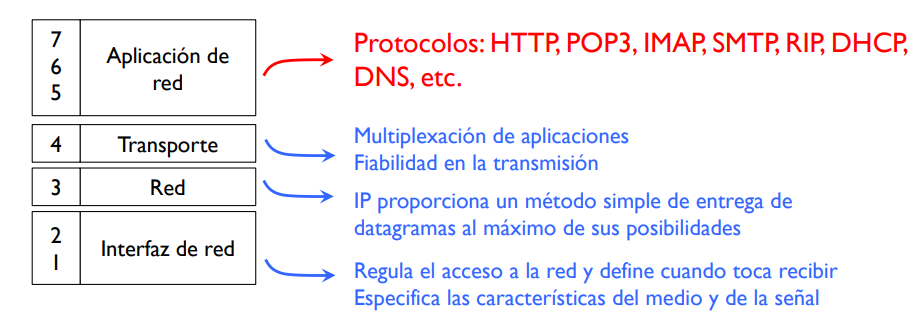
**Tema 5: Aplicacions en Xarxa**

**Introducción**

****

Objetivo de la capa aplicación:

* Mecanismos y protocolos que proporcionan determinados servicios a la red o al usuario.
* Proporcionar formatos estándares a los datos.

**Protocolos del nivel aplicaciones**

Se incluyen en esta capa aquellos protocolos que hacen funcionar las aplicaciones y que sirvan para facilitar su funcionamiento, su aspecto de cara al usuario, su mantenimiento, etc.

Por ejemplo:

* RIP es una aplicación de red ya que facilita la construcción de las tablas de encaminamiento y su mantenimiento en los routers.
* DHCP es una aplicación de red que permite la autoconfiguración de un host.
* DNS es un sistema que facilita los usuarios ya que pueden tratar los hosts con nombre en lugar de con números (@IP).

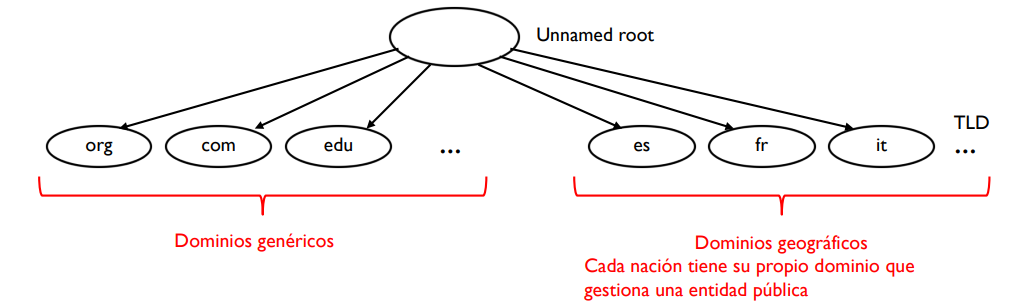
***DNS (Domain Name Service)***

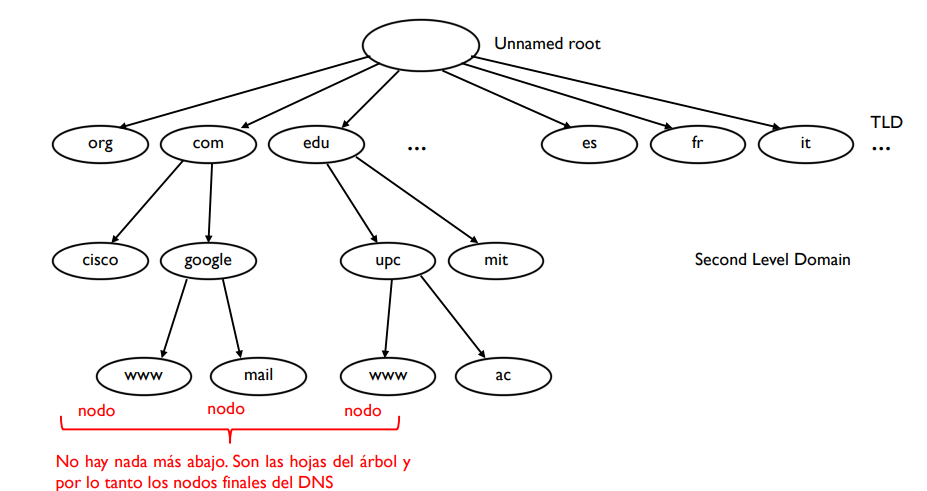
Objetivo:

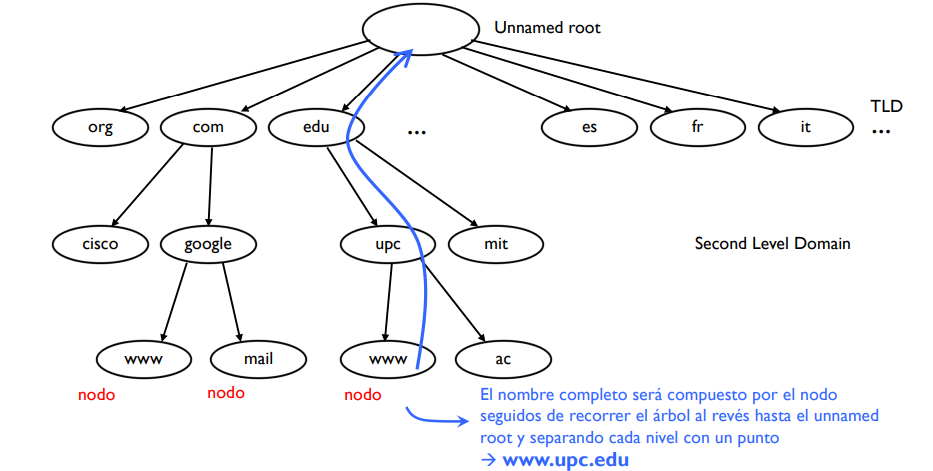
* Permitir al usuario el uso de nombre en lugar de @IP.
* Por ejemplo, es más fácil acordarse:
  + www.upc.edu que 147.83.2.135.
  + mail.google.com que 64.233.184.109.
* ¿Como?
  + Manteniendo una base da datos distribuida y jerárquica de asociaciones @IP – nombre (llamados Resource Record, RR).
* Cada nombre en DNS consiste en un:
  + Nombre del nodo (node name o hostname).
  + Nombre del dominio (domain name).
* ICANN:
  + Internet Corporation for Assigned Names and Numbers.
  + Es el organismo que gestiona DNS.

Jerarquía de los dominios

La base de datos distribuida y jerárquica del DNS tiene estructura árbol donde la raíz es el único punto que no tiene nombre (Unnamed root).







No confundir:

* www.upc.edu -> nombre de host, en particular el servidor de páginas web www de la UPC.
* www.upc.edu/agenda/ -> no es otro nombre de un host, es una página interna del mismo servidor de la UPC que se llama www.upc.edu.

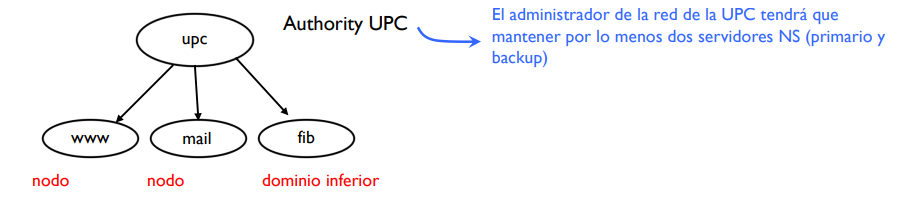
No hay diferencias entre mayúsculas y minúsculas:

* www.upc.edu y WWW.UPC.EDU son los mismos nombres.

Los Name Servers (NS) son los servidores que se ocupan de cada dominio (o zona):

* Son las autoridades del dominio.
* En cada dominio hay dos servidores NS:
  + Servidor NS primario.
  + Servidor NS de backup (se hace una copia del primario generalmente cada 3 h).
* Estos servidores NS mantienen una parte de la base de datos, en concreto:
  + Nombres y @IP de los nodos del dominio.
  + Nombres y @IP de las autoridades (los NS) del nivel inferior.
  + Cada entrada de esta base de datos se llama Resource Record (RR):
    - Hay RR permanentes (se configuran manualmente).
    - Hay cached RR (se eliminan pasado un tiempo).

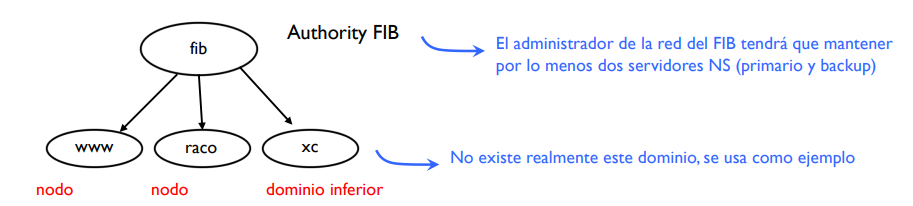
Ejemplo:



* Los NS del dominio UPC tendrán que mantener los RR del propio dominio.
* Por un lado, las asociaciones nombre de los nodos -@IP:
  + El nodo www (servidor de páginas web de la UPC) y su @IP 147.83.2.135.
  + El nodo mail (servidor de correos de la UPC) y su @IP 147.83.2.34.
* Por el otro lado, los nombres y @IP de las autoridades del nivel inferior:
  + El NS del subdominio citm y su @IP 147.83.5.55.

www.fib.upc.edu

* Dominio: fib.upc.edu.
* Nombre del nodo de este dominio: www.
* Autoridad de este dominio: el servidor de nombres (NS) del a FIB.

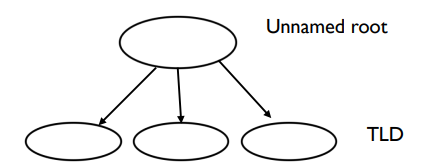


* Los NS del dominio FIB tendrán que mantener los RR del propio dominio.
* Por un lado, las asociaciones nombre de los nodos - @IP:
  + El nodo www (servidor de páginas web de la UPC) y su @IP 147.83.53.43.
  + El nodo raco (servidor del campus virtual) y su @IP 147.83.41.15.
* Por el otro lado, los nombre y @IP de las autoridades del nivel inferior.
  + El NS del subdominio xc y su @IP 147.83.55.59.

Los Name Servers (NS) de la autoridad de root se llaman Root Servers.

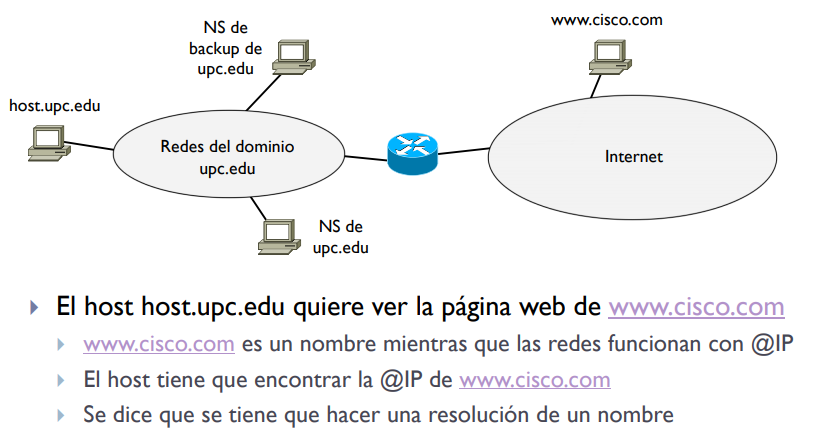
Hay 13 Root Servers (RS) primarios en el mundo:

* Se llaman A, B, C, D, …, M.
* Hay varios backups.
* Tiene las @IP y los nombres de los NS del nivel TLD.



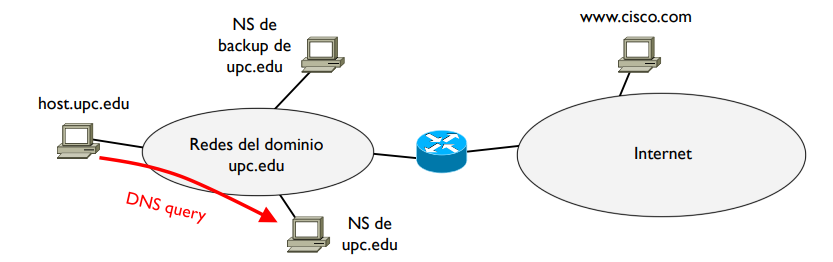
Funcionamiento del DNS:

* Arquitectura cliente – servidor.
* Servidor UDP puerto 53 (well-known port de TCP/IP).
  + Comunicación no fiable.
* DNS usa TCP cuando se hacen backups entre el NS primario y el de backup.



El host comprueba si tiene una resolución ya disponible en su memoria cache de DNS:

* En efecto, cada vez que se hace una resolución DNS, los hosts mantienen esta resolución durante un cierto tiempo en una memoria llamada cache.
* Este tiempo es programable.

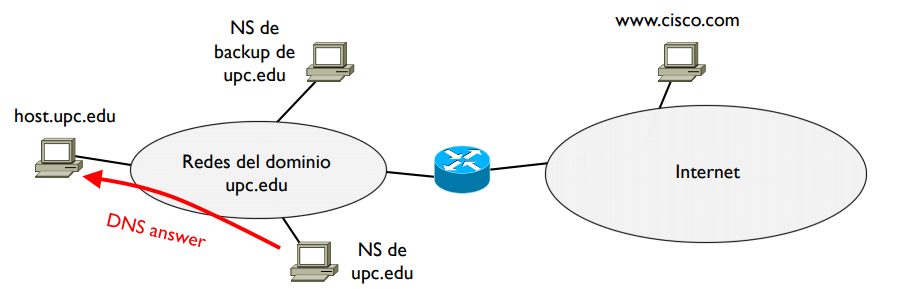


Si no tiene esta resolución, contacta la autoridad (el NS primario) de su dominio:

* Envía al NS una petición de resolución (DNS query) con el nombre a resolver, es decir www.cisco.com.
* Se usa el protocolo UDP con puerto destino 53.
* El puerto origen es un número efímero.

El NS comprueba si ya tiene resuelto el nombre www.cisco.com:

* El NS tiene una base de datos con algunas resoluciones ya configuradas (estáticas, es decir están configuradas manualmente y no se borran con el tiempo).
* A parte, el NS se guarda todas las resoluciones hechas de forma dinámica durante generalmente 2 días (en una memoria llamada cached RR).
* Es decir, podría ser que otro host del dominio upc.edu haya pedido una misma resolución en el pasado y por lo tanto el NS podría tenerla guardada en su memoria.



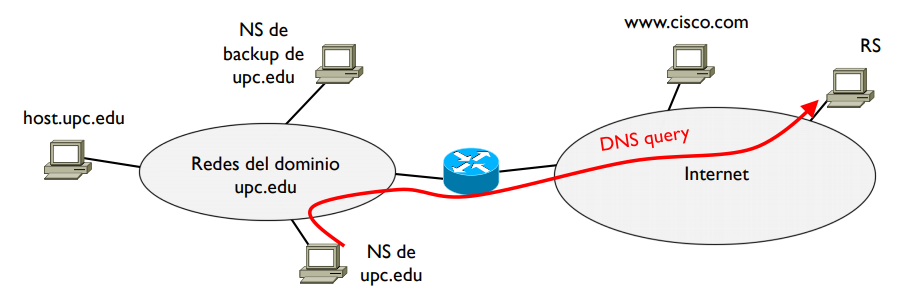
Si ya tiene esta resolución, el NS contesta con un DNS answer al host:

* En la respuesta se indica la @IP de www.cisco.com y también el nombre y @IP de la autoridad que ha proporcionado esta resolución.

Si no la tiene, el NS debe buscar la resolución en Internet.

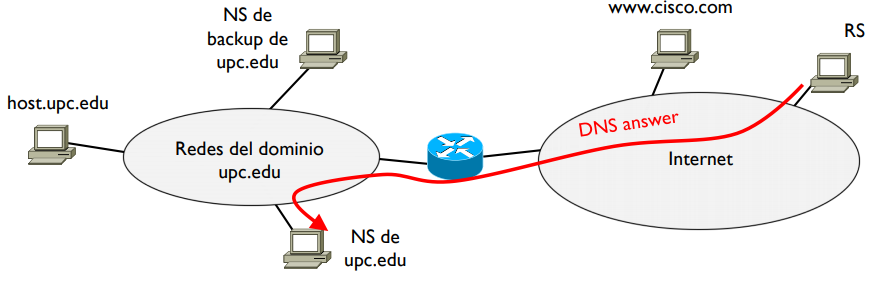
Suponiendo que el NS no sabe nada del nombre www.cisco.com, el NS debe empezar la resolución a partir de un RS.

* Un NS debe por lo tanto tener configurado en su base de datos el nombre y @IP de por lo menos un RS.

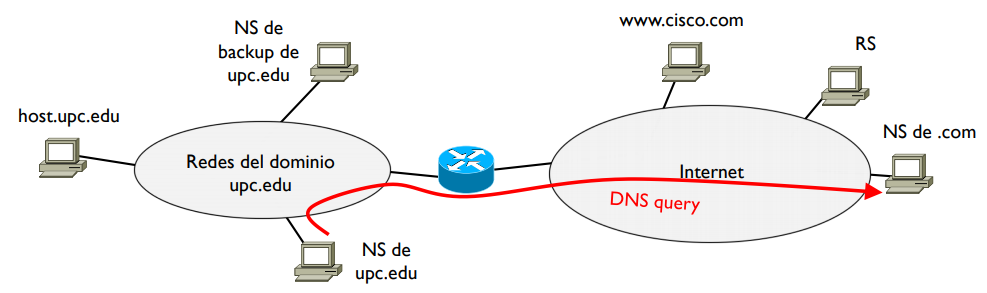


El NS de upc.edu pide a un RS la @IP de un NS del dominio .com:

* Es decir, la resolución siempre es a partir de la raíz y se recorre el árbol hasta la hoja.

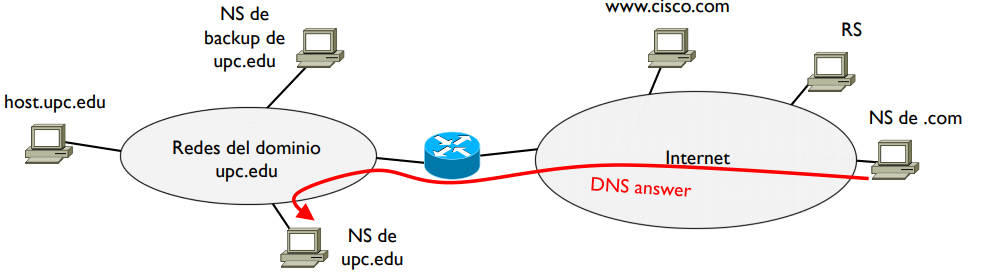


El RS contesta con la @IP de un NS del dominio .com.

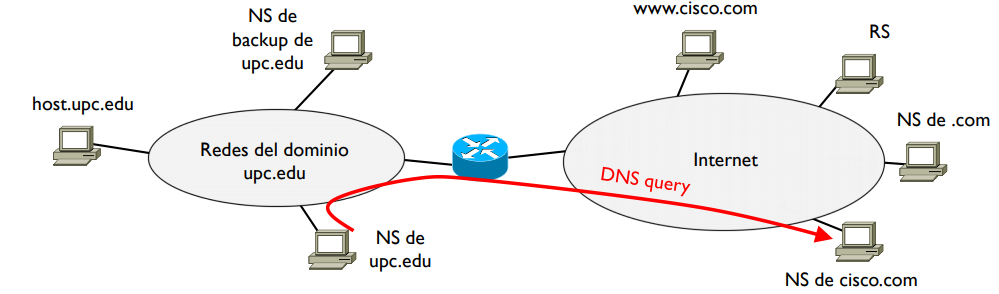


Con esta información, el NS de upc.edu ahora puede pedir al NS del dominio .com la @IP de un NS del dominio inferior cisco.com:

* Es decir, se ha bajado de la autoridad unnamed root a una autoridad del nivel TLD.

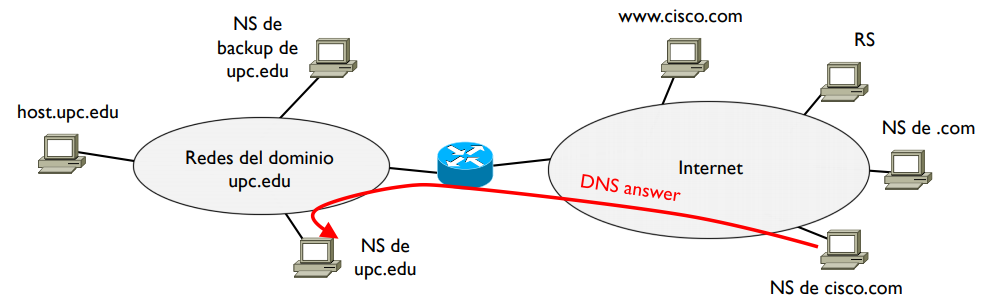


El NS del dominio .com contesta con la @IP del NS del subdominio cisco.com.

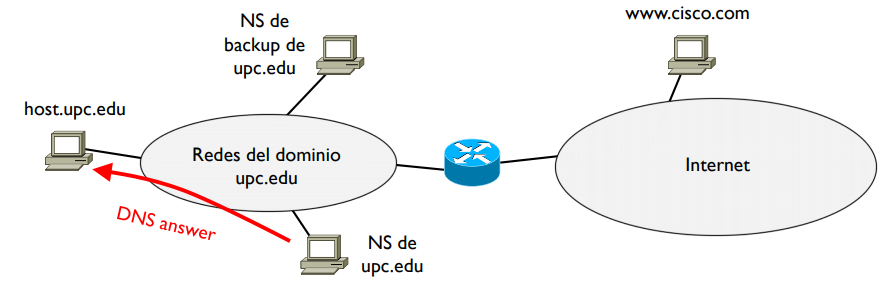


Con esta información, el NS de upc.edu ahora puede pedir al NS del dominio cisco.com la @IP de uno de sus nodos, en concreto el www (servidor de páginas web):

* Es decir, ya se ha llegado a la autoridad que conoce lo que se estaba buscando.



El NS de cisco.com proporciona la @IP del servidor de páginas web de su dominio, es decir la @IP de www.cisco.com.



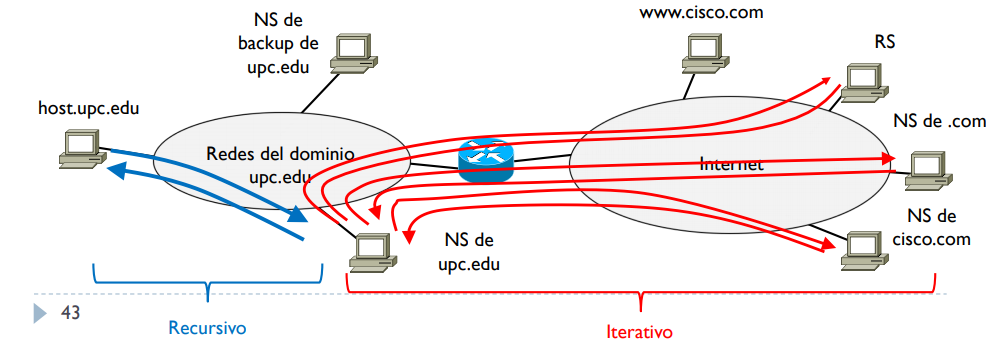
Ahora el NS de upc.edu tiene lo que le había pedido el host y finalmente le puede contestar.

DNS detalles:

Una resolución puede ser:

* Recursiva: cuando un host pide a otro que resuelva un nombre.
* Iterativa: cuando un host puede resolver un nombre recorriendo el árbol jerárquico DNS.

Es decir, generalmente un servidor NS usa una resolución iterativa mientras que un host cliente usa la recursiva (ya que necesita el NS para resolver un nombre).



Un mismo nombre puede estar asociado a diferentes @IP (es decir diferentes nodos):

* Por ejemplo, no existe un único servidor www.google.com.
* En estos casos, los NS generalmente proporcionan la @IP del nodo más próximo al que ha pedido la resolución).

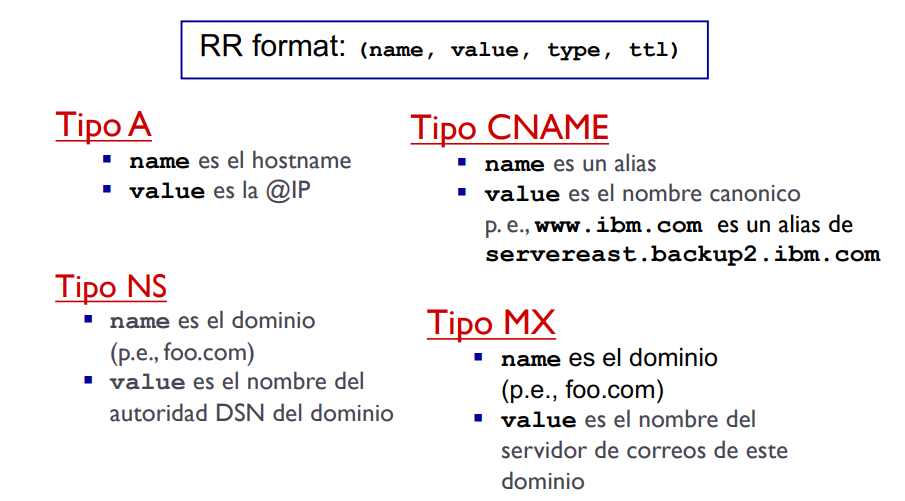
Un mismo host puede tener diferentes nombres:

* Se llaman alias.
* Se puede pedir que un NS proporcione todos los alias de un nombre.
* Es una resolución que se dice de tipo CNAME.

También existe la resolución inversa:

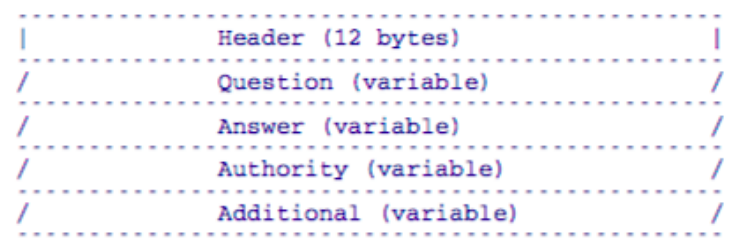
* Conocida la @IP, se pide por el nombre.
* Es una resolución que se dice de tipo PTR.

DNS RR:



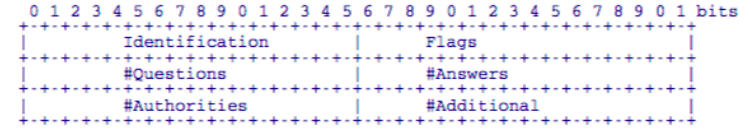
Formato DNS:

Todos los mensajes DNS tienen el mismo formato:



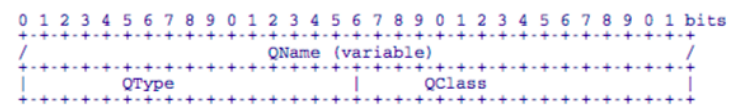
* Header: cabecera, indica el tipo de mensaje que se está enviando.
* Question: lo que se quiere resolver.
* Answer: la respuesta.
* Authority: Nombres de las autoridades que han proporcionado la respuesta.
* Additional: información adicional, típicamente la @IP de las autoridades.

*Header*



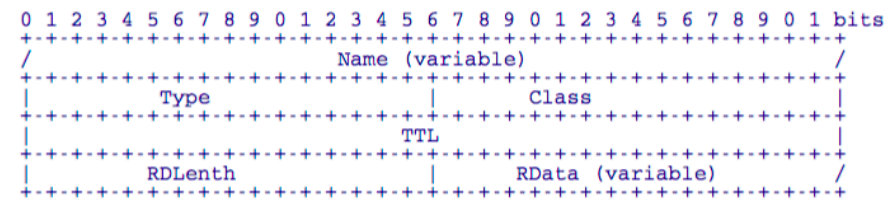
* Tamaño fijo de 12 bytes.
* Identification: un número aleatorio de 16 bits que sirve para relacionar una pregunta con una respuesta.
* Flags: bits que indican información sobre el tipo de mensaje:
  + Flag QR: 0 si pregunta, 1 si respuesta.
  + Flag AA: 1 si es una respuesta de una autoridad.
  + Flag RD: 1 indica que se pide una resolución recursiva.
* El resto de campos indican el número de preguntas, respuestas, autoridades y additional hay en el resto del mensaje.

*Question*



* QName: indica el nombre que se quiere resolver.
* QType: indica el tipo de pregunta:
  + A: Para conocer una dirección @IP a partir de un nombre.
  + NS: Para conocer el nombre de un Name Server de un domain.
  + PTR: Para conocer un nombre a partir de una @IP (inverso de A).
  + MX: Para conocer el nombre del server de email de un domain.
  + CNAME: Para conocer todos los alias de un nombre.
* QClass: indica el tipo de direccionamiento que se usa (1 para IPv4).

*Answer, authority y additional*

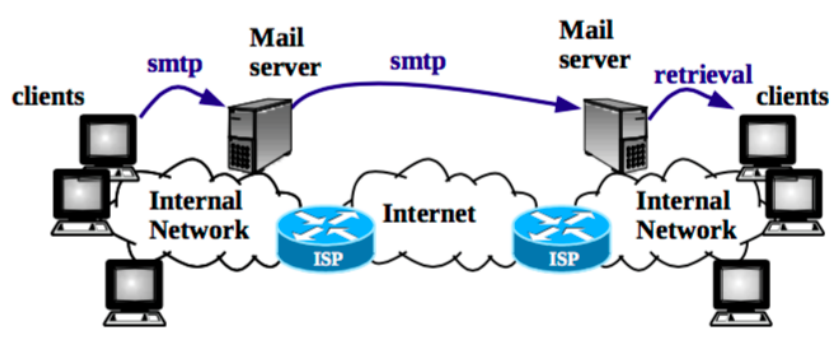


* Name, Type y Class como en Question.
* TTL (Time To Live): número de segundos que hay guardar la resolución en la cached RR.
* RDLength: tamaño de la resolución en bytes.
* RData: la resolución:
  + @IP si es de tipo A.
  + Un nombre si es de tipo NS, MX o PTR o varios nombres si es CNAME.

***EMAIL***

* Electronic mail, correo electrónico.
* Una de las primeras aplicaciones que se desarrollaron cuando se creó Internet.
* Basado en el paradigma cliente – servidor.
* Hay dos componentes básicos:
  + Protocolo para el envío de los correos: SMTP.
  + Protocolo para recuperar correos (retrieval): POP, IMAP, HTTP.

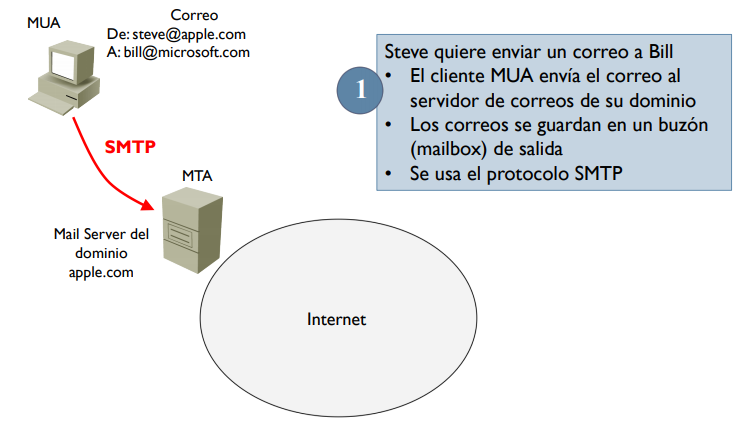
Modelo:

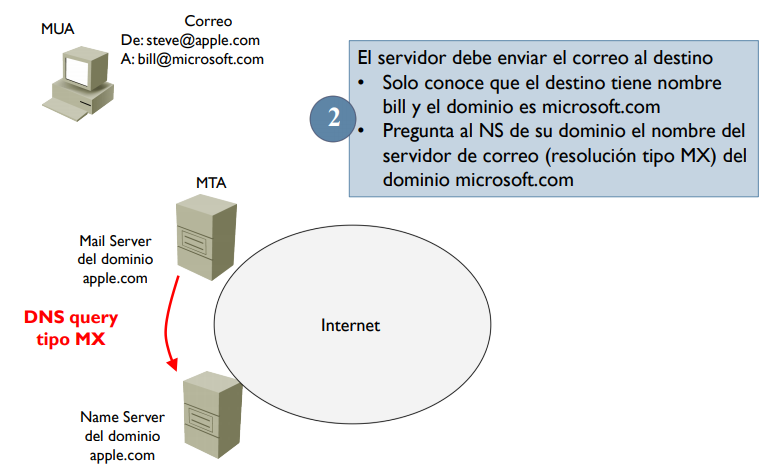


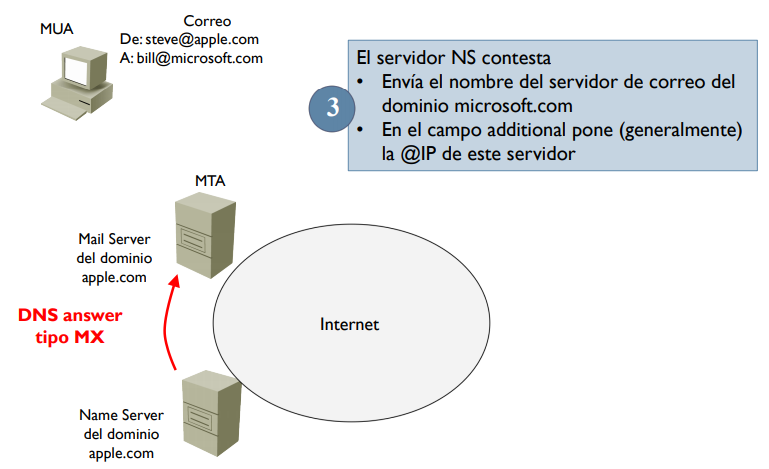
SMTP (Simple Mail Transfer Protocol):

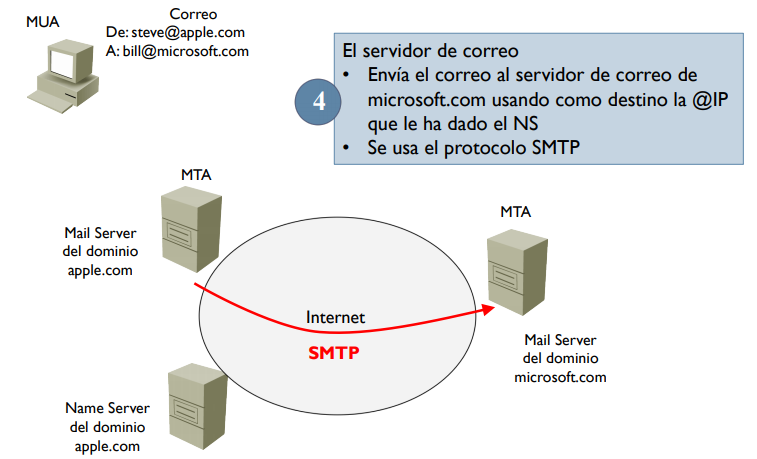
* TCP, puerto 25 (Transmisión fiable).
* Definiciones:
  + MUA: Mail User Agent, p.e. Mail, Outlook, Thunderbird, etc.
  + MTA: Mail Transfer Agent, p.e. Sendmail, Postfix, etc.
* El formato de un correo es Nombre@dominio:
  + Nombre: el nombre del usuario, debe tener una cuenta en el servidor de correo del dominio.
  + Dominio: el dominio del sistema donde se encuentra el servidor de correos y el usuario.

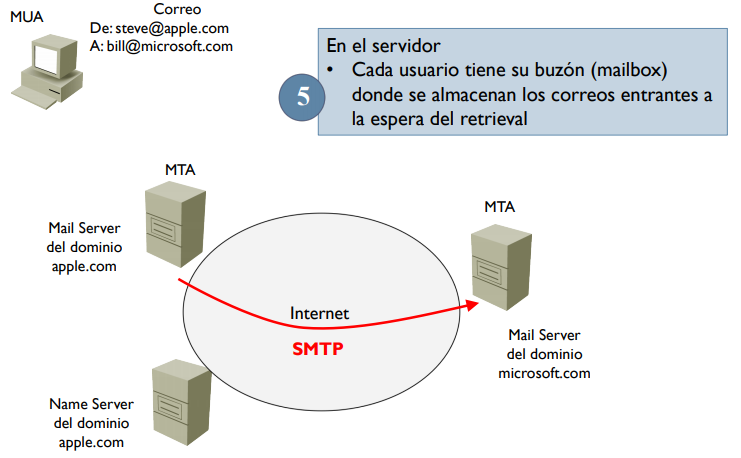
Funcionamiento:

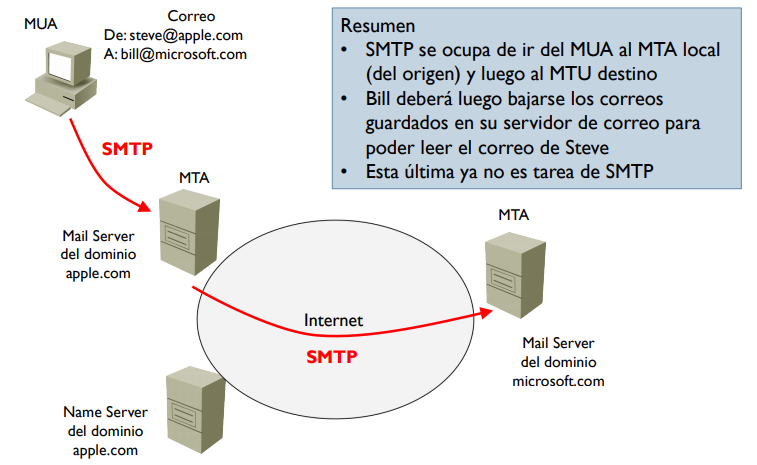


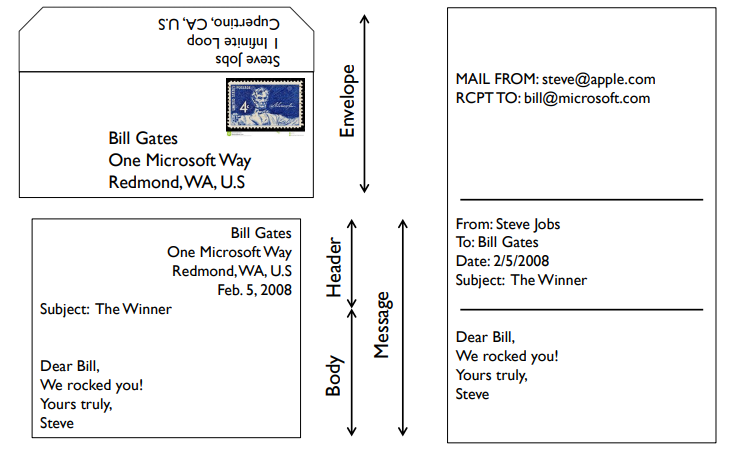




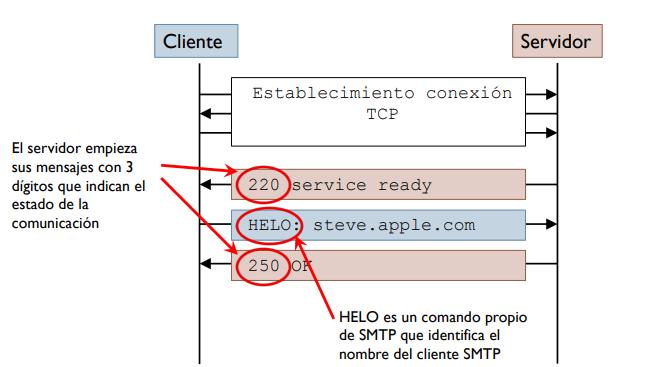


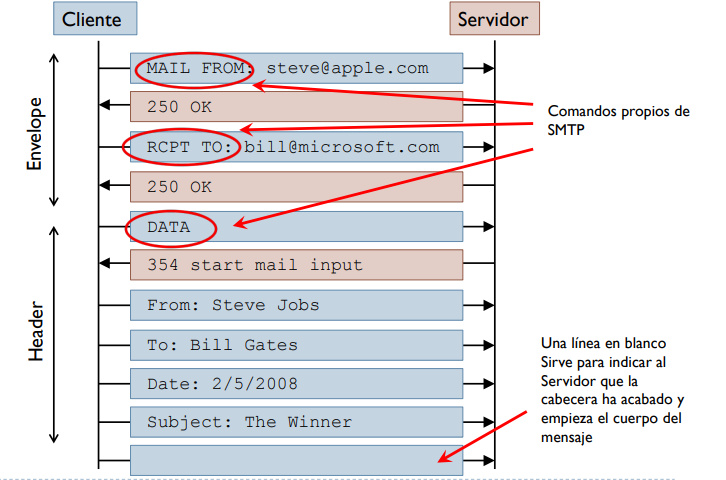


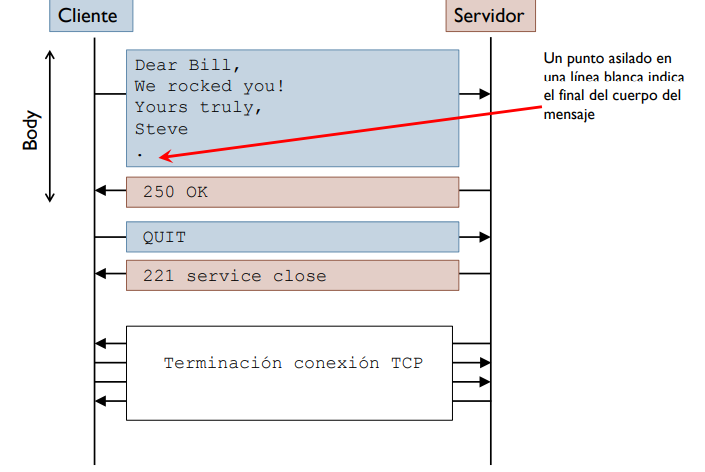


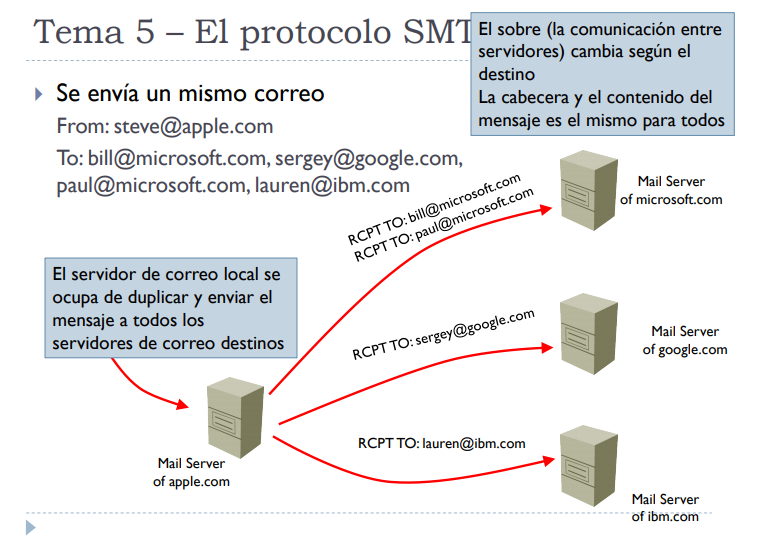


El protocolo SMTP es diseñado para que sea muy simple (pocos comandos) y basado en texto ASCII.







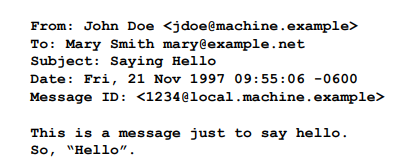


Un mensaje email puede usar dos formatos distintos:

* Internet Message Format (IMF).
* Multipurpose Internet Mail Extensions (MIME).

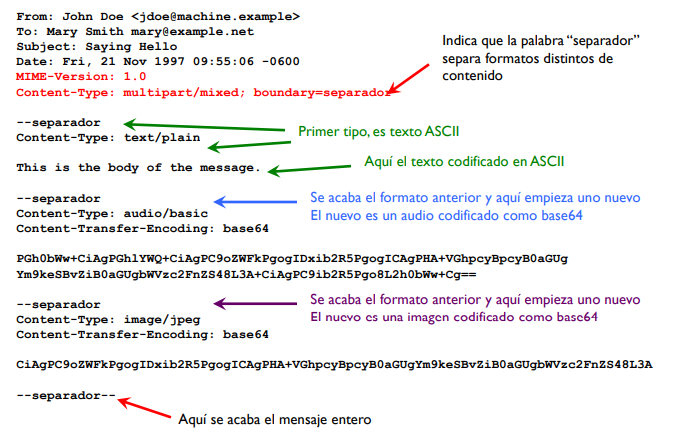
*Internet Message Format (IMF):*

* Mensaje exclusivamente en formato texto ASCII.

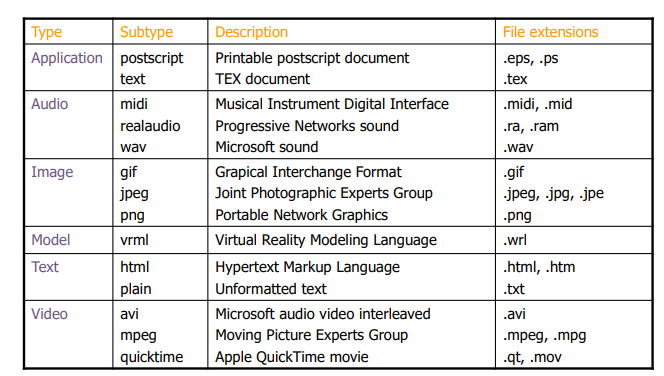


*Multipurpose Internet Mail Extensions (MIME):*

* Para incluir datos no-ASCII (por ejemplo, audio, video, fichero…)
* El mensaje se compone:
  + De una cabecera que incluye la definición de una “palabra” que hace de frontera entre tipos diferentes.
  + De un cuerpo que puede contener datos de diferente formato.



Tipos MIME más comunes:

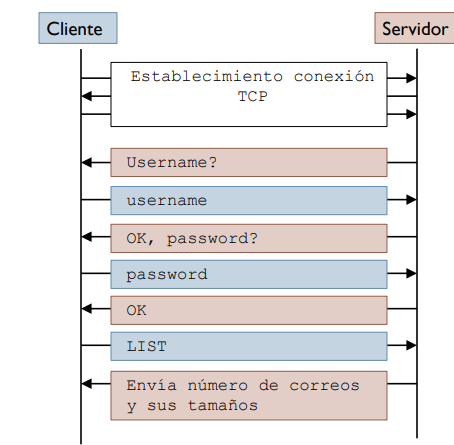
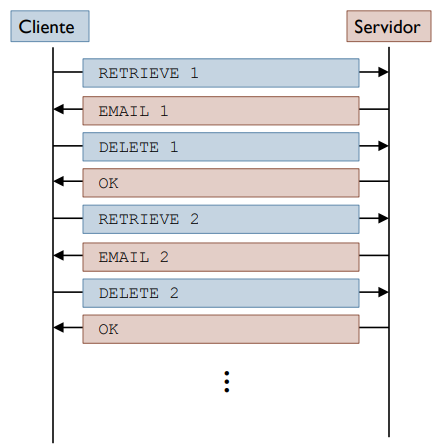
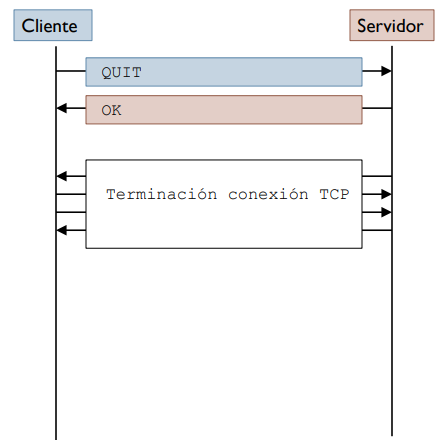


Actualmente se usa la versión ESMTP, una versión extendida de SMTP. Algunas diferencias:

* El primero comando usado indica la versión: HELO para SMTP y EHLO para ESMTP.
* Se puede añadir autentificación usando el comando AUTH para que el cliente se autentifique antes de enviar el mensaje.
* Se puede añadir una capa de seguridad usando encriptación TLS para codificar los mensajes.
* Hay más comandos como SIZE, HELP, PIPELING, etc. Definidos en el estándar ESMTP.

POP (Post Office Protocol):

* TCP puerto 110.
* Características principales:
  + El servidor guarda los mensajes de un usuario todos en un mismo sitio.
  + El cliente al conectarse, baje todos los mensajes en local y estos (por defecto) se borran del servidor.
  + Se puede configurar para que el servidor guarde una copia de los correos durante un cierto tiempo (que puede ser infinito).



IMAP (Internet Message Access Protocol):

* TCP Puerto 143.
* Características principales:
  + Los correos se quedan guardados en el servidor hasta que el usuario no los borres explícitamente. De esta forma se puede acceder a los correos desde cualquier dispositivo ya que no se guardan en local.
  + También hay la opción de crear una copia en local (de manera que no se necesita estar conectados a Internet siempre).
  + El usuario puede manejar los correos en el servidor. Por ejemplo, puede crear carpetas, mover correos, bajar solo parte de un mensaje, etc.

Basado en la web:

Típicamente un web server (HTTP), p.e. Gmail, Hotmail, Yahoo, webmail, etc. Se accede a través de un cliente **HTTP** (Firefox, Safari, Chrome, etc.) directamente al propio servidor de correos. Características principales:

* Los correos se quedan guardados en el servidor hasta que el usuario no los borres explícitamente. De esta forma se puede acceder a los correos desde cualquier navegador web ya que no se guardan en local.
* El usuario puede manejar los correos en el servidor. Por ejemplo, puede crear carpetas, mover correos, bajar solo parte de un mensaje, etc.

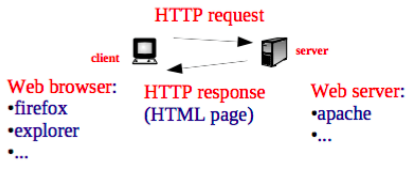
***WEB***

WorldWide Web, www

* Creado por Tim John Berners-Lee en 1989 y desarrollado en los 90s para proporcionar acceso simple a la información en Internet.

Componentes:

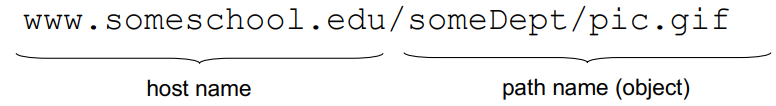
* Capa de transporte: TCP, puerto 80.
* Protocolo de aplicación: HyperTextTransfer Protocol (HTTP).
* Lenguaje: HyperText Markup Language (HTML).



Una página web consiste de objetos o recursos.

* Un objeto puede ser un fichero HTML, una imagen JPEG, un applet Java, un fichero audio, etc.
* Cada objeto es localizable a través de un URL (Uniform Resource Locator).

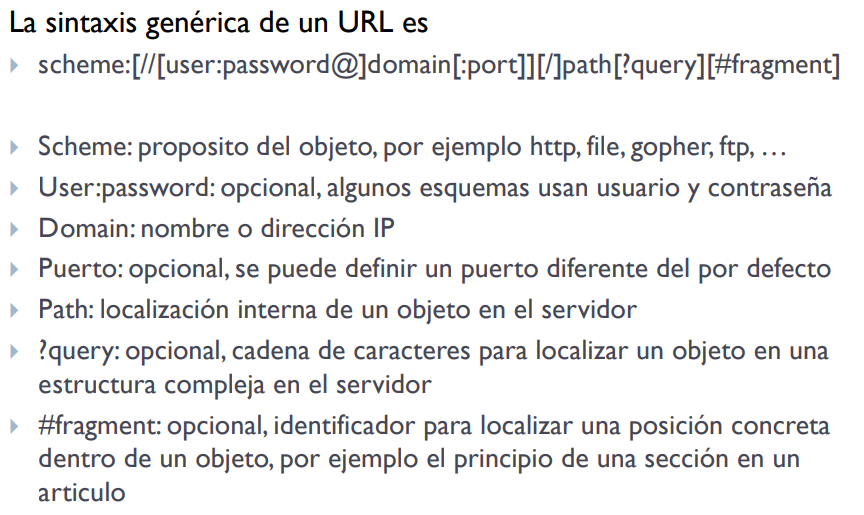
Ejemplo de URL:



Típicamente hay un objeto base que es un fichero HTML que incluye luego varios otros objetos referenciados.

Ejemplo de referencia:





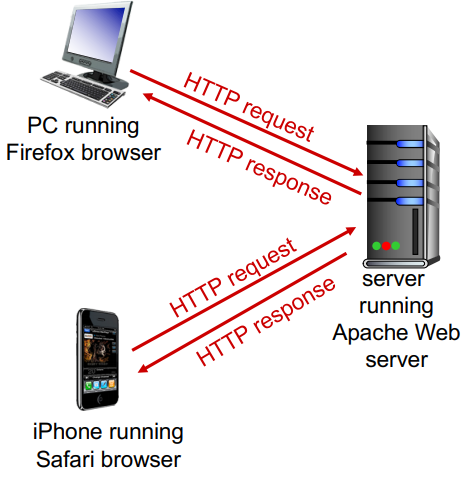
Funcionamiento HTTP:

Basado en la arquitectura cliente-servidor:

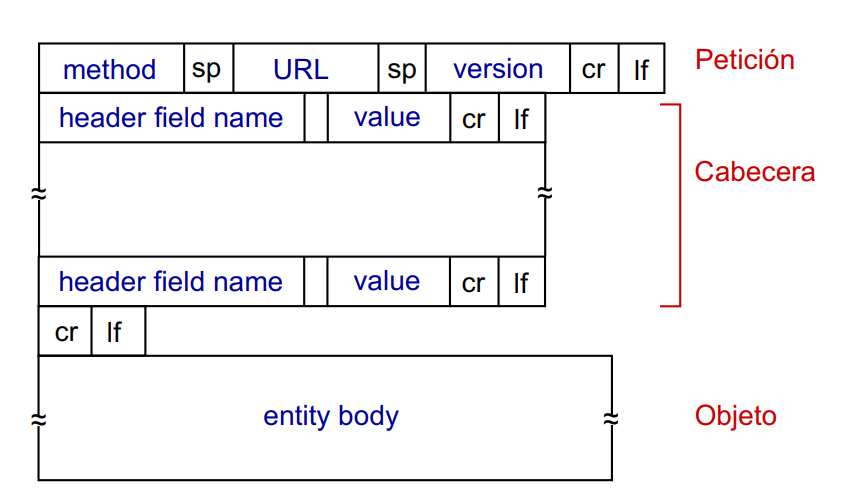
* Cliente: navegador que solicita, recibe (usando el protocolo HTTP) y muestras objetos Web.
* Servidor: servidor Web que envía (usando el protocolo HTTP) objetos en respuesta a peticiones de los clientes.

HTTP es “stateless”:

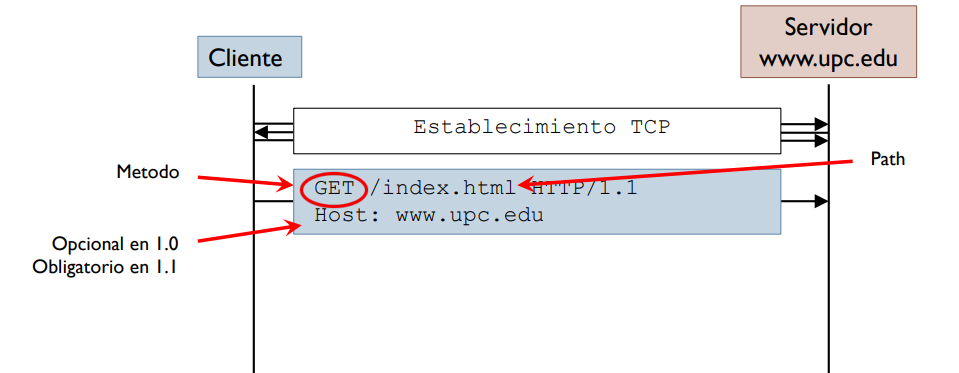
* Los servidores Web no mantienen información sobre las peticiones pasadas de los clientes.
* No mantienen estados.



Formato HTTP:



Petición HTTP:



Métodos HTTP:

Métodos:

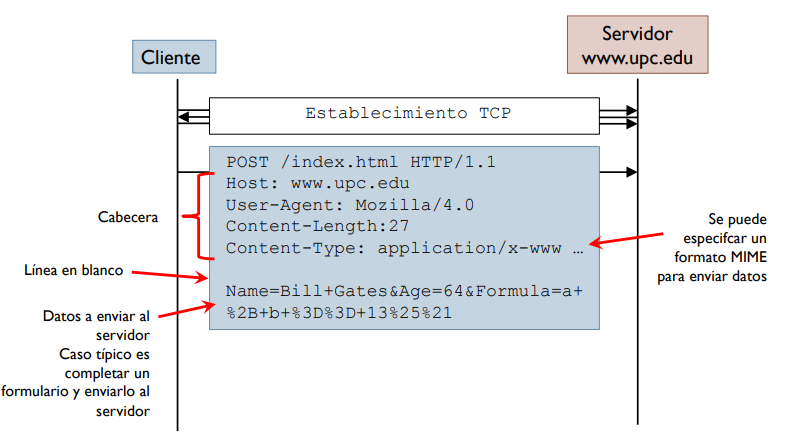
* GET: solicitar el contenido de una web.
* POST: enviar datos a un servidor Web, por ejemplo, el contenido de un formulario.
* HEAD: para obtener solo la cabecera del objeto, pero no el objeto mismo.

HTTP 1.1 añade:

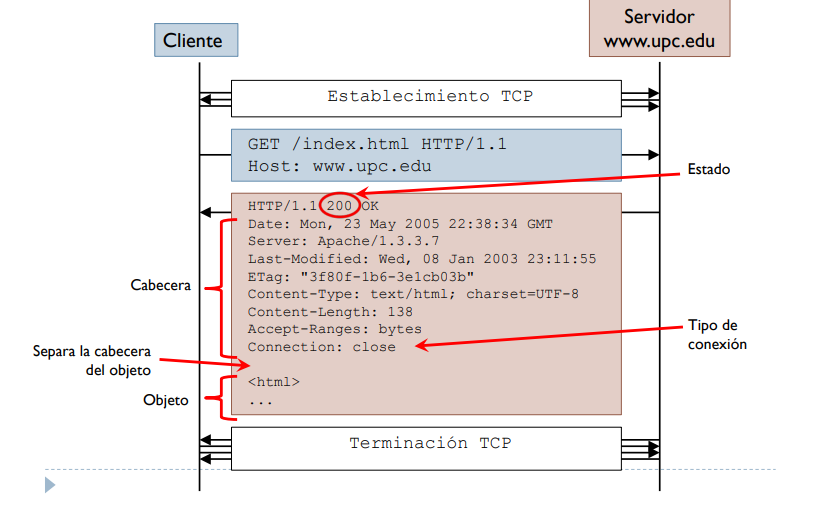
* OPTIONS: para conocer los métodos aceptados por el servidor.
* PUT: para insertar o substituir un determinado objeto.
* DELETE: para eliminar un determinado objeto en el servidor web.
* PATCH: para modificar parcialmente un objeto.
* TRACE: para recibir ecos y saber si ha habido cambios.
* CONNECT: generalmente usado para añadir una capa de seguridad SSL a la comunicación (HTTPS).

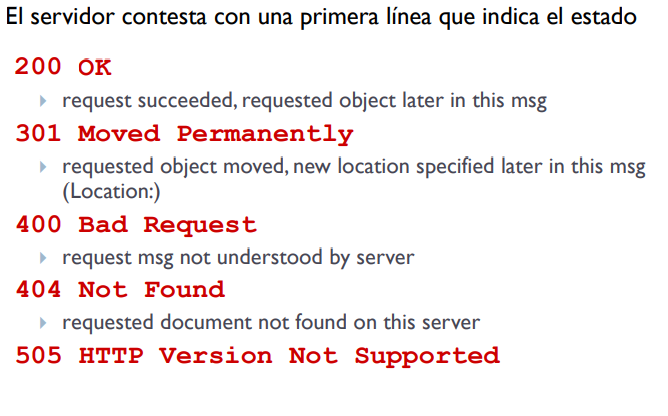
Petición HTTP:

Para enviar datos.



Respuestas HTTP:





Tipos de conexión HTTP:

No persistente (por defecto en HTTP 1.0):

* El servidor envía el objeto solicitado y cierra la conexión.
* Si hay más objetos en una página, hay que abrir una conexión para cada uno.
* Connection: close.

Persistente (por defecto en HTTP 1.1):

* El servidor mantiene la conexión abierta durante un tiempo determinado.
* Si hay más objetos, se pueden pedir uno después del otro (secuencialmente).
* Connection: keep-alive.

Persistente con pipeling (solo en HTTP 1.1 pero uso no recomendado):

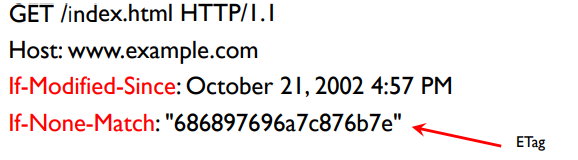
* El servidor mantiene la conexión abierta durante un tiempo determinado y el cliente puede pedir nuevos objetos en el momento en que encuentra nuevas referencias en el objeto que se está bajando aunque la descarga no se haya completado (transmisiones en paralelo).
* Si página con muchos objetos distintos, sobrecarga de GET que el servidor debe servir y contestar.
* Connection: keep-alive.

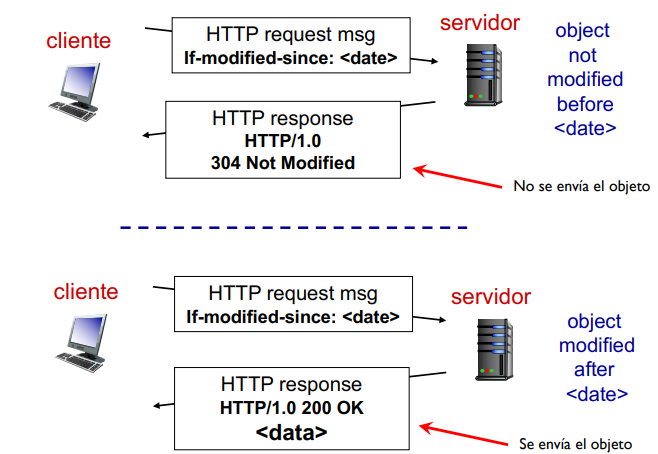
HTTP caching:

Objetivo: no enviar objetos si ya se han visitado previamente.

* El cliente almacena las páginas web visitadas en una cache local.
* Cuando el usuario quiere ver una página web, el cliente mira su cache local para ver si ya tiene esta página:
  + Si la tiene, envía un Conditional GET al servidor.
  + Esta petición le dice al servidor que le envíe la página solo si es diferente de la que ya tiene.
  + Para saber si es diferente se usan los campos Etag o Date.

Ejemplo:



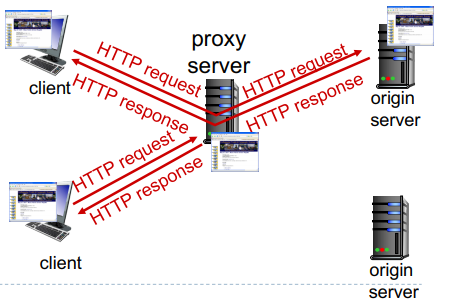


HTTP proxy:

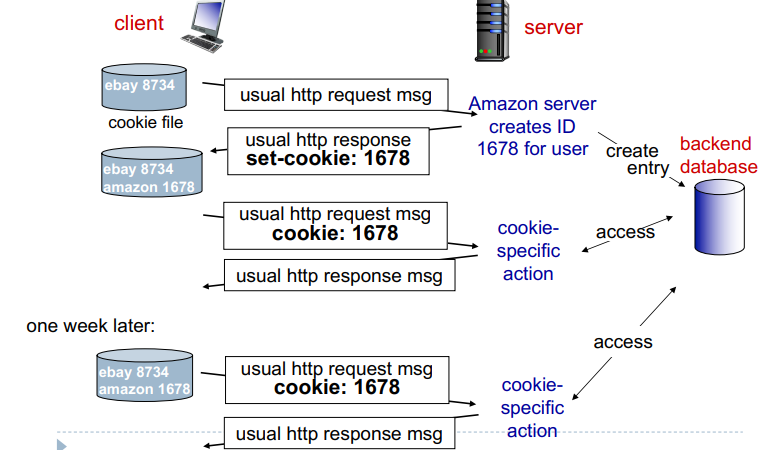
Objetivo: contestar a los clientes sin que estos accedan al servidor.

* Se usa un proxy como intermediario.
* Los clientes se comunican con el proxy.
* El proxy accede al servidor y almacena las páginas web solicitada y sus objetos temporáneamente en su cache local.

Ventajas:

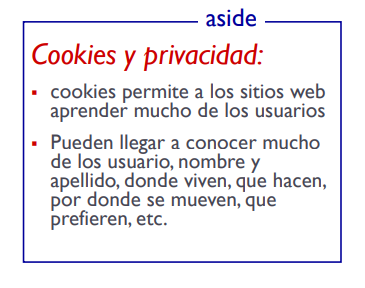
* Seguridad.
* Rapidez.
* Menor carga en los servidores.
* Servidor sin acceso a Internet (@IP privada).

HTTP cookies:



Objetivos:

* Facilitar autorización.
* Mantener carritos de la compra.
* Recomendar productos.
* Mantener sesiones abiertas (Web e-mail).



***HTML (Hyper-Text Markup Language)***

* Basado en tags.

Ejemplo:

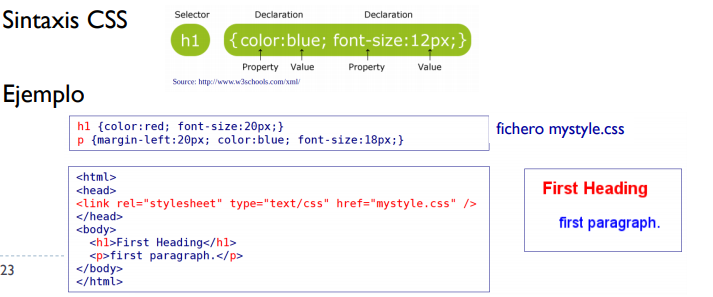


Forms: el documento acepta inputs del usuario que se envían al server.

Scripting: permite ejecutar programas. El programa se ejecuta en la máquina del cliente cuando se carga el documento o en el momento de hacer un click a un enlace.

Cascading Style Sheets (CSS):

* Permite separar el contenido del documento y la forma de presentación de este, como las capas o layouts, los colores y las fuentes.
* De esta forma varios documentos HTML pueden usar el mismo CSS para definir su presentación.
* Si se quiere cambiar de estilo, solo se cambia el CSS y no los HTML.



***CHARSETS***

Dificultad a la hora de crear un programa para “todo el mundo”.

Muchos idiomas, muchos alfabetos, muchas monedas, muchas maneras de indicar la fecha y la hora, etc.

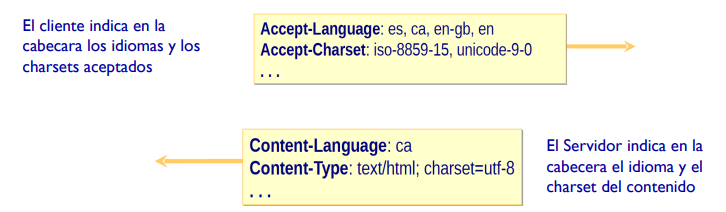
* Un mismo idioma puede tener además variantes, por ejemplo, es\_ES, es\_CO.
* Un mismo alfabeto puede estar compartido entre varios idiomas, por ejemplo, catalán y francés.

Se necesita que un software sea “adaptable”:

* Internationalization (i18n): adaptación de un software a diferentes idiomas sin necesidad de modificar el código.
* Localization (L10n): proceso mediante el cual un producto internacionalizado se configura para una determinada región, aprovechando las opciones que la internacionalización previa de este producto ha permitido.

Típicamente se negocia el idioma y charset a usar en los protocolos de nivel aplicación.

Por ejemplo, en HTTP:



Ejemplo de charset:

* ISO 8859-1 (Latin-1): 190 caracteres + control = 256 combinaciones -> 8 bits.

Unicode (Universal Coded Character Set):

Todos los caracteres de todos los idiomas + símbolos matemáticos + emoticonos + símbolos musicales + … = Universal Character Set (ucs).

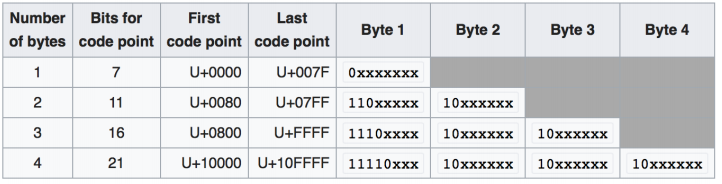
* Actualmente 136,755 caracteres (versión 10.0, junio 2017).
* Hay 1,114,112 posiciones posibles, desde el U+0 hasta U+0FFFF.
* U+0030 es el 0, U+0041 es A, U+00C7 es Ç, U+0BB6 es ண, etc.

Se codifican en una secuencia de bits de acuerdo a método (algoritmo) de encoding:

* Universal Transformation Format (UTF).
* UTF-8: la codificación dominante en la web desde 2009, 90.3% en noviembre 2017. Compatible con ASCII.

*UTF-8*

Codificación orientada a byte con símbolos de longitud variable, según la tabla:



Ejemplo 1:

* 0 es U+0030:
* 1 byte (entre U+0000 y U+007F).
* la codifica a usar es 0xxx xxxx -> se necesitan 7 bits.
* 30 en binario es 11 0000.
* Se añaden 0 delante 011 0000.

Se envía: 0011 0000.

Ejemplo 2:

* ¢ es U+00A2:
* 2 bytes (entre U+0080 y U+07FF).
* la codifica a usar es 110x xxxx 10xx xxxx -> se necesitan 11 bits.
* A2 en binario es 1010 0010.
* 0 delante 000 1010 0010.

Se envía: 1100 0010 1010 0010.

Ejemplo 3:

* € es U+20AC:
* 3 bytes (entre U+0800 y U+FFFF).
* la codifica a usar es 1110 xxxx 10xx xxxx 10xx xxxx -> 16 bits.
* 20AC en binario es 10 0000 1010 1100 -> 0010 0000 1010 1100.

Se envía: 1110 0010 1000 0010 1010 1100.