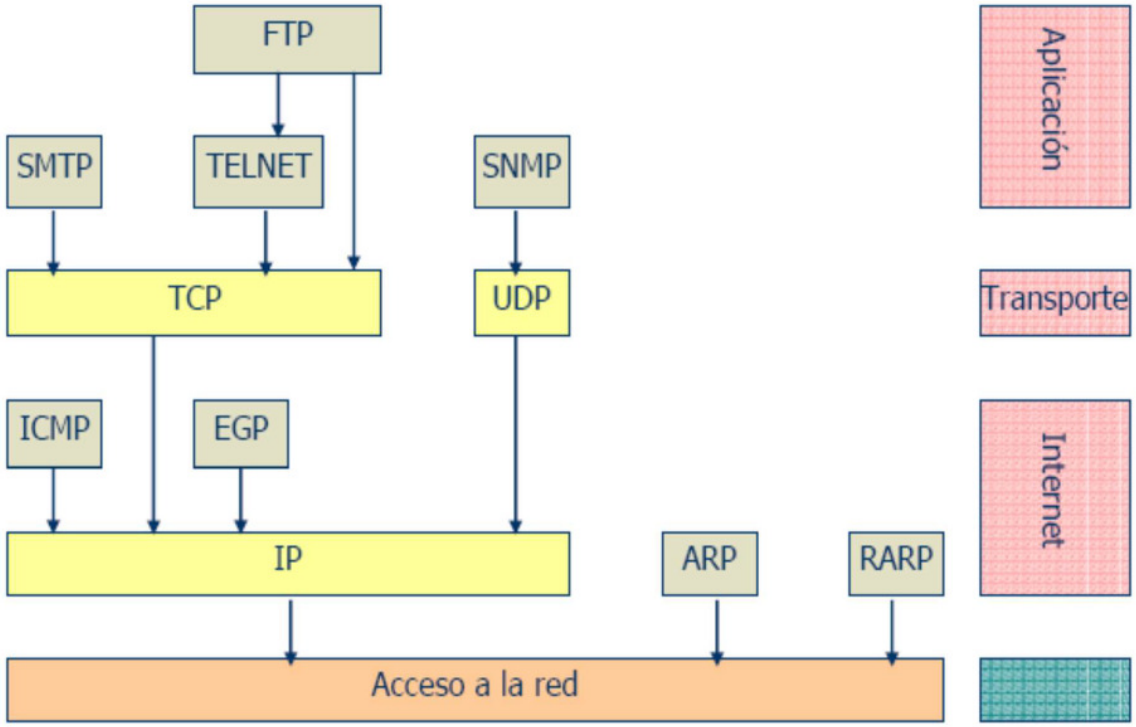
Capa de aplicación

Introducción a las aplicaciones de red

Esquema de protocolos TCP/IP



Arquitectura de la aplicación de red

Establece cómo debe estructurarse la aplicación en los distintos sistemas terminales.
Hay dos predominantes: Cliente-Servidor y P2P.

Cliente-Servidor

Hay un host siempre en funcionamiento como servidor, que da servicio a las solicitudes de otros host (clientes) que funcionan de forma intermitente. Los servidores tienen una dirección IP fija y pública (conocida) y suelen agruparse en granjas o clusters (centro de datos).
Los clientes pueden comunicarse con el servidor, pero no entre ellos y suelen tener una dirección IP dinámica y privada. El cliente es quien inicia la comunicación, mientras que el servidor espera a ser contactado. Un proceso envía/recibe mensajes a través de una interfaz software llamada socket. Para recibir mensajes, un proceso debe tener un identificador (IP + puerto).
Un socket es un descriptor de una transmisión a través del cual una aplicación puede enviar/recibir información a través de otro proceso de aplicación (es una puerta de acceso entre la aplicación y los servicios de transporte). En la práctica es un puntero a una estructura.

P2P

Hay pocos o ningún servidor siempre activo, la comunicación se realiza entre parejas de hosts conectados intermitentemente.

Retardo en cola

Para estimarlo se usa la teoría de colas:

El retardo en cola es: $$R = \frac{\lambda \cdot (T_s)^2}{1 - \lambda \cdot T_s}$$

donde Ts (distribución Exponencial) es el valor esperado del tiempo de servicio y
λ (distribución de Poisson) es el valor esperado de la tasa de llegada de solicitudes.

Protocolo de aplicación

Define cómo se pasan los mensajes entre los procesos de una aplicación, que se ejecutan en distintos hosts. En particular define:

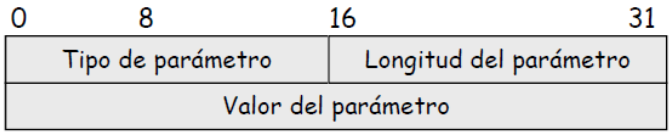
- Tipo de servicio: SNOC/SOC, realimentado o no, …
- Tipo de mensaje: (request, response, …)
- Sintaxis de los mensajes: definición y estructura de campos. Generalmente orientado a texto (HTTP), aunque hay excepciones (DNS). La tendencia es usar el formato Type-Length-Value (hay información que puede tener presencia opcional y longitud variable).
- Semántica o significado de los campos.
- Reglas para determinar cuándo los procesos envían/reciben mensajes.

Los protocolos son componentes de las aplicaciones de red.

Tipos de protocolos:

- De **dominio público** (definidos en RFC, abiertos. Ej: HTTP, SMTP) vs de **dominio propietario** (privados. Ej: Skype, IGRP).
- **In-band** (la información de señalización y los datos van en el mismo canal. Ej: teléfono) vs **out-of-band** (van por distintos canales. Ej: FTP).
- **Stateful** (exige un estado común entre las entidades, el mensaje depende de la historia. Ej: web) vs **stateless** (cada solicitud es independiente, no tiene nada que ver con las anteriores. Ej: IP).
- **Persistente** (al solicitar más de una conexión solo habrá una abierta persistentemente) vs **no persistente** (se realizan múltiples conexiones). (sobre servicios SOC).

La tendencia es hacer los protocolos flexibles con una cabecera fija y una serie de “trozos” (obligatorios y opcionales). dichos trozos pueden incluir una cabecera específica más una serie de datos en forma de parámetros fijos (en orden) u opcionales o de longitud variable (formato TLV (Type-Length-Value)). Ej: 0 ↔ 16 b tipo de parámetro, 16 ↔ 31 b longitud del parámetro, 0↔ 31 b valor del parámetro.



Características/requisitos de las aplicaciones de red

- **Tolerancia a pérdidas de datos (errores)**: Algunas aplicaciones pueden tolerar algunas pérdidas de datos (aplicaciones de audio/vídeo). Sin embargo otras necesitan una transferencia 100% fiable, es decir, que al poner los datos en el socket saben con certeza que llegarán sin errores al destino (aplicaciones financieras o correo).
- **Exigencia de requisitos temporales:** Algunas aplicaciones (denominadas inelásticas, Ej: telefonía, juegos interactivos) requieren retardo (latencia o delay) acotado para ser efectivas, mientras que en otras no importa (correo o transferencia de archivos).
- **Demanda de ancho de banda (tasa de transmisión o throughput)**: Las aplicaciones sensibles al ancho de banda requieren envío de datos a una tasa determinada (Ej: códec de vídeo), mientras que otras pueden funcionar con la disponible, sea mucha o poca (correo o transferencia de archivos).
- **Nivel de seguridad**: Los requisitos de seguridad para las distintas apps son muy variables y numerosos. Algunos ejemplos son encriptación, autenticación, integridad o no repudio.

Protocolos de transporte

TCP

Es un servicio orientado a conexión (SOC) y de transferencia de datos fiable. En cuanto a lo primero, TCP hace que el cliente y el servidor intercambien información de control de la capa de transporte antes de que fluyan los mensajes de la capa de aplicación. Este proceso de negociación, reconocimiento o de establecimiento de la conexión abierta a cliente y servidor alerta a cliente y servidor para que se preparen para el intercambio de paquetes. Después de esto, hay una conexión TCP entre los sockets (full-dúplex), y una vez que la aplicación termina de enviar mensajes, hay que desactivar dicha conexión. Respecto a lo segundo (transferencia de datos fiable), TCP se encarga de que cualquier flujo de bytes en el socket del emisor llegue sin pérdidas ni duplicación al del receptor.
TCP también incluye un mecanismo de control de congestión, que regula al emisor cuando la red está congestionada y limita cada conexión para que use una cuota equitativa de ancho de banda (lo que perjudica a aplicaciones de tiempo real, por lo que suelen usar UDP) y de flujo.

UDP

No está orientado a conexión (SNOC), por lo que no hay proceso de negociación previo a la comunicación. y no ofrece transferencia de datos fiable, por lo que no hay garantías de que el mensaje del emisor llegue al receptor y , si llega, puede hacerlo de forma desordenada. Tampoco incluye mecanismo de control de congestión, por lo que el emisor puede introducir datos en la capa de red a cualquier velocidad. Aunque las aplicaciones en tiempo real a veces van con UDP, cada vez más se montan sobre TCP porque muchos cortafuegos bloquean casi todo el tráfico UDP.

Tanto TCP como UDP (capa de transporte) son usuarios del protocolo IP de la capa de red, por lo que no garantizan Calidad de servicio (QoS), es decir:

- El retardo no está acotado
- Las fluctuaciones en el retardo no están acotadas
- No hay una velocidad de transmisión mínima garantizada
- No hay una probabilidad de pérdida acotadas

Ni tampoco tienen garantías de seguridad.

Protocolo DNS (Domain Name Service)

La comunicación en Internet precisa de direcciones IP (magnitud de 32 bits que identifica inequívocamente un host), sin embargo, es más cómodo usar nombres de dominio. Los routers usan las direcciones, por lo que hay que "traducir". Para esto sirve el DNS, que se encarga de la resolución de nombres o traducción de nombres a direcciones IP.
Los dominios siguen una estructura jerárquica: lo que hay antes del primer punto es la parte local, después los dominios desde el nivel n al 1 (este último es el dominio genérico).

Parte_local.dominio_niveln. … .dominio_nivel2.dominio_nivel1.

A los dominios de nivel1 se les denomina **Top Level Domians (TLD)** (.com .es .edu etc).

El dominio raíz "." lo gestiona el ICANN (Internet Corporation for Assigned Names and Numbers).

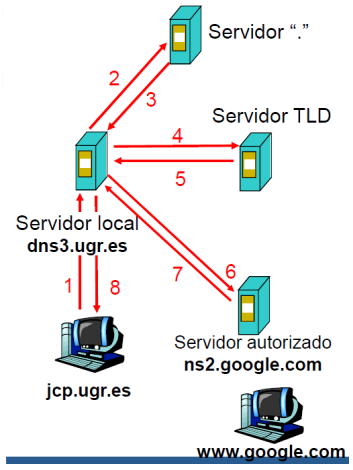
Inicialmente se definen 9 dominios genéricos (TLD):

- .com → Organizaciones comerciales
- .edu → Instituciones educativas de EEUU
- .gov → Instituciones gubernamentales de EEUU
- .mil → grupos militares de EEUU
- .net → proveedores de Internet
- .org → Organizaciones distintas de las anteriores
- .arpa → Propósitos exclusivos de infraestructura de Internet
- .int → Organismos establecidos por tratados internacionales entre gobiernos
- .xy → Indicativos de la zona geográfica (ej: .es → españa, .jp → japón, …)

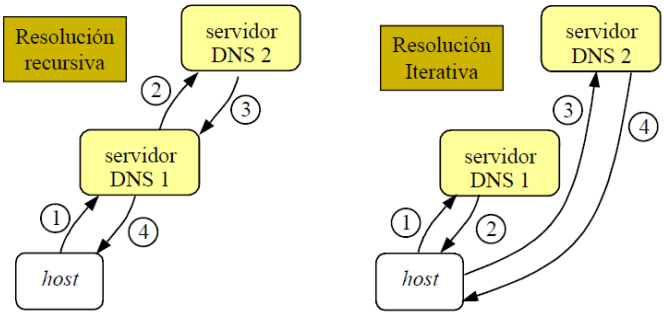
DNS es un protocolo de aplicación para acceso a una base de datos distribuida con una gestión distribuida. Hay 3 niveles de servidores DNS:

- **Servidores DNS raíz "."** : En Internet ha 13 servidores DNS raíz, de la A a la M. El F tiene un servidor en Madrid (Espanix: punto neutro).
- **Servidores de dominio** (de nivel superior, TLD) : Responsables de los dominios de nivel superior (genéricos).
- **Servidores locales**

Funcionamiento



Primero se consulta el "resolver" local (caché), después al servidor local. Este consulta al servidor raíz, que le devuelve una listas de las direcciones IP de los servidores TLD que son responsables del dominio buscado. El local reenvía el mensaje de consulta a uno de esos servidores y este responde con la IP del servidor autorizado correspondiente. Por último, el servidor local reenvía la consulta ahí y recibe la dirección IP del dominio que buscaba.
Puede haber consultas recursivas (obtener por sí mismo la respuesta o dirección completa) e iterativas (el cliente pregunta a varios que le dan trozos de dirección).
Normalmente, la consulta del host al server local es recursiva, mientras que las de este son iterativas



Conceptos y gestión de la base de datos DNS (distribuida y jerárquica)

Está formada por un conjunto de servidores cooperativos que almacenan parcialmente la base de datos que se denomina BIND (Berkeley Internet Name Domain). Cada servidor es responsable de una zona (conjunto de nombres de dominio contiguos (por debajo de uno dado en el árbol) de los que un servidor tiene toda la información y es su autoridad). Los servidores de autoridad (Start of Authority Servers) deben contener toda la información de su zona y la autoridad puede delegarse jerárquicamente a otros servidores.
Cada zona debe tener, al menos, un servidor de autoridad. En cada zona hay servidores primarios (copia master de la base de datos) y secundarios (obtienen la base de datos por

transferencia). Existe un servicio caché para mejorar prestaciones. Cuando un cliente, a través de un resolver local, solicita una resolución de nombres a su servidor puede pasar:

- **Respuesta con autoridad**: El servidor tiene autoridad sobre la zona del nombre solicitado y devuelve su dirección IP.
- **Respuesta sin autoridad**: No tiene autoridad sobre la zona en la que se encuentra el nombre solicitado, pero lo tiene en caché.
- **No conoce la respuesta**: el servidor preguntará a otros servidores de forma recursiva o iterativa. Normalmente se "eleva" la petición a uno de los servidores raíz.

En la base de datos DNS todo dominio está asociado, al menos, a un registro RR (Resource Record), y cada registro RR es una tupla con 5 campos:

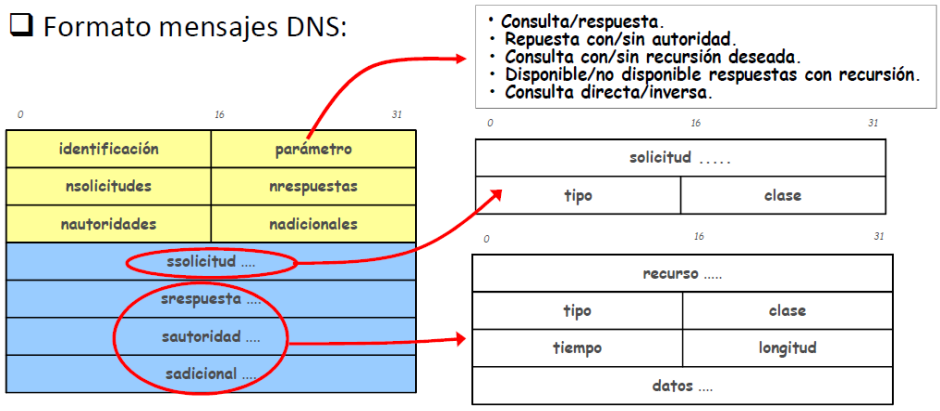
- **Nombre del dominio**
- **Tiempo de vida**: tiempo de validez de un registro en la caché
- **Clase**: En internet siempre es IN
- **Tipo** (tipo de registro):
    - SOA. → Registro (Start of Authority) con la autoridad de la zona
    - NS. → Registro que contiene un servidor de nombres
    - A. → Registro que define una dirección IPv4
    - MX. → Registro que define un servidor de correo electrónico
    - CNAME. →Registro que define el nombre canónico de un nombre de dominio
    - HINFO. → Información del tipo de máquina y sistema operativo.
    - TXT. → Información del dominio
- **Valor**: Su contenido depende del campo tipo

Existe una base de datos asociada de resolución inversa, para traducir direcciones IP a nombres de dominio (in-addr.arpa).

Formato de los mensajes DNS

Los mensajes DNS solo pueden ser de consulta o de respuesta y ambos usan el mismo formato:

- Los primeros 12 B son la cabecera, que contiene varios campos:
    - número 16 bits que identifica la consulta (copiado de la respuesta). En parámetros o indicadores se pone:
        - 1 bit para indicar si el mensaje es de consulta (0) o respuesta (1)
        - 1 bit autoritativo para saber en el mensaje de respuesta cuando un servidor DNS es autorizado para el nombre solicitado
        - 1 bit para indicar que se desea que el servidor haga una búsqueda recursiva cuando no tenga el registro
        - 1 bit para indicar en un mensaje de respuesta que el servidor tiene la recursión disponible
    - 4 campos "número de" que indican el nº de solicitudes, respuestas, autoridades y adicionales:
        - SSolicitud: contiene el nombre que se va a consultar, el tipo y la clase
        - SRespuesta: Contiene los registros RR para el nombre consultado
        - SAutoridad: Contiene registros de servidores autorizados
        - SAdicional: Contiene otros registros



DNS se ofrece en el puerto 53 mediante UDP, normalmente, o TCP.

La navegación Web (protocolo HTTP)

Una página web está formada por objetos (archivos como una imagen, un vídeo o un archivo HTML que pueden direccionarse con un único URL). Normalmente, las páginas web tienen un archivo base HTML y objetos referenciados (el archivo HTML hace referencia a los otros objetos mediante la URL de estos). Las páginas se sirven con el protocolo HTTP (HyperText Transfer Protocol), el cual se implementa en 2 programas, uno cliente (navegador que solicita, recibe y muestra objetos web) y otro servidor (que envía objetos web en respuesta a peticiones).

Características del protocolo HTTP

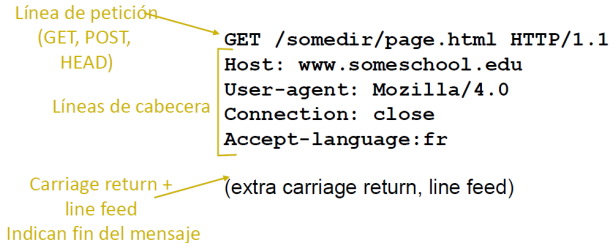
- Usa los servicios de TCP (SOC) en el puerto 80. El cliente HTTP inicia una conexión TCP con el servidor y cuando esta se ha establecido, los procesos navegador y servidor acceden a TCP a través de su interfaces socket. El cliente envía mensajes a su interfaz socket y estos pasan a TCP, que se encarga de enviarlos correctamente (y recibirlos). (inicio de conexión TCP, envío TCP, cierre de conexión TCP).
- Es stateless, ya que no mantiene información sobre las peticiones de los clientes. Para esto están las Cookies.
- Es un protocolo in-band, orientado a texto.
- Hay 2 tipos de servidores:
    - No persistentes (en cada conexión TCP se envía solo 1 objeto). HTTP/v1.0
    - Persistentes (sobre una misma conexión TCP se pueden enviar múltiples objetos). Ej: HTTP/v1.1

Mensajes HTTP

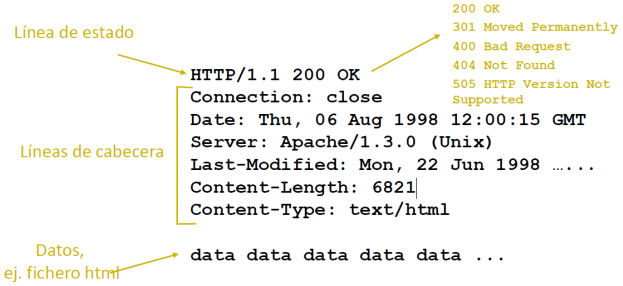
1) El cliente HTTP inicia conexión TCP al servidor HTTP en el puerto 80
2) El servidor HTTP acepta la conexión y solicita al cliente abrirla (SYNC + ACK)
3) El cliente HTTP confirma con un ACK
4) El cliente HTTP envía un request message al servidor a través de su socket
5) El servidor HTTP devuelve la respuesta.
6a) Servidor no persistente → El proceso servidor pide a TCP que cierre la conexión, pero este no lo hace hasta que el cliente no recibe la respuesta.
6b) Servidor persistente → Se envían más objetos pr la misma conexión y después se cierra
7) Nuevas conexiones.

HTTP define dos tipos de mensaje:

- **HTTP request message** (del cliente al servidor):
  - Línea de petición con 3 campos: el método (GET, POST, HEAD), el campo URL y el campo versión de HTTP.
  - Líneas de cabecera: Host (donde reside el objeto), User-agent (tipo y versión del navegador que hace la solicitud), Connection (close indica que las conexiones no son persistentes), Accept-language (idioma).
  - Retorno de carro y salto de línea que indican el fin del mensaje



- **HTTP response message** (respuesta del servidor al cliente):
  - Línea de estado: contiene 3 campos → versión del protocolo, código de estado y descripción de estado (200 = OK, 301 = Moved Permanently, 400 = Bad Request, 404 = Not Found, 505 = HTTP Version Not Supported).
  - Líneas de cabecera: Connection (close es que el cliente va a cerrar la conexión después del mensaje), Date (fecha y hora de la creación y envío de las respuesta por el servidor), Server (indica tipo de servidor), Last-Modified (hora y fecha de la última modificación), Content-Length(bytes del objetivo enviado) y Content-Type (indica tipo de objeto)
  - Cuerpo de entidad: contiene el objeto solicitado (Ej.: fichero html).



Métodos (acciones solicitadas por los clientes en los request messages)

- OPTIONS: Solicitud de información sobre las opciones disponibles.
- GET: Solicitud de un recurso (puede ser condicional).
- HEAD: Igual que GET, pero el servidor solo devuelve cabeceras.
- POST: Solicitud al servidor para que acepte y subordine a la URI (Uniform Resource Identifier, que identifica recursos de una red de forma unívoca. Se diferencian de las URL en que estas hacen referencia a recursos que pueden variar en el tiempo) los datos incluidos en la solicitud.
- PUT: Solicitud de sustituir la URI especificada con los datos incluidos en la solicitud.
- DELETE: Solicitud de borrar la URI especificada.

Códigos de respuesta (para los response messages)

- 1xx → Indican mensajes exclusivamente informativos.
- 2xx → Indican algún tipo de éxito.
- 3xx → Redirección al cliente a otra URL.
- 4xx → Error.
- 5xx → Error.

Cabeceras (47 request headers y 49 response headers)

En **peticiones y respuestas** se pueden poner las cabeceras Content-Type, Content-Length, Content-Encoding (formato de codificación de los datos enviados), Date (deben incluir la zona horaria).

Cabeceras solo para peticiones del cliente:

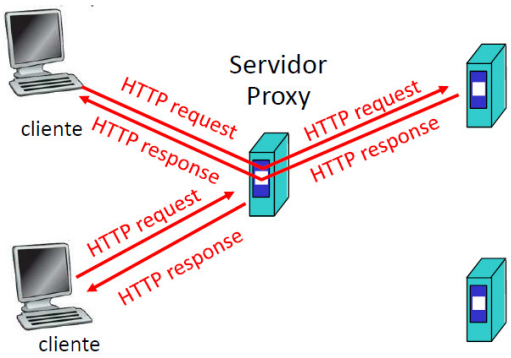
- Accept: campo opcional que contiene una lista de tipos MIME (Multipurpose Internet Mail Extensions. Son especificaciones dirigidas al intercambio de archivos por Internet) que acepta el cliente.
- Authorization: clave de acceso que envía un cliente para acceder a un recurso protegido o limitado. Incluye el formato de autorización y la clave.
- From: campo opcional con la dirección de correo electrónico del usuario del cliente web que realiza el acceso.
- If-Modified-Since: para operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la dada. Equivale a un HEAD + un GET; lo usan sistemas de almacenamiento temporal de páginas.
- Referer: contiene la URL del documento desde donde se ha activado este enlace. Un servidor puede informar al creador del documento de cambios en los enlaces que contiene. No todos los clientes lo envían.
- User-Agent: cadena que identifica el tipo y versión del cliente que realiza la petición.

Cabeceras solo para respuestas del servidor:

- Allow: informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere la respuesta.
- Expires: fecha de expiración del objeto enviado. Las cachés deben descartar las copias posteriores. No todos los sistemas lo envían.
- Last-Modified: fecha local de modificación del objeto devuelto.

Web caché

El objetivo de la caché web (o servidor proxy) es satisfacer solicitudes HTTP del cliente sin involucrar al servidor destino. El usuario puede configurar el navegador para que sus solicitudes HTTP vayan a la caché web. Si el objeto está en caché, esta lo devuelve (actúa como servidor), si no, lo pide al servidor (actúa como cliente) y después lo envía al cliente.
Las cachés pueden reducir el tiempo de respuesta a las solicitudes y reducir el tráfico web.
Si la caché tiene los datos actualizados al enviar un mensaje de solicitud al servidor (con if-modified-since), este no le devuelve nada en el mensaje de respuesta. Si no, le responde con los datos.



Cookies

Son pequeños ficheros de texto que se intercambian los clientes y servidores HTTP para solucionar la falta de información de estado entre dos transacciones (protocolo stateless). Fueron introducidas por Netscape y ahora se han estandarizado. La primera vez que un usuario accede a un documento de un servidor, este proporciona una cookie con datos que relacionarán posteriores operaciones. El cliente almacena la cookie en su sistema para usarla después, y los futuros accesos al servidor, el navegador podrá proporcionar la cookie original, que servirá de enlace entre ese acceso y los anteriores. Todo esto se hace automáticamente (sin intervención del usuario). Las cookies son líneas de texto con pares variable/valor, y hay un conjunto de nombres de variable necesarios para el correcto funcionamiento de estas:

- Domain= → Conjunto de direcciones para las que es válida la cookie
- Path= → Subconjunto de URL para las que sirve la cookie
- Version= → Permite seleccionar entre diferentes versiones del modelo de cookies
- Expires= → Cuándo expira la información. Si no se incluye, los datos se descartan al finalizar la sesión con el cliente web

Al recibir una petición, el servidor HTTP envía los diferentes campos de una cookie con la cabecera HTTP set-Cookie. Cuando se accede a una URL que verifica dominio/path registrados, el cliente envía automáticamente la información de los campos de la cookie con la nueva cabecera HTTP cookie.

El Correo electrónico (SMTP/IMAP/POP3)

El correo electrónico tiene 4 componentes:

- **Agente de usuario o cliente de correo** (MUA (Mail User Agent)): Es quien compone, edita y lee mensajes del buzón. Por ejemplo, Outlook o Thunderbird.
- **Servidor de correo** (MTA (Mail Transfer Agent o Mail Server)): Reenvía mensajes salientes y almacena mensajes entrantes de cada usuario.
- **Protocolo de envío SMTP** (Simple Mail Transfer Protocol).
- **Protocolos de descarga** o lectura: POP3, IMAP, HTTP.

SMTP

Transfiere mensajes desde los servidores de correo de los emisores a los de los destinatarios. Se implementa mediante dos programas (ambos incluidos en los servidores de correo):

- Cliente SMTP: se ejecuta en el MTA que envía correo.
- Servidor SMTP: se ejecuta en el MTA que recibe.

SMTP usa TCP en el puerto 25 y es un protocolo orientado a texto.
Es SOC, in-band y statefull e implica 3 fases: handshaking, transferencia de mensajes y cierre. La interacción entre cliente SMTP y servidor SMTP se realiza mediante comando/respuesta (comando = texto ASCII, respuesta = código de estado y frases explicativas). Los mensajes deben estar codificados en ASCII de 7 bits, para mejorar esto se usan las extensiones MIME

Pasos en envío recepción de correo

1. El usuario origen compone mediante su MUA un mensaje dirigido a la dirección de correo del usuario destino.
2. Se envía con SMTP o HTTP el mensaje al MTA del usuario origen, que lo sitúa en la cola de mensajes salientes
3. El cliente SMTP abre una conexión TCP con el MTA (obtenido por DNS) del usuario destino
4. Después de la fase de negociación inicial de SMTP, el cliente SMTP envía el mensaje sobre la conexión TCP.
5. El servidor de correo del destino recibe el mensaje y lo pone en el buzón del usuario
6. El usuario destino invoca su MUA para leer el mensaje con POP3, IMAP o HTTP

Comandos cliente SMTP

- **EHLO** (antes HELO) → Identifica el remitente al destinatario
- **MAIL FROM** → Identifica una transacción de correo y al emisor
- **RCPT TO** → Identifica un destinatario individual (para varios, se repite).
- **DATA** → Permite enviar líneas de texto, la última solo un punto y todas seguidas de retorno de carro y avance de línea.
- **RSET** → Aborta la transacción del correo actual
- **NOOP** → No operación. Indica al extremo que envíe una respuesta positiva.
- **QUIT** → Pide al otro extremo respuesta positiva y cierre de conexión
- **VRFX** → Pide al receptor que confirme que un nombre es un destinatario válido
- **EXPN** → Pide al receptor la confirmación de una lista de correo y que devuelva los nombres de usuario de la lista
- **HELP** → Pide información de los comandos disponibles
- **TURN** → El emisor pide actuar como receptor (este puede negarse)
- **SOML** → Si el destinatario está conectado, entrega el mensaje al terminal, si no, lo entrega como correo normal
- **SAML** → Entrega el mensaje al buzón y, si está conectado, también al terminal
- **SEND** → Si el destinatario está conectado, entrega el mensaje al terminal

Comandos de respuesta servidor SMTP

Los más comunes son:

- **211** → Estado del sistema
- **214** → Mensaje de ayuda
- **220** → Servicio preparado
- **221** → Servicio cerrando el canal de transmisión
- **250** → Solicitud completada con éxito
- **354** → Introduzca el texto finalizando con retorno de carro, avance de línea, punto, retorno de carro, avance de línea.
- **421** → Servicio no disponible
- **501** → Error de sintaxis. Por ejemplo, contestación de SMTP a ESMTP
- **503** → Secuencia de comandos errónea
- **551** → Usuario no local (no se tiene cuenta)
- **554** → Fallo en la transacción

Las extensiones MIME no cambian la arquitectura de correo anterior, pero soportan texto con caracteres distintos de US-ASCII, adjuntos que no son de tipo texto, cuerpos de mensaje con múltiples partes e información de encabezados con caracteres distintos de ASCII.
Las cabeceras de mensajes MIME tienen varios campos:

- **MIME-version** → versión de MIME. Si no esta se considera que el mensaje es texto normal en inglés
- **Content-Description** → Texto que describe el contenido, necesario para que el destinatario sepa si quiere descodificar y leer o no el mensaje
- **Content-Id** → Identificador único con el mismo formato que Message-Id
- **Content-Transfer-Encoding** → Indica cómo está envuelto el cuerpo del mensaje. Hay 5 tipos de codificación: ASCII 7, ASCII 8, codificación binaria, base64 y entrecomillada-imprimible.7.2.
- **Content-Type** → Naturaleza del cuerpo del mensaje. Hay varios tipos y subtipos. Text, Image, Audio, Video, Application, Message y Multipart.
    - El tipo application es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos.
        - El subtipo octet-stream es una secuencia de bytes no interpretados, tal que a su recepción, un MUA debería presentarla en pantalla sugiriendo que se copie en un archivo y pidiendo un nombre de archivo.
        - El subtipo postscript se refiere al lenguaje PostScript. Puede haber riesgos al visualizarlo.
    - El tipo message permite que un mensaje esté encapsulado dentro de otro, útil para reenviar correo.
        - El subtipo rfc822 se usa cuando se encapsula un mensaje RFC 822 en otro.
        - El subtipo partial permite dividir un mensaje encapsulado y enviarlo por trozos (los parámetros posibilitan juntarlos bien en el destino).
        - El subtipo external-body puede usarse para mensajes muy grandes. En vez de incluir el archivo se da una dirección FTP y el MUA del receptor lo obtiene a través de la red cuando quiera.
    - El tipo multipart permite que un mensaje tenga más de una parte, teniendo cada un inicio y un fin delimitados.
        - El subtipo mixed permite que cada parte sea diferente.
        - El subtipo alternative indica que cada parte tiene el mismo mensaje. con distinta codificación.
        - El subtipo parallel se usa cuando todas las partes deben verse a la vez.
        - El subtipo digest se usa cuando se juntan muchos mensajes en uno compuesto.

Protocolos de acceso a correo electrónico

POP3 (Post Office Protocol - Version 3)

Se inicia cuando el MUA abre una conexión TCP en el puerto 110 al MTA. Una vez establecida la conexión, POP3 pasa por 3 fases:

- **Autorización**: el MUA envía user = nombre de usuario y pass = contraseña y el servidor responde +OK o -ERR
- **Transacción**: el MUA recupera mensajes, los borra, … (los administra en general)
- **Actualización**: después de que el cliente haga quit y se termine la sesión POP3, el MTA borra los mensajes marcados para el borrado.

IMAP4 (Internet Message Access Protocol)

Las ventajas contra POP3 son:

- Permite organización en carpetas en el lado del MTA manteniendo información entre sesiones (asociando flags a los mensajes).
- Permite la descarga de partes de los mensajes.
- Es posible acceder con varios clientes (también POP, pero en modo descragar y guardar).

Web MAIL

El MUA es un navegador web corriente y el usuario se comunica con su buzón remoto a través de HTTP (uso extendido de HTTPS por seguridad).

Listado de puertos relacionados con e-mail

- POP3 → 110
- IMAP → 143
- SMTP → 25
- HTTP → 80
- Secure SMTP (SSMTP) → 465
- Secure IMAP (IMAP4-SSL) → 585
- IMAP4 over SSL (IMAPS) → 993
- Secure POP3 (SSL-POP) → 995

Aplicaciones multimedia

Conceptos

- **IP**: tecnología de convergencia
- **Aplicaciones Multimedia**: Audio, vídeo, juegos, tiempo real, …
- **Calidad de servicio (QoS)**: Capacidad de ofrecer el rendimiento requerido para una aplicación
- **IP ofrece Mejor esfuerzo** (best effort) sin garantías de QoS

Tipos de aplicaciones multimedia

- Flujo de audio y vídeo (streaming) almacenado → Ej. youtube.
- Flujo de audio y vídeo en vivo → Ej. emisora de radio o IPTV.
- Audio y vídeo interactivo → Ej. Skype.

Característica fundamentales

- Elevado ancho de banda.
- Tolerantes relativamente a la pérdida de datos.
- Exigen Delay (retardo) acotado.
- Exigen Jitter (fluctuación del retardo) acotado.
- Se pueden beneficiar de usar direcciones multicast (direcciones destino de grupo).