de varios objetos antes de recibir los anteriores.





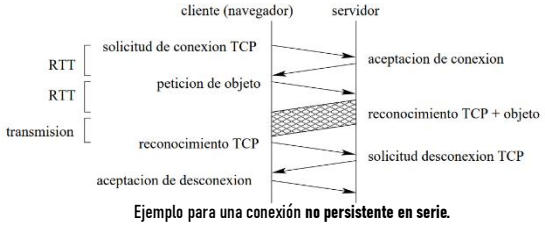
sin entubamiento



con entubamiento

TIEMPO DE TRANSFERENCIA DE UNA PÁGINA WEB

- **RTT** (tiempo de ida y vuelta) → tiempo necesario para que un paquete pequeño (de tamaño despreciable, como 1 bit) vaya del cliente al servidor y vuelva.
  - ↳ Incluye todos los posibles retrasos (tiempo de propagación, espera en colas y de procesamiento).
- Tiempo de **transmisión** del archivo → tiempo necesario para almacenar y reenviar un fichero.
  - ↳ Depende del tamaño de este.
- Tiempo de transferencia del **primer objeto** → $2RTT + t_{transmisión}$ en **todos los tipos de conexión**.
- Tiempo de transferencia en el **resto de objetos** → **depende** del tipo de conexión.



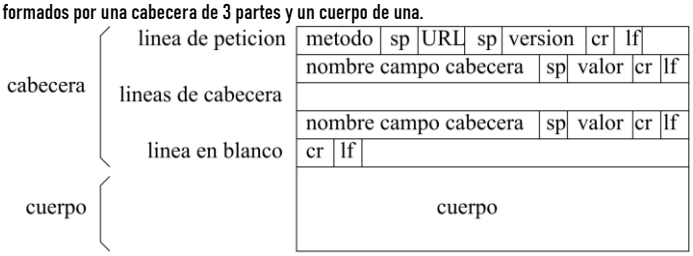
Ejemplo para una conexión **no persistente en serie.**

MENSAJES HTTP

- **Tipos** de mensajes HTTP:
  - **Petición** → se usan para pedir objetos.
  - **Respuesta** → contienen los objetos pedidos.
- **Partes** de los mensajes HTTP:
  - **Cabecera** → contiene información de control en **ASCII** de 7 bits.
  - **Cuerpo** → contiene los datos del mensaje en **binario**.

MENSAJES DE PETICIÓN

Están formados por una cabecera de 3 partes y un cuerpo de una.



- **Línea de petición** → se usa para **pedir un objeto**. Tiene 3 campos separados por espacios:
  1. **Método** → será get para pedir una página normal o un objeto y $post$ para rellenar un formulario.
  2. (localizador de recursos uniforme) → indica el directorio y el nombre del objeto que se pide.
  3. **Versión** del protocolo que desea usar el cliente.
- **Líneas de cabecera** → cada una contiene una **opción.** Puede haber tantas como haga falta. Tienen 2 campos separados por espacios:
  1. **Nombre** de la opción.
  2. **Valor** de la opción
- **Línea en blanco** → separa la cabecera del cuerpo.
- **Cuerpo** → si se solicita una página u objeto con get estará **vacío**. Si se rellena un formulario con $post$, contiene sus **datos**.

Host:	nombre servidor web (puede haber varios para una IP)
Connection:	close (si queremos conexiones no persistentes)
User-agent:	mozilla/4.0 (el navegador)
Accept-languaje:	es (el idioma preferido de la página, si la hay)

MENSAJES DE RESPUESTA

Están formados por una cabecera de 3 partes y un cuerpo de una.

cabecera	linea de estado	version	sp	cod. estado	sp	frase	cr	lf
	lineas de cabecera	nombre campo cabecera			sp	valor	cr	lf
	linea en blanco	nombre campo cabecera			sp	valor	cr	lf
		cr	lf					
cuerpo		objeto						

- **Línea de estado** → Tiene 3 campos separados por espacios:
  1. **Versión** del protocolo que usa el servidor.
  2. **Código de estado** → indica el resultado de la petición.
  3. **Frase** → pequeña explicación del código de estado (ejemplos en la imagen).

200	OK (el objeto o página se sirve sin problemas)
404	Not found (la página no existe)
400	Bad Request (no se entendió el formato de la petición)

- **Líneas de cabecera** → cada una contiene una **opción.** Puede haber tantas como haga falta. Tiene 2 campos separados por espacios:
  1. **Nombre** de la opción.
  2. **Valor** de la opción.
- **Línea en blanco** → separa la cabecera del cuerpo.
- **Cuerpo** → objeto solicitado.

HTTP/2

- El HTTP/2 es la nueva versión del protocolo HTTP.
  - Se basa en el protocolo SPDY de Google, que es un 64% más rápido.
  - Su especificación se publicó en mayo de 2015.
  - Es soportado por la mayoría de navegadores.

MEJORAS RESPECTO A HTTP

- **Multiplexación** de peticiones HTTP sobre la misma conexión TCP.
- **Compresión** de cabeceras.
- Soporte para **server push** (notificaciones).
- **Pipelining** de solicitud-respuesta.
- **Cifrado** de la información → uso obligatorio de SSL (seguridad de la capa de socket).

FTP: PROTOCOLO DE TRANSFERENCIA DE ARCHIVOS

- El FTP define la comunicación con un **servidor de archivos**.
- Usa 2 conexiones **TCP paralelas**:
  - Conexión de **control** → para enviar los **comandos** y recibir las **respuestas**.
    - Es una conexión TCP **persistente**, dura todo el tiempo de la sesión FTP.
    - Usa ASCII DE 7 bits.
    - Usa por defecto el puerto 21.
  - Conexión de **datos** → para transmitir **datos** en respuesta a los comandos.
    - Es una conexión TCP **no persistente**, se abre y se cierra una nueva por cada fichero transferido.
    - Usa binario.
    - Usa por defecto el puerto 20.
- Es un protocolo **con estado** → mientras dura la sesión, el servidor FTP mantiene la información de estado (como el nombre de usuario, directorio actual, etc.).

MENSAJES FTP

- **Tipos** de mensajes FTP:
  - **Comandos** → tecleados por los usuarios.
    - Palabra de 4 caracteres en mayúscula.
    - Parámetros adicionales.
  - **Respuestas** → devueltas por el servidor.
    - Código de 3 dígitos.
    - Frase explicativa.

  ↳ Ambas líneas terminan en $cr + lf$.

USER	nombre de usuario
PASS	clave
LIST	
RETR	nombre de fichero (traer fichero)
STOR	nombre de fichero (almacenar fichero)

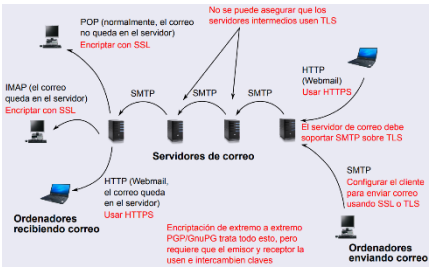
331	Username OK, password required
125	Data connection already open; transfer starting
425	Can't open data connection
452	Error writing file

CIFRADO DE TRANSMISIONES

Transmisión sin cifrar { FTP, telnet, HTTP } — Transmisión cifrada { SFTP, SSH, HTTPS }

Tipos de protocolos de correo:

- Para **enviar** correo al servidor o entre servidores → SMTP (o HTTP si se usa un navegador).
- Para **acceder** al correo por el AU → POP3, IMAP (o HTTP si se usa un navegador).



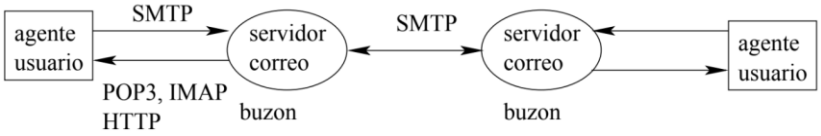
Funcionamiento del correo:

- El **buzón de correo** está en el servidor → los correos se pueden recibir en cualquier momento, incluso cuando el ordenador local no está disponible.

1. Cuando el usuario envía un correo, el AU se lo pasa al servidor. El ordenador local ya no tendrá que hacer nada más.
2. El servidor de correo origen examina el destino del correo y selecciona el servidor de correo destino apropiado.
3. El servidor de correo destino almacena el correo en el buzón de correo del destinatario.
   - Si el correo llegó bien → el AU destino usará alguno de los protocolos de acceso al correo para visualizarlo.
   - Si el servidor de destino está fuera de servicio → el servidor de origen guarda el mensaje en una cola para intentar enviarlo en unos minutos.
     - ↳ Si después de varios días no se ha conseguido enviar, el servidor se da por vencido y devuelve al remitente un mensaje de error.

SMTP: PROTOCOLO DE TRANSFERENCIA SIMPLE DE CORREO

- El SMTP define cómo se envía correo al **servidor de correo** o entre servidores de correo.
  - Usa una conexión **TCP persistente**, mantiene abierta la conexión mientras dura la sesión y por ella viajarán varios mensajes.
    - Usa por defecto el puerto 25.
  - Usa **ASCII de 7** bits tanto para las cabeceras como para los cuerpos.
    - ↳ El AU usa las extensiones MIME para transformar, por ejemplo, imágenes a ASCII.
  - Es un protocolo **con estado** → un comando se atiende teniendo en cuenta los anteriores.
  - Es un protocolo completamente **inseguro** → el servidor permite enviar correos a cualquiera sin pedir identificación.



MENSAJES SMTP

- **Tipos** de mensajes SMTP:
  - **Comandos:**
    - Palabra en mayúscula.
    - Parámetros adicionales.
  - **Respuestas**:
    - Código de 3 dígitos.
    - Frase explicativa.
  - ↳ Ambas líneas terminan en $cr + lf$.
  - **Datos** → contenido de los correos.
    - ↳ Todos los objetos que formen parte de un mensaje van encapsulados en **un único fichero**.

HELO	nombre servidor
MAIL FROM:	dirección de correo del remitente
RCPT TO:	dirección de correo del destinatario
DATA	(el contenido del correo)
QUIT	

220	nombre del servidor
250	comando que se ejecutó satistactoriamente
354	envíe el correo, terminando en .
221	cierre de conexión

```
S:  220 smtp.usc.es SMTP USC Server
C:  HELO smtp.usc.es
S:  250 smtp.usc.es
C:  MAIL FROM: pepito@usc.es
S:  250 Ok
C:  RCPT TO: juanito@yahoo.es
S:  250 Ok
C:  DATA
S:  354 Enter data, end data with <CR><LF>.<CR><LF>
C:  Hola
C:  Esto es una prueba, adios
C:  .
S:  250 Ok: message queued as C5F0B13210D
C:  quit
S:  221 Bye
```

COMPARACIÓN CON HTTP

HTTP	SMTP
Transmiten **archivos**	
Usan conexiones TCP **persistentes**	
Es un protocolo de **demanda** (el cliente demanda el archivo)	Es un protocolo de **oferta** (el cliente oferta el archivo)
El cuerpo es en **binario**	El cuerpo es en **ASCII de 7 bits**
Envía objetos diferentes en **archivos diferentes**	Envía objetos diferentes en el **mismo archivo**
No tiene **estado**	Tiene **estado** (debe recordar en qué parte de la sesión se encuentra)

POP3: PROTOCOLO DE OFICINA POSTAL, VERSIÓN 3

- El POP3 define cómo el AU transfiere el correo del **servidor destino** al **ordenador local**.
  - Usa una conexión **TCP persistente**, mantiene abierta la conexión mientras dura la sesión y por ella se descargarán varios correos.
    - Usa por defecto el puerto 110.
  - Es un protocolo **con estado**, pero **no** mantiene información de estado **entre sesiones**.
  - Es un protocolo más **seguro** → para usar el protocolo se necesita una cuenta con clave.

FASES

1. **Autorización** → autenticación del usuario.
2. **Transacción:**
   2.1. Recuperar los mensajes.
   2.2. Marcar/desmarcar para borrado.
   2.3. Estadísticas de correo.
3. **Actualización** → borrado de los mensajes marcados.
4. **Finalización** sesión POP3.

MENSAJES POP3

- **Tipos** de mensajes POP3:
  - **Comandos:**
    - Palabra de 4 caracteres.
    - Parámetros adicionales.
  - **Respuestas** → sólo hay dos tipos, $+OK$ y $-ERR$.
    - Ambas líneas terminan en $cr + lf$.
  - **Datos** → datos de la respuesta (lista de mensajes, contenidos del correo, etc.).
    - Se envía en un único mensaje la cabecera y el cuerpo del correo (en SMTP cada campo de la cabecera era su propio mensaje).

user	nombre del usuario
pass	palabra clave
list	
retr	número de correo (traer correo)
dele	número de correo (borrar correo)
quit	

+OK frase explicativa
−ERR frase explicativa

S:	+OK POP3 pop.usc.es server ready
C:	user pepito
S:	+OK User name accepted, password please
C:	pass miclave
S:	+OK Mailbox open, 2 messages
C:	list
S:	+OK Mailbox scan listing follows
S:	1 10064
S:	2 1054
S:	.
C:	retr 1
S:	(texto ASCII del correo)
S:	.
C:	dele 1
S:	+OK
C:	quit
S:	+OK POP3 server bye

INCONVENIENTES

- × No permite el acceso al correo en **múltiples dispositivos**, cuando se descarga un correo desaparece del servidor.
- × No permite **carpetas** en el servidor.

IMAP: PROTOCOLO DE ACCESO A MENSAJES DE INTERNET

- Tiene **más funcionalidades** que POP3, por lo que es **más complejo** → hay comandos para realizas búsquedas, manejar carpetas, descargar partes de un mensaje, etc.
- ✓ Permite el acceso a correos desde **múltiples dispositivos**, aunque se descargue un correo permanece en el servidor.
- ✓ El servidor asocia cada mensaje a una **carpeta**.
  - Cuando llega un mensaje va a la carpeta $inbox$.
- Es un proceso **con estado**, y mantiene información de estado **entre sesiones**.

DNS: SERVICIO DE NOMBRES DE DOMINIO

- El DNS se usa para **traducir** nombres de host a direcciones IP y viceversa. Consta de los siguientes **elementos**:
  - Numerosos **servidores de nombres** distribuidos por Internet.
  - Una **base de datos** que está distribuida de forma jerárquica entre el gran número de servidores de nombres.
  - Un **protocolo** que permite { a los hosts pedir traducciones a los servidores. / a los servidores intercambiar datos entre ellos.
- Es un protocolo **sin conexión**, usa **UDP**.
  - Cada host tiene una lista de direcciones de servidores de nombres (se suele introducir cuando se instala el SO).
    Cuando el host necesita una traducción, se dirige al primer servidor de la lista.
    Si por alguna razón no llega la respuesta, pasa al siguiente y así sucesivamente.
    Por esto se usa UDP en lugar de TCP → si la respuesta no llega se prefiere probar con el siguiente servidor antes que reintentarlo con ese.
  - Usa por defecto el puerto 53.
- Es un protocolo **sin estado**.

SERVICIOS PROPORCIONADOS

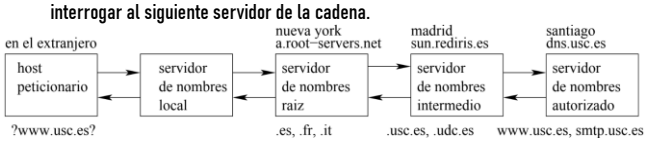
- **Traducciones** de nombres de hosts a direcciones IP y viceversa.
- Informa qué **servidor autorizado** es el encargado de proporcionar información sobre un determinado dominio.
- Obtención de **aliases** de **hosts**.
- Proporciona **aliases** de **servidores de correo** para simplificar las direcciones de correo.
- **Distribución de la carga** → en el DNS, a un mismo nombre de host se le pueden asignar varias direcciones IP distintas.
  - Se suelen usar en **servidores espejo** → servidores con el mismo contenido y que se usan de forma indistinta.
  - El DNS devuelve de forma cíclica cada una de las IPs del conjunto.

TIPOS DE SERVIDORES DE NOMBRES

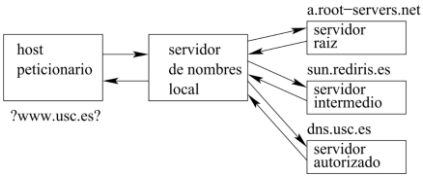
- **Servidores locales** → atienden a las **consultas** de los hosts.
  - Cada host tiene una lista de estos servidores (proporcionada por el ISP e introducida cuando se instala el SO). Cuando el host necesita una consulta, empieza preguntando al primer servidor de su lista.
- **Servidores autorizados** → los nombres de **hosts** almacenados en ellos son accesibles desde el internet.
  - Para que un host sea accesible en internet debe estar incluido al menos dos de estos servidores (por fiabilidad).
  - Muchos de ellos se comportan como servidores locales.
  - Suelen pertenecer al ISP.
- **Servidores raíz** → proporcionan información sobre los TLD (dominios de nivel superior) (es, uk, fr, com, org).
  - Hay sobre 400 en internet, gestionados por 13 organizaciones.
- **Servidores intermedios** → proporcionan información sobre los dominios de nivel intermedio ($usc.es,\ udc.es, elmundo.es$).

TIPOS DE CONSULTAS

- **Recursivas** → se le indica a cada servidor de nombres actual que se encargue de interrogar al siguiente servidor de la cadena.



- **Iterativas** → el servidor de nombres local es el encargado de ponerse en contacto con todos los demás servidores.



CACHÉ DNS

- La CACHÉ DNS consiste en que, cuando un servidor DNS obtiene una correspondencia, además de servirla, haga una **copia** para ella en su **memoria local**.
  - Esta copia se usará cuando se vuelva a hacer la misma consulta.
  - Las copias se borrarán después de unos días sin usarse.
- Hay cachés DNS en **todos los servidores** de la jerarquía, incluso en los locales.

MENSAJES DNS

- **Tipos** de mensajes FTP:
  - **Consultas.**
  - **Respuestas.**
- Ambos tipos de mensaje tienen el mismo formato, compuesto por una cabecera y un cuerpo:
  - Cabecera → contiene la información de **control**. Consta de 6 campos:
    - **Identificación** → número de 16 bits que identifica la consulta y su respuesta (ya que un host puede hacer varias consultas a la vez).
    - **Señales** → 4 bits que indican si el mensaje es una consulta o respuesta, si la respuesta la dio un servidor autorizado, el tipo de consulta, etc.
    - **Tamaño de los campos del cuerpo** (4 campos) → cuántas cuestiones, cuántas respuestas, cuántos servidores autorizados y cuánta información adicional hay.
  - Cuerpo → consta de 4 campos:
    - **Cuestiones** → una o varias preguntas (nombre de host o dirección IP a traducir, dirección de email a convertir, etc.).
    - **Respuestas** → una o varias respuestas obtenidas (pareja nombre host/dirección IP, dirección email convertida, etc.).
      - Cada cuestión puede originar varias respuestas (por ejemplo, un nombre de host puede corresponder a varias IPs).
    - **Servidores autorizados** → servidores de nombres autorizados a los que se puede consultar para un determinado dominio.
      - Esto permite hacer una cadena de consultas.
    - **Información adicional**.
    - Todos los campos del cuerpo tienen el mismo formato → una 4-tupla denominada REGISTRO DE RECURSO formada por (tipo, nombre, valor y TTL).
      - TTL (time to live) → tiempo durante el cual es válida la respuesta.

Tipo	Nombre	Valor	TTL
A	nombre de host	dirección IP	
NS	dominio	servidor autorizado para el dominio	
CNAME	alias	nombre de host	
MX	alias de correo	servidor de correo	

cabecera	identificacion	senales
	num. cuestiones	num. respuestas
	num. s. autorizados	num. inf. adicional
cuerpo	cuestiones	
	respuestas	
	servidores autorizados	
	informacion adicional	

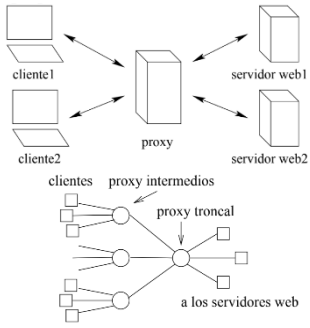
Ejemplo:

1	1
2	2

www.usc.es?
www.usc.es –> 193.144.74.224
usc.es –> dns.usc.es, dns2.usc.es
direcciones IP de los servidores autorizados

DISTRIBUCIÓN DE CONTENIDOS

- El acceso a servidores centralizados puede ser lento debido a que { el camino de los mensajes es lento o está congestionado, o / el servidor está sobrecargado.
- **Solución** → en lugar de tener todos los contenidos en un mismo servidor, se distribuyen (y duplican) en distintas zonas y se dirigen las peticiones al servidor con menos tiempo de respuesta.
  Métodos de distribución de contenido:
  - **Caché web** → infraestructura proporcionada por el ISP.
  - **CDN** → infraestructura proporcionada por el distribuidor de contenidos.
  - **Redes P2P** (de igual a igual) → infraestructura proporcionada por el usuario.



CACHÉ WEB

- La CACHÉ WEB consiste en poner un servidor intermedio, el **servidor proxy,** por el cual pasarán todas las peticiones de los hosts de una red.
  - ↳ El servidor proxy suele ser proporcionado por el ISP.
- Los servidores proxys almacenan durante un cierto tiempo una **copia del contenido** que les devuelven los servidores destino.
  - ↳ Las siguientes peticiones de ese contenido se resolverán en el proxy, sin tener que acceder al servidor destino.
- El usuario debe **configurar su navegador** para poder usar el servidor proxy.
- Los servidores proxy admiten un **sistema jerárquico**.

CDN – Redes de distribución de contenidos

- Las CDN son empresas que poseen **centros de host de Internet** y alquilan su infraestructura a otras empresas.
  - Los CENTROS DE HOST DE INTERNET son conjuntos de cientos de servidores distribuidos a lo largo del mundo y conectados a Internet por enlaces de alta velocidad.
- La empresa CDN replica los contenidos de sus clientes en los servidores CDN y los mantiene actualizados.
- La empresa CDN proporciona un mecanismo para que el contenido sea entregado por el servidor CDN que lo pueda hacer más rápidamente.
- El **acceso** al contenido almacenado en estos centros de hosts se puede hacer de dos formas:
  - **Redirección de objetos** → los documentos base estarán en el sitio web del cliente.
    - ↳ Serán unos documentos base mínimos, conteniendo prácticamente sólo una redirección a los objetos almacenados en los servidores del centro de host.
  - **Balanceo de las peticiones usando el DNS** → permite seleccionar el servidor CDN más cercano o rápido al hacer la traducción del nombre de host a una dirección IP.

REDES P2P

- En una RED P2P (peer to peer) todos los usuarios son a la vez servidores (comparten los ficheros almacenados en su disco duro) y clientes de los contenidos (descargan ficheros).
- Un usuario se une a la red P2P ejecutando una aplicación, que buscará a otros usuarios. Se necesita por lo menos un **nodo de arranque**, es decir, conocer por lo menos un usuario que ya esté en la red.
- Cuando el usuario se conecta a la red, da a conocer públicamente el contenido de su **disco duro**.
- La red necesita construir un **directorio** con todos los contenidos y su ubicación. El directorio puede ser:
  - Directorio **centralizado** (*napster*) → un único host centraliza la construcción y el almacenamiento del directorio.
  - Directorio **no centralizado** (*kaaza*) → el directorio se distribuye entre los denominados usuarios líderes (aquellos con alta velocidad de conexión, antigüedad, etc.).
  - **Inundación de consultas** (*gnutella*) → no se usa un directorio, si no que las consultas se retransmiten de usuario a usuario hasta encontrar el contenido solicitado.

DESCARGA DE FICHEROS (bittorent)

- En una aplicación P2P **clásica**, una vez que un usuario descarga un fichero se convierte en servidor de ese fichero.
- En *bittorent* el fichero se descarga por **segmentos**, de manera que cuando un usuario descarga un segmento se convierte en servidor para él.
- Cada archivo se comparte en **su propia red P2P** en la que los usuarios se transmiten los segmentos del archivo entre sí.
- Cada red P2P tiene un nodo de infraestructura denominado **tracker**, el cual conoce qué sistemas están participando en la red en un momento dado.
  - ↳ Cuando un nuevo usuario se une a la red, se registra mediante el tracker. Periódicamente, el tracker comprobará si el usuario sigue en la red.
- **No** se necesita construir un **directorio** pues la información de cada archivo está en los $.torrent$.

CARACTERÍSTICAS Y APLICACIONES DE LAS REDES P2P

- Las arquitecturas P2P son **autoescalables**.
- **DHTs** (tablas has distribuidas) → bases de datos simples en las que los registros se distribuyen entre los usuarios de una red P2P.
- Telefonía sobre internet (Skype).

FLUJOS DE VÍDEO Y CDN

- Vídeo por internet.
- Flujo de vídeo HTTP y DASH (tecnología de flujos dinámicos adaptativos sobre HTTP).
- Redes de distribución de contenidos (CDN).

▷ Casos de estudio: Netflix, YouTube y Kankan.