

TEMA 5 Capa de aplicación



Profesora: Guadalupe Miñana Roperó

Servicios básicos de red: DHCP

DHCP: Dynamic Host Configuration Protocol

- **Configuración automática de los parámetros de la red**
 - Dirección IP, máscara, router predeterminado, servidores DNS...
- **Clientes DHCP**
 - No disponen de una configuración de red fija
 - Cuando arranca el sistema busca un servidor DHCP que le proporcione la información de configuración de red necesaria
- **Servidor DHCP**
 - Proporciona los parámetros de configuración de la red a los clientes que lo solicitan
 - Ámbitos de aplicación
 - Entornos móviles (redes inalámbricas, hoteles, congresos, etc.)
 - Es utilizado en redes locales para asignar direcciones IP dinámicas y otros parámetros de red a los dispositivos conectados
 - Acceso telefónico o ADSL a través de ISP
- **Usa protocolo UDP**

Cómo funciona DHCP

El proceso de asignación de IP sigue cuatro pasos clave, conocidos como DORA:

1. **Discovery** (Descubrimiento):

- El cliente envía un mensaje broadcast **DHCPDISCOVER** para buscar para localizar a los servidores DHCP activos

2. **Offer** (Oferta):

- El servidor responde con un mensaje **DHCPOFFER**, con una oferta de parámetros de configuración conforme a la situación del cliente

3. **Request** (Solicitud):

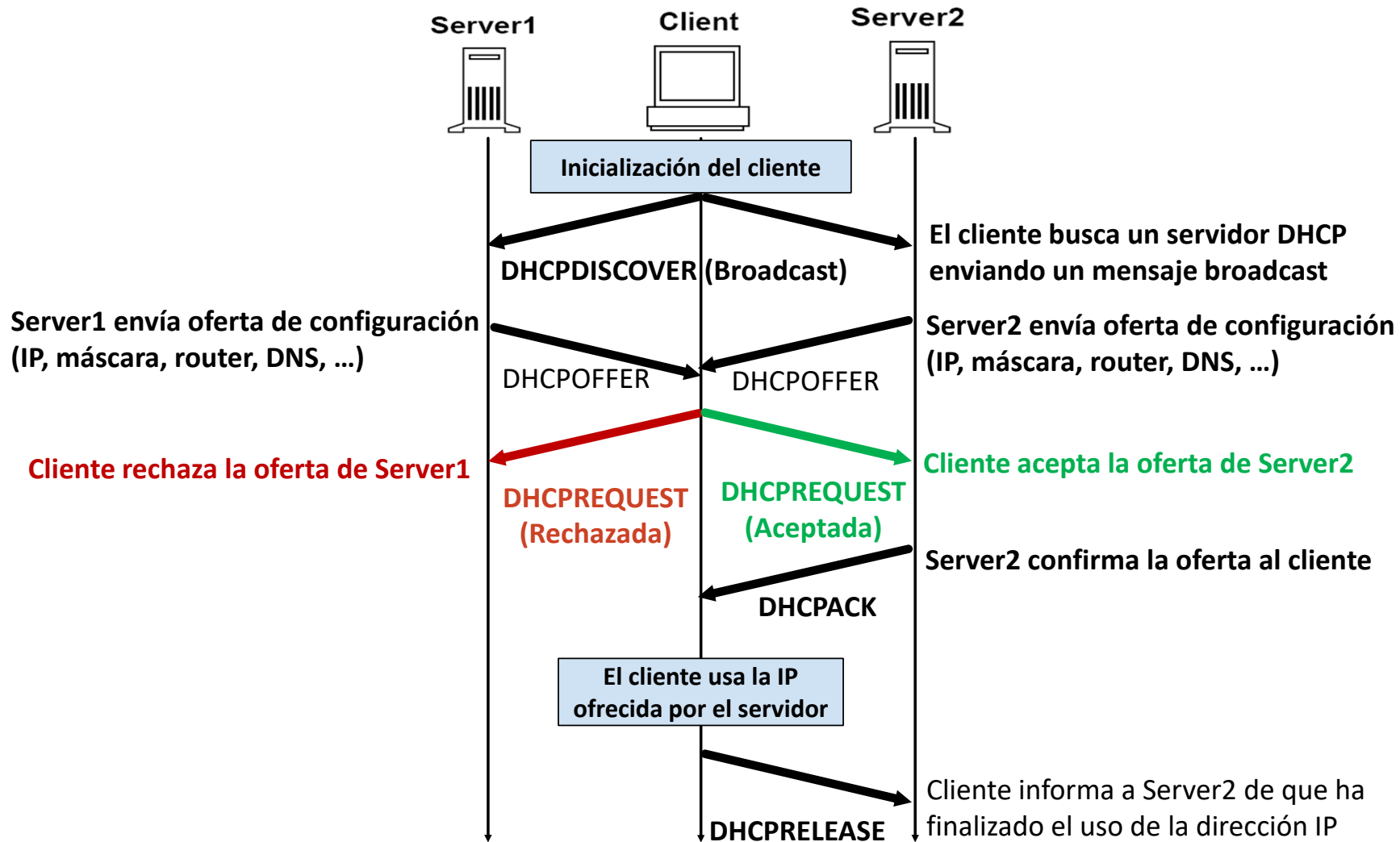
- El cliente responde con un mensaje **DHCPREQUEST**. Este mensaje tiene dos posibles respuestas
 - **Oferta aceptada** y solicitud del uso los parámetros ofertados
 - **Oferta rechazada**

4. **Acknowledge** (Confirmación):

- El servidor envía un **mensaje DHCPACK** indicando los parámetros definitivos

Cuando el cliente ha finalizado el uso de la dirección IP envía un mensaje **DHCPRELEASE** al servidor para informarle

Cómo funciona DHCP



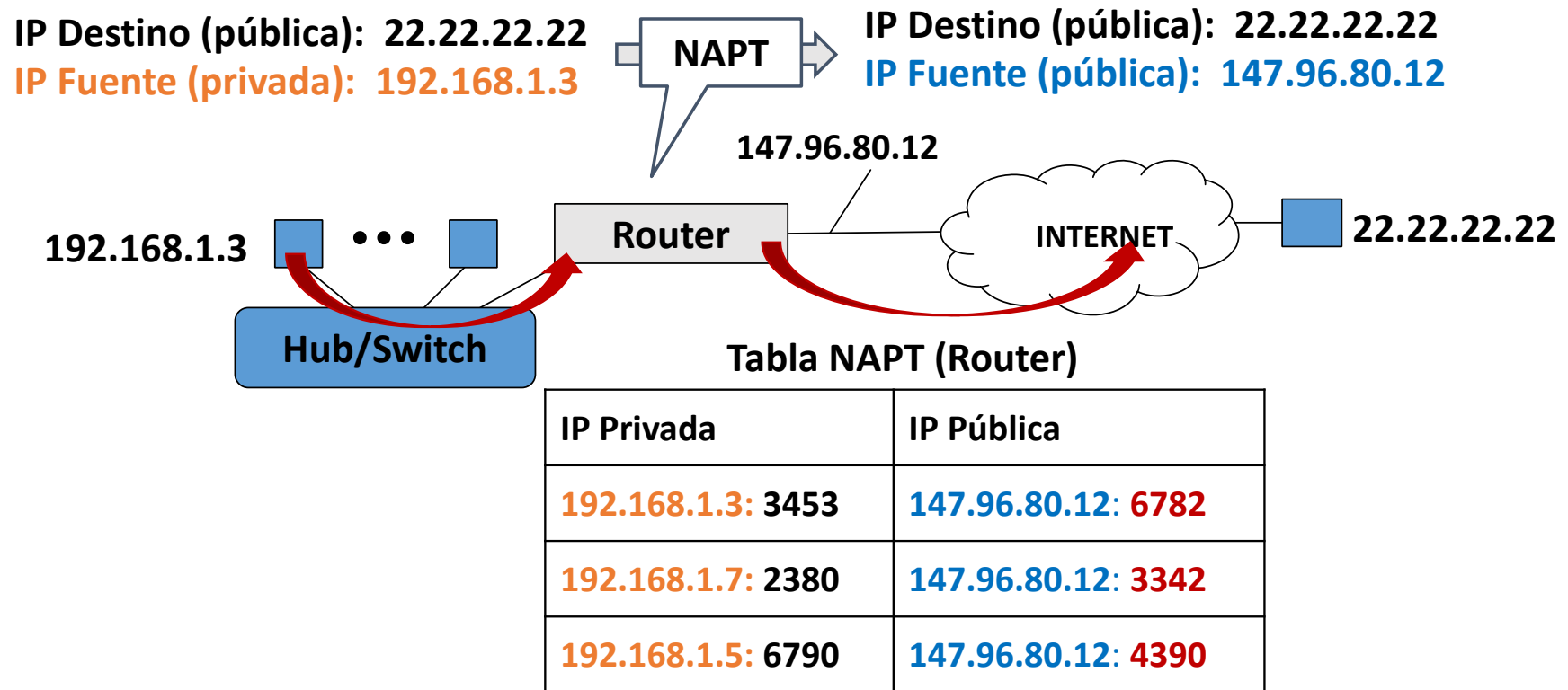
Servicios básicos de red: NAPT

NAPT: Network Address/Port Translation

- **NAPT o masquerading** es un servicio de red que **permite la traducción de direcciones IP privadas en direcciones IP públicas y viceversa**
- Su principal objetivo es permitir que varios dispositivos en una red privada puedan acceder a Internet utilizando una sola dirección IP pública
- **Funcionamiento:** Cuando un dispositivo en una red privada quiere comunicarse con Internet:
 - El router cambia la dirección IP privada a la IP pública del router
 - Modifica el puerto de origen en la cabecera del paquete para diferenciar las conexiones
 - Mantiene una tabla de asignación para recordar qué puerto corresponde a qué dispositivo
 - Cuando la respuesta llega desde Internet, el router revisa la tabla y reenvía el paquete al dispositivo correcto en la red privada
- **Ventajas**
 - **Ahorro de direcciones IP:** Reduce la cantidad de IP públicas necesarias
 - **Mayor seguridad:** Oculta las direcciones IP internas de los dispositivos
 - **Facilita el acceso a Internet para redes privadas**

NAPT: Network Address/Port Translation

- Funcionamiento:
 - La única **IP pública** disponible es la **IP pública del Router**
 - El nº puerto cliente de la máquina origen **se traduce a un puerto libre del Router**

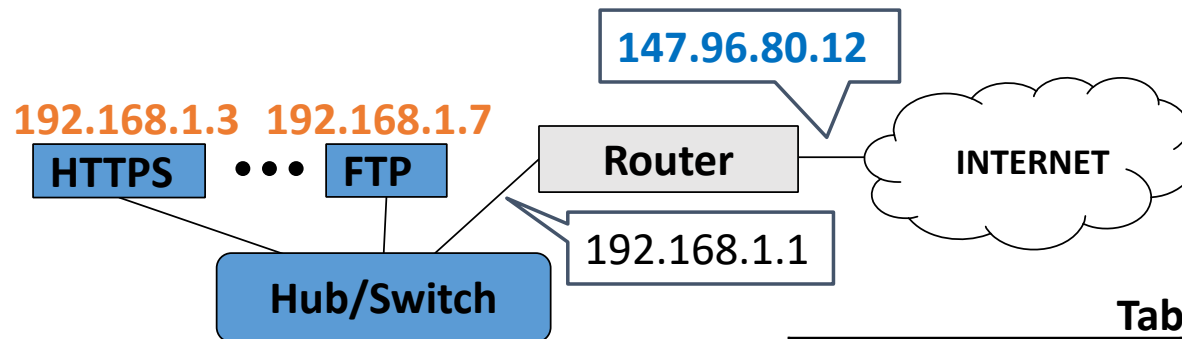


NAT: Network Address Translation

Port forwarding o virtual servers

- Permite tener servidores en la red privada “visibles” desde Internet
- **Funcionamiento:**
 - Desde Internet, todos los servidores usan la misma IP pública (la del Router)
 - El Router redirecciona los paquetes al servidor real de la red interna

Ejemplo:



Puertos por defecto

Puerto HTTPS: 443

Puerto HTTP: 80

Puerto FTP: 20 Para Transferencia de datos

Puerto FTP: 21 Para Conexión control

Tabla NAT

IP Privada	IP Pública
192.168.1.3: 443	147.96.80.12: 443
192.168.1.7: 20	147.96.80.12: 20
192.168.1.7: 21	147.96.80.12: 21

Servicios básicos de red: DNS

DNS: Sistema de Dominio de Nombres

- Cuando queremos comunicarnos con otra máquina **todo empieza por conocer su dirección IP**
 - Dirección IPv4 (32 bits): 192.168.1.10
- Estas direcciones son difíciles de manejar para los humanos de forma que en internet existe un sistema de traducción de nombres para identificar clientes y servidores con nombres amigables



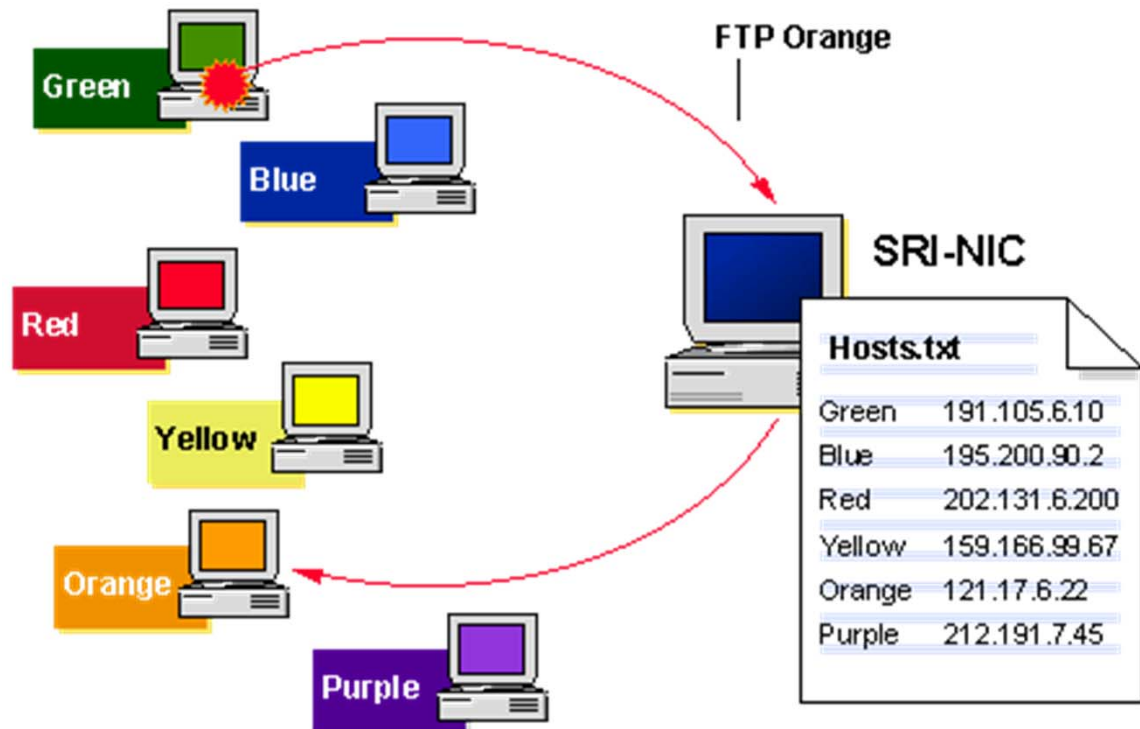
El primer paso es conocer:

- ✓ Puerto destino: **son conocidos**
 - P=80 para servidor web
 - P=53 para servidor DNS
 - ...
- ✓ IP destino: **¿?**

- **Propósito**
 - **Ofrecer un sistema para referenciar hosts de forma simbólica** (más fácil de recordar)
 - Traduce nombres simbólicos a direcciones IP
 - Ejemplo: `www.ucm.es` → `147.96.1.15`

DNS: Sistema de Dominio de Nombres

- Originalmente la traducción consistía en un fichero de texto (Host.txt) centralizado
 - Se distribuía periódicamente
- Inconvenientes:
 - Colisiones de nombres
 - Escalabilidad



¿Qué es un servidor DNS?

- Surgió como una alternativa distribuida y escalable, **para gestionar la administración y la traducción de nombres**

Problema

Colisiones de nombres —————→

Escalabilidad —————→

Solución

Espacio de dominios de nombres

- ✓ Asociar a cada recurso un nombre **único**, en formato alfabético y con jerarquía

Mapeo entre nombre – IP

- ✓ Asociar una dirección IP y un nombre de forma unívoca
- ✓ **Base de datos**
 - Distribuida**
 - Jerárquica** (estructura de árbol)
 - Escalable**
 - Información almacenada en registros RR (Resource Records)

¿Qué es un servidor DNS?

- Es un **conjunto de programas y protocolos** que:
 - Utiliza una **arquitectura cliente servidor**
 - Se encarga fundamentalmente de **traducir nombres simbólicos a direcciones IP**. Asocia nombres con IPs de:
 - Ordenadores
 - Servicios
 - ... cualquier recurso conectado a Internet
 - A la asociación nombre del recurso-IP, se le llama **Resolución DNS**



Elementos de DNS

- **Espacio de dominios de nombres**

- Estructura jerárquica en árbol donde cada nodo contiene información de su dominio

- **Servidores de nombres (servidores DNS)**

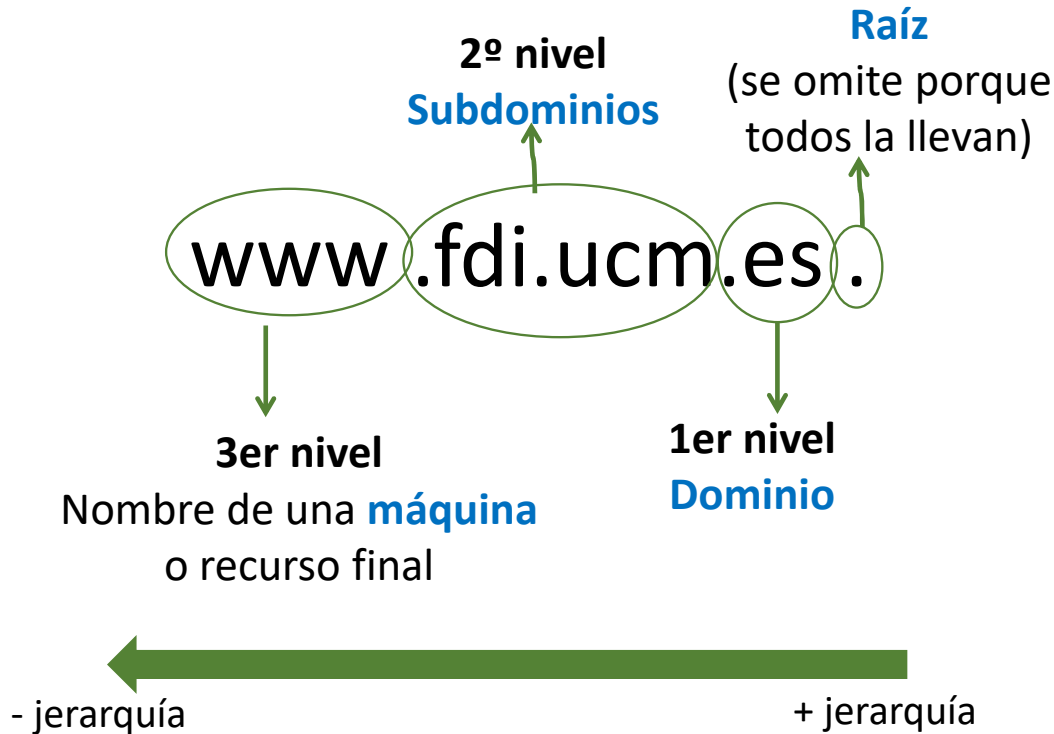
- Encargados de mantener y proporcionar información del espacio de dominios

- **Resolvers**

- Se encargan de generar las consultas y obtener la información solicitada y ofrecérsela al usuario
- Pueden ser **servidores caché o programas cliente**

Espacio de dominios de nombres

Nombre de dominio completamente cualificado (FQDN)

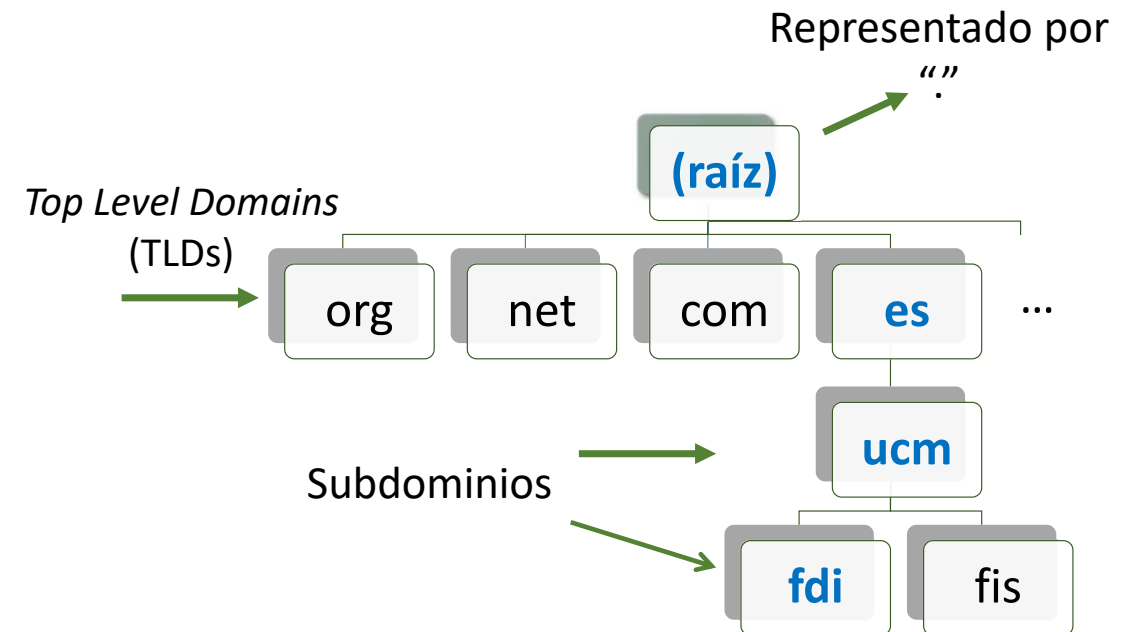


Restricciones en los nombres de dominios

- Puede ocupar un máximo de 255 caracteres (incluyendo los puntos)
- No hay límite en el número de subdominios de la jerarquía
- Cada nombre de dominio o subdominio, un máximo de 63 caracteres
- Los nombres solo pueden llevar caracteres alfanuméricos y guiones

Se almacena de forma Jerárquica

- Estructura de árbol
- Cada nodo representa un dominio



Espacio de dominios de nombres

Dominios de nivel superior (*Top Level Domains* (TLDs))

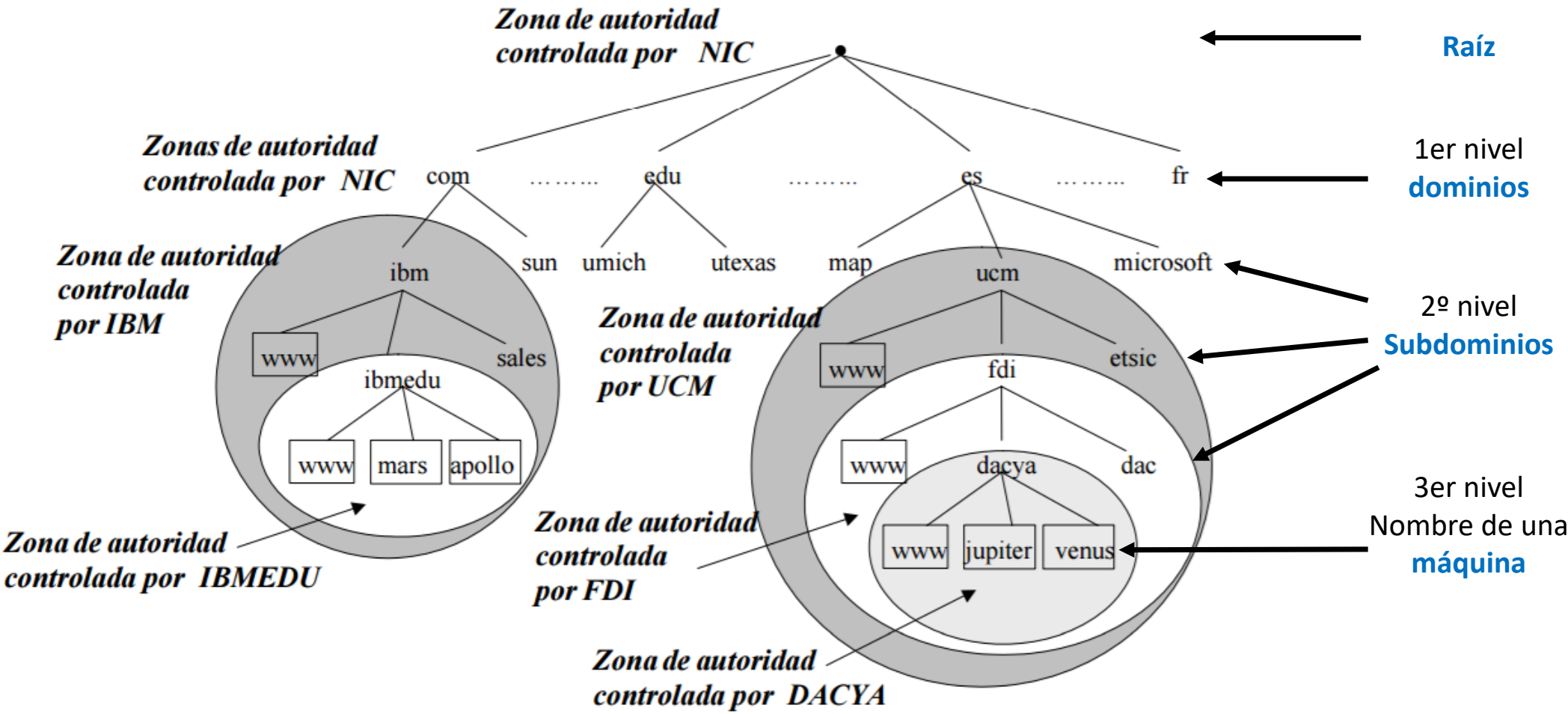
- **Ejemplos de dominios genéricos**

- **com**: organizaciones comerciales
- **edu**: organizaciones educativas (principalmente norteamericanas)
- **org**: organizaciones sin ánimo de lucro
- **net**: organizaciones relacionadas con Internet y servidores de acceso
- **gov**: instituciones gubernamentales norteamericanas
- **mil**: instituciones militares norteamericanas

- **Ejemplos de dominios de países**

- **es**: España
- **fr**: Francia
- **uk**: Reino Unido
- **it**: Italia
- **de**: Alemania
- **jp**: Japón
- **mx**: Méjico
- **ar**: Argentina

Servidores DNS y zonas de autoridad



Servidores DNS de la zona raíz

- La zona de autoridad raíz (.), está gestionada por el NIC (Network Information Center)
- Sólo Conocen** las IP de todos los servidores DNS de los dominios de nivel superior (com., org., net., es., fr., ...)

Se nombran con

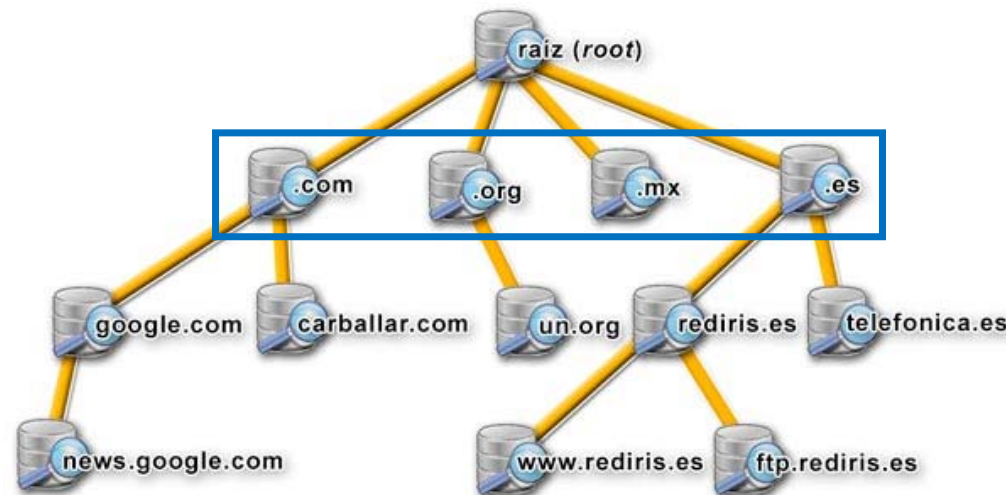
“letra”.root-servers.net



- Las direcciones de todos los servidores raíz se pueden encontrar en: <https://www.internic.net/domain/named.root>
 - Actualmente 13 repartidos por distintos países (por motivos de seguridad)

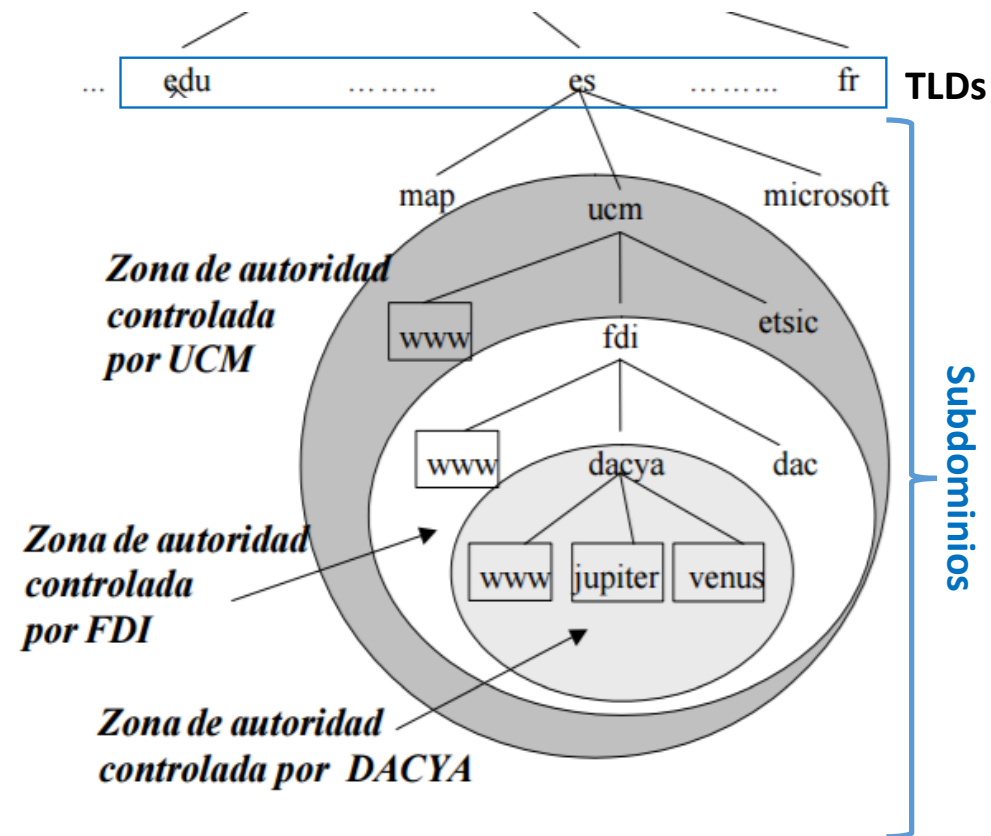
Zonas de autoridad de dominios de nivel superior

- Cada dominio de nivel superior (TLDs) es una zona de autoridad distinta
- Estas zonas **están gestionadas por el NIC** (Network Information Center)
- Cada zona cuenta con un **número variable de servidores DNS**
- **Conocen** las **IP** de todos los servidores DNS **de los subdominios que dependen directamente** de su zona de autoridad
- **No conocen** en detalle las IP del resto de máquinas de cada uno de los subdominios



Zonas de autoridad de subdominios

- Cada subdominio **puede estar dividido en una o varias zonas de autoridad**
- Subdominio con **una única zona de autoridad**
 - El servidor DNS del dominio deberá conocer en detalle los nombres y las IP de todas las máquinas del subdominio
- Subdominio con **varias zonas de autoridad**
 - El servidor DNS de mayor nivel conocerá:
 - Los nombres y las IP de las máquinas que dependen de él
 - La lista de servidores DNS raíz
 - Los servidores DNS de las zonas de autoridad independientes por debajo de él
 - No conocerá en detalle la organización de estas zonas de autoridad independientes
 - El servidor DNS de zonas de autoridad por debajo conocerá
 - Los nombres y las IP de las máquinas que dependen de él
 - La lista de servidores DNS raíz



Tipos de servidores DNS

- **Servidores primarios o maestros**

- Mantiene la **base de datos con la información sobre la zona**
- Los cambios sobre la información del dominio se llevan a cabo en el servidor primario

- **Servidores secundarios o esclavos**

- Poseen una **copia de la base de datos del servidor primario**
- Proporciona redundancia frente a fallos
- Permiten **equilibrar la carga de la red**, ya que **pueden resolver nombres igual que los servidores primarios**
- Periódicamente se sincronizan con el servidor primario para tener siempre la información actualizada

- **Servidores de sólo cacheo**

- No mantiene ninguna zona
- Sólo **almacena en su memoria temporal las consultas que recibe de los clientes**, para utilizarlas en caso de una nueva consulta.

Tipos de transacciones DNS

- **Consultas/respuestas DNS** (*Queries*)

- **Iterativas:** Si el recurso solicitado no se encuentra en el servidor DNS **devuelve un puntero al servidor autoritativo del dominio un nivel por debajo** del solicitado al que debe dirigirse (*Referral*)
- **Rekursivas:** El servidor **devuelve siempre la respuesta**, si no la tiene la busca. Cuando la consigue la guarda en cache

- **Transferencia de zona**

- Mecanismo de replicación de ficheros de zona (maestro a esclavo)

- **Actualizaciones dinámicas**

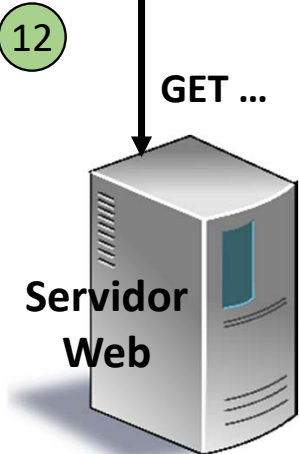
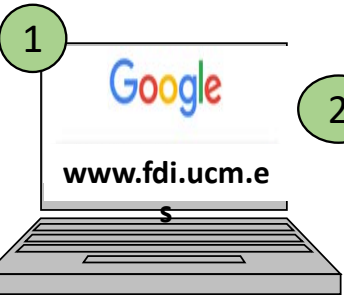
- Mecanismo utilizado para actualizar los ficheros de zona de un servidor DNS

- **Notificaciones**

- Transacciones que usa un servidor maestro para notificar cambios en su base de datos

Consulta DNS iterativa

Busca si ya lo tiene guardado por consultas anteriores



3 Busca si ya lo tiene almacenado en la cache

Resolver



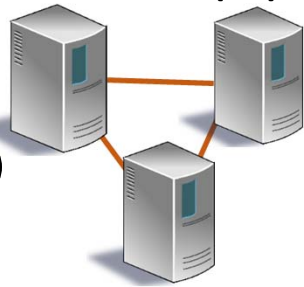
2 Pregunta DNS: ¿IP de www.ucm.es ?

11 www.ucm.es. IP=147.96.1.15

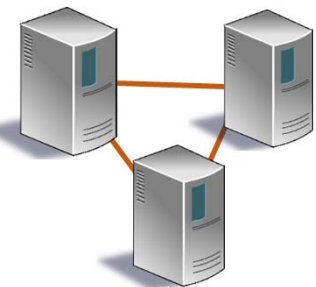
10 Añade información a cache

Servidores DNS

Zona raíz (.)



Zona TLD es.



Zona ucm.es.



Aquí está la información

4 ¿IP de www.ucm.es ?
5 Respuesta DNS: Consulta a servidor DNS del dominio es (10.1.2.3)

6 ¿IP de www.ucm.es ?
7 Respuesta DNS: Consulta a servidor DNS del dominio ucm.es (147.96.2.4)

8 ¿IP de www.ucm.es ?
9 www.ucm.com. IP= 147.96.1.15

Referral Response: informa dónde debe consultar para obtener la información que solicita

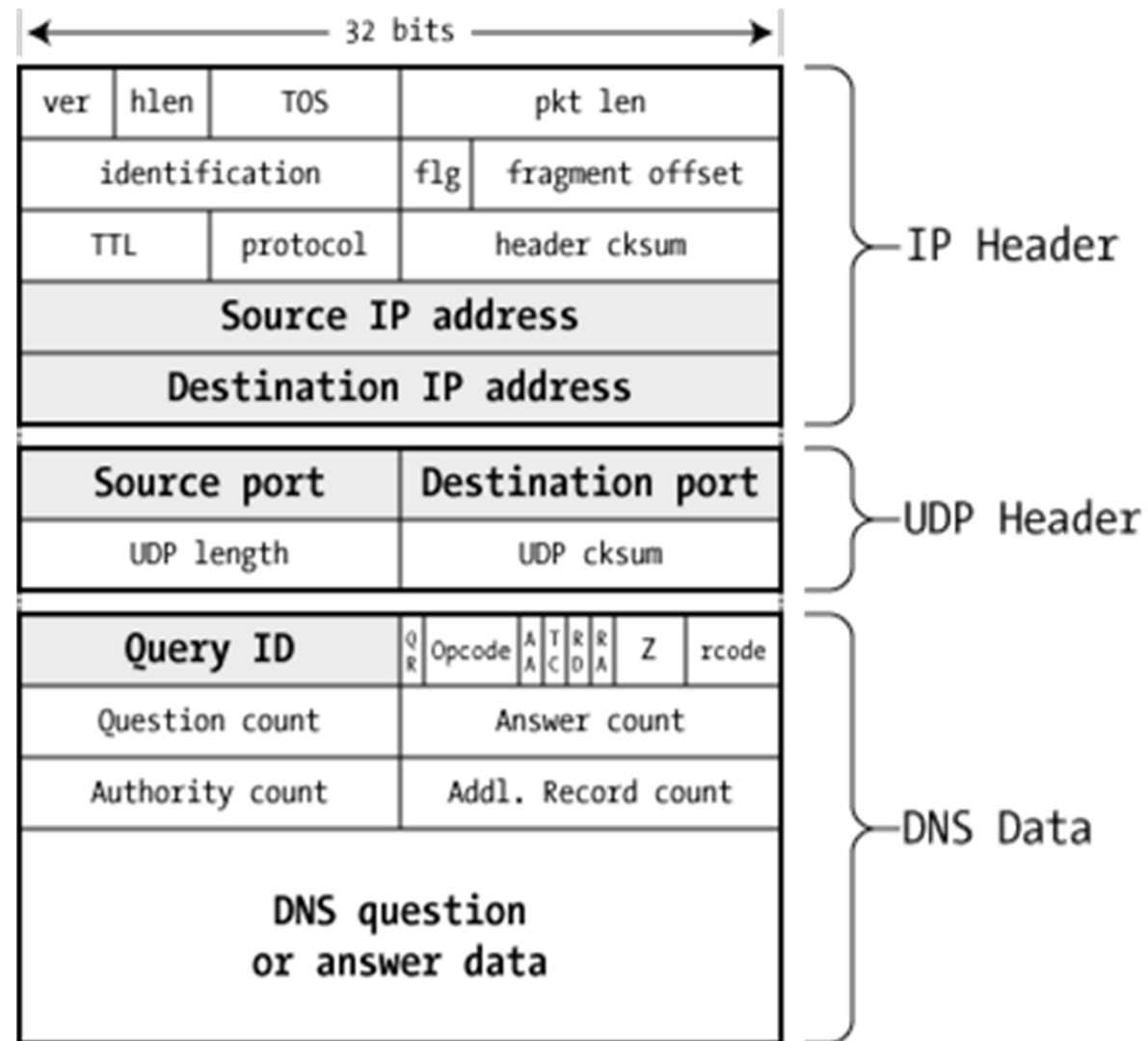
Mensaje DNS

- Usa principalmente **UDP** (puerto 53)

- Cada consulta consiste en una petición UDP del cliente seguida por una única respuesta UDP del servidor

- Se usa TCP cuando la respuesta excede de 512 bytes o para transferencias de zona

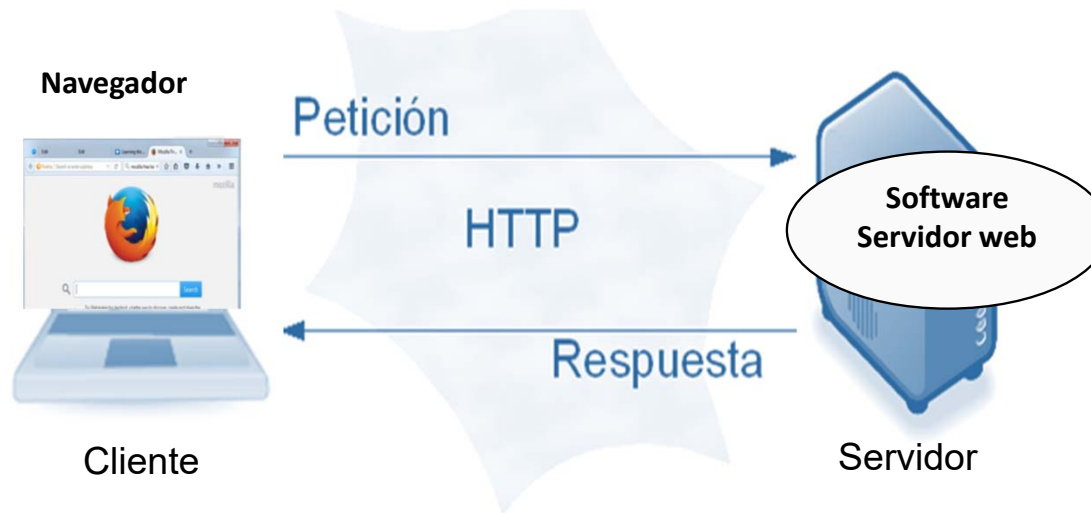
- Algunas implementaciones usan siempre TCP



Aplicaciones básicas: Protocolo HTTP

¿Qué es un servidor web?

- Es un **programa informático** que:
 - **Almacena** páginas web
 - **Acepta y gestiona peticiones** web de clientes
 - **Genera una respuesta** que envía al cliente
- Para la **transmisión de todos estos datos** se **usa el protocolo HTTP**
- El **código recibido** en el cliente es **interpretado y renderizado** en el **navegador web** para poder mostrarlo por pantalla
- **También** podemos referirnos con este nombre a la **máquina sobre la que corre este programa**



Servidor web y servidor de aplicaciones

Estos dos términos a veces se usan de forma indistinta, pero son diferentes

Un **servidor web** está diseñado para gestionar y servir **contenido web estático**

- **Contenido web estático**

- Es aquel que **no cambia ni se actualiza dinámicamente en función del usuario o del tiempo** y se entrega a los usuarios tal como **está almacenado en el servidor**
- Se compone de archivos HTML, CSS, JavaScript y multimedia (imágenes, videos, etc.) que el navegador muestra tal como están almacenados en el servidor
- **Características:**
 - **Fijo e inmutable:** El mismo contenido se muestra a todos los usuarios
 - **Rápido y eficiente:** Como no requiere procesamiento en el servidor, se carga rápidamente
 - **Fácil de alojar:** Se puede almacenar en servidores básicos o incluso en servicios de almacenamiento en la nube
 - **Menor complejidad:** No requiere bases de datos ni lenguajes de programación del lado del servidor
 - **Más difícil de actualizar:** Puede resultar difícil para usuarios no técnicos hacer pequeños ajustes en el contenido
- **Ejemplos:**
 - Páginas de presentación de empresas sin funciones interactivas
 - Folletos de pequeñas empresas, Páginas de eventos, Páginas de destino para campañas de marketing.
 - Portafolios personales o blogs con HTML y CSS sin backend
 - Documentación técnica en archivos HTML y Markdown

Servidor web y servidor de aplicaciones

Un **servidor de aplicaciones** gestiona **contenido web dinámico y aplicaciones web**

- **Contenido web dinámico**

- Es aquel que **cambia en función del usuario, el contexto o la interacción con el sitio web**
- Se **genera en tiempo real** mediante **tecnologías del lado del servidor** (como PHP, Python, Node.js) y **bases de datos**

- **Características:**

- Se usa en **sitios web con elementos interactivos** (no necesariamente en aplicaciones web)
- **Requiere procesamiento del servidor:** Necesita lenguajes de programación backend y bases de datos para funcionar
- En general, son más versátiles, pero requieren más recursos y configuración que los sitios estáticos
- El contenido web dinámico es una parte de muchas aplicaciones web, pero **no todas las páginas con contenido dinámico son aplicaciones web**

- **Ejemplos:**

- Noticias actualizadas automáticamente, comentarios en tiempo real
- Tiendas en línea donde los productos y precios cambian en función del usuario
- Paneles de administración donde los datos se actualizan en tiempo real

Servidor web y servidor de aplicaciones

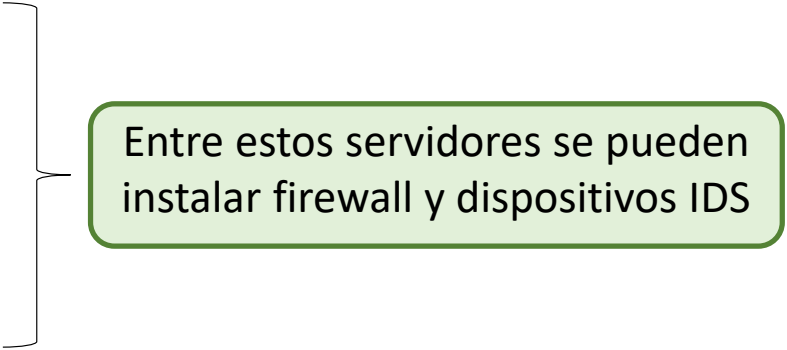
Aplicación web

- Programas que se ejecutan en un navegador web
- **Se almacena en un servidor de aplicaciones** y puede ser **accedida** por los usuarios **desde un navegador**
 - Interactúa con bases de datos y servidores de aplicaciones web
 - Se ejecuta en el lado del:
 - **Cliente** (frontend)
 - Es la interfaz que ve el usuario
 - Se construye con HTML, CSS y JavaScript (React, Angular, Vue.js)
 - **Servidor** (backend)
 - Procesa la lógica de negocio y maneja bases de datos
 - Se programa en PHP, Python, Node.js, Java, etc
- Características:
 - **Acceso desde cualquier lugar:** Solo necesitas un navegador y conexión a internet
 - **Interacción dinámica:** permiten interacción
 - por ejemplo, llenar formularios, subir archivos, ver contenido personalizado, etc
 - **Multiplataforma:** Funcionan en Windows, Mac, Linux, Android, iOS, etc
 - **Actualización centralizada:** No necesitas instalar nuevas versiones, todo se actualiza en el servidor
- Ejemplos:
 - Aplicaciones web como Google Drive, Gmail, Facebook, Twitter, Netflix, Youtube,



Servidor web y servidor de aplicaciones

- En muchos casos, ambos servidores trabajan juntos
- **Una arquitectura en tres capas** es preferible desde el punto de vista de la **seguridad**
 - Servidor web
 - Procesa peticiones HTTP y responde con contenido estático
 - Servidor de aplicaciones
 - Puede ejecutar código en lenguajes como Java, Python, PHP, o .NET
 - Genera contenido dinámico antes de enviarlo al servidor web
 - Permite el manejo de sesiones, autenticación y acceso a bases de datos
 - Servidor de base de datos



Entre estos servidores se pueden instalar firewall y dispositivos IDS

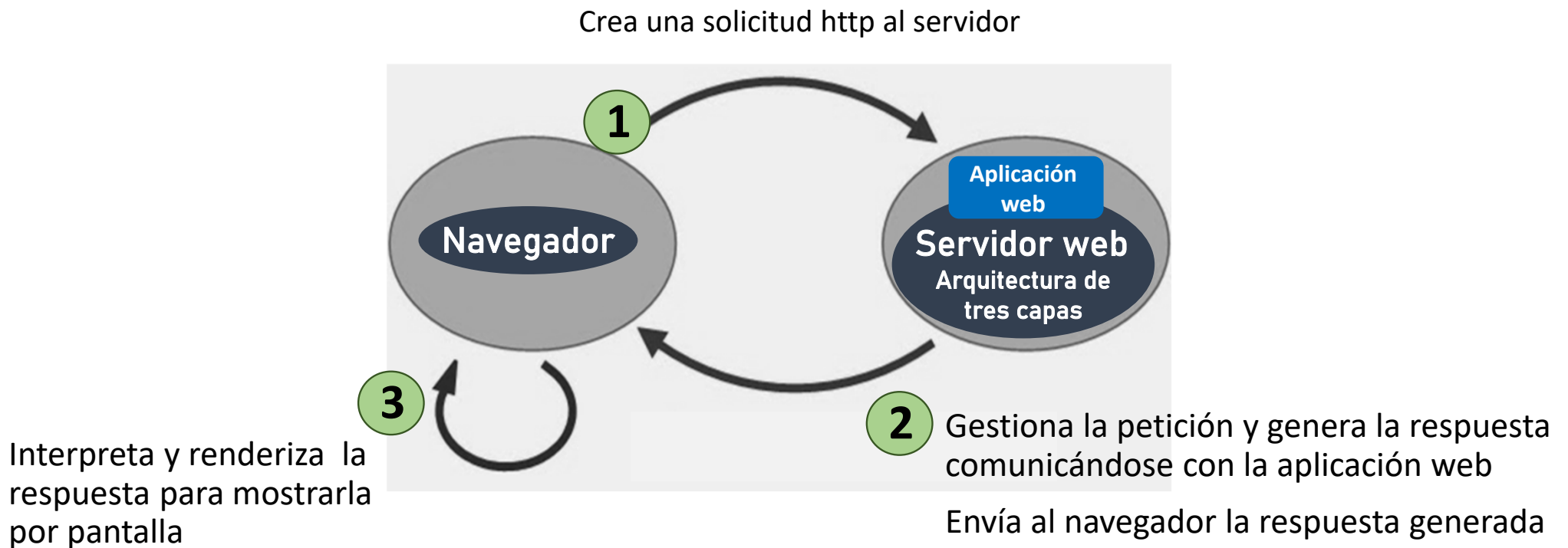
Servidor web y servidor de aplicaciones



- Si el cliente solicita **contenido estático** el **servidor web responde** directamente
- Si el cliente solicita **contenido dinámico**
 - Si ese contenido ha sido solicitado recientemente, lo tendrá guardado y responderá directamente
 - Si no, enviará la petición, de forma transparente al usuario, al **servidor de aplicaciones**, que hará las **consultas** necesarias al **servidor de bases de datos**

Servidor web y servidor de aplicaciones

- Cómo se accede a una aplicación web:

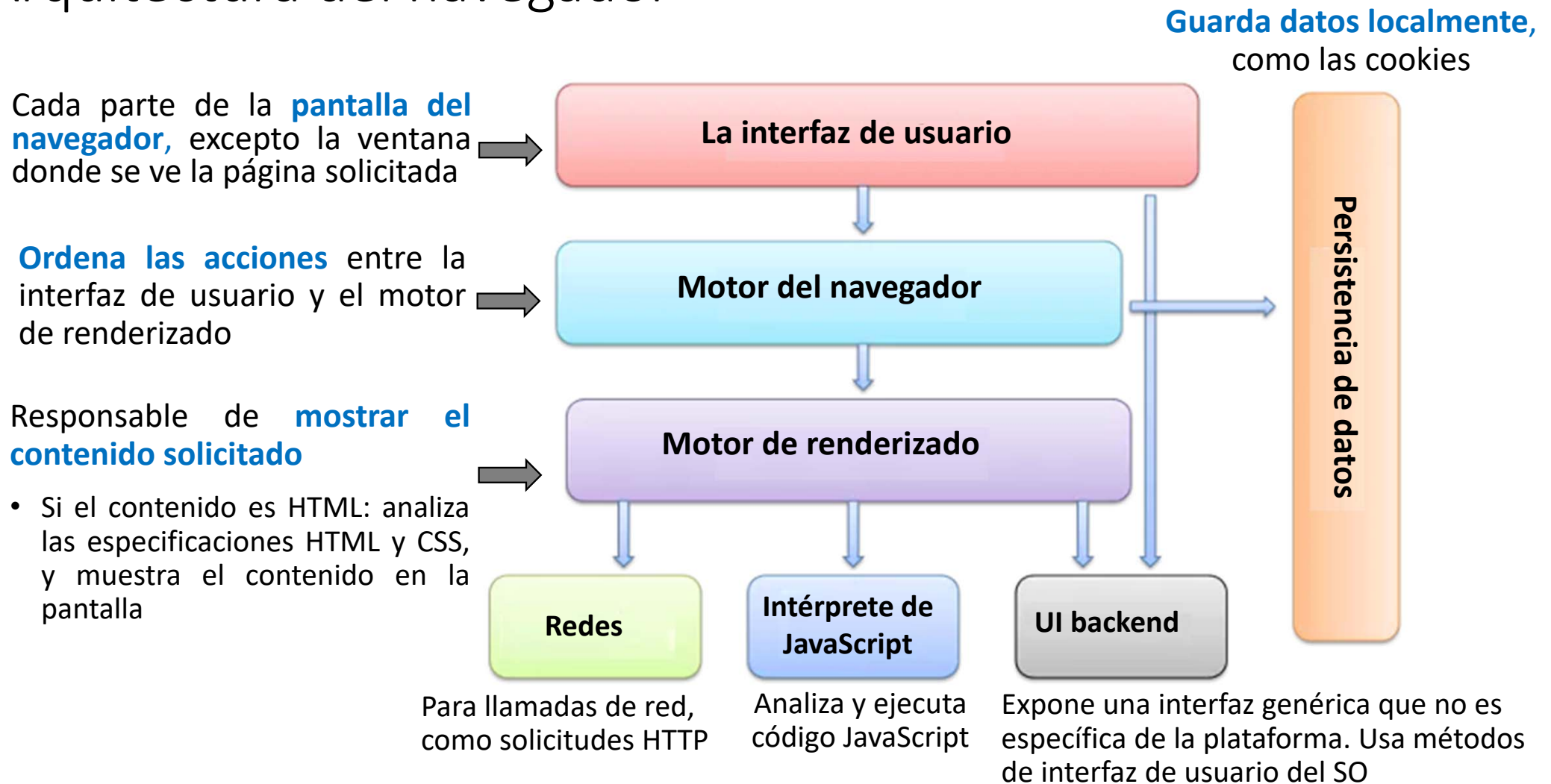


¿Qué es un navegador?

- Es un **software de cliente** que permite el **acceso a los contenidos web** almacenados en los servidores:
 - Realiza las **peticiones del contenido web** en nombre del cliente
 - **Interpreta y renderiza** el código recibido en respuesta a estas peticiones para poder mostrarlo por pantalla



Arquitectura del navegador



Fuente: <http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/>

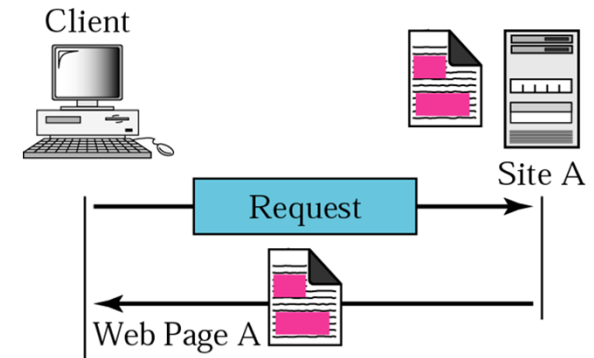
Protocolo HTTP

- **Protocolo de transferencia de hipertexto** (*Hypertext Transfer Protocol*)
 - Es el protocolo de comunicación que permite las transferencias de información en la World Wide Web
- **Define la sintaxis y la semántica** que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) **para comunicarse**
- Modelo **petición - respuesta**
 - El cliente (un navegador web) establece conexión con el **puerto 80 del servidor** y envía una solicitud HTTP a un recurso del servidor
 - El servidor Web recibe la solicitud, la procesa y responde con un mensaje HTTP de respuesta, que puede incluir una página HTML, imágenes, videos, etc
- Es un **protocolo sin estado**: no guarda ninguna información sobre conexiones anteriores
 - **Una vez se ha producido la solicitud y la respuesta, se cierra la conexión** (tanto cliente como servidor olvidan que ha existido esa conexión)
- El **intercambio de datos** se lleva a cabo en **texto en claro, sin cifrar**

HTTP: Transacciones

Funcionamiento:

- Cliente inicia conexión TCP al **puerto 80 (443 si es HTTPS)** del servidor
- Cliente pide un documento con un mensaje de petición http
- **El servidor envía el documento con un mensaje de respuesta http**
- El cliente cierra la conexión TCP



El cliente establece una conexión con el servidor
(SYN,SYN+ACK, ACK)

El cliente envía una
petición HTTP al
servidor

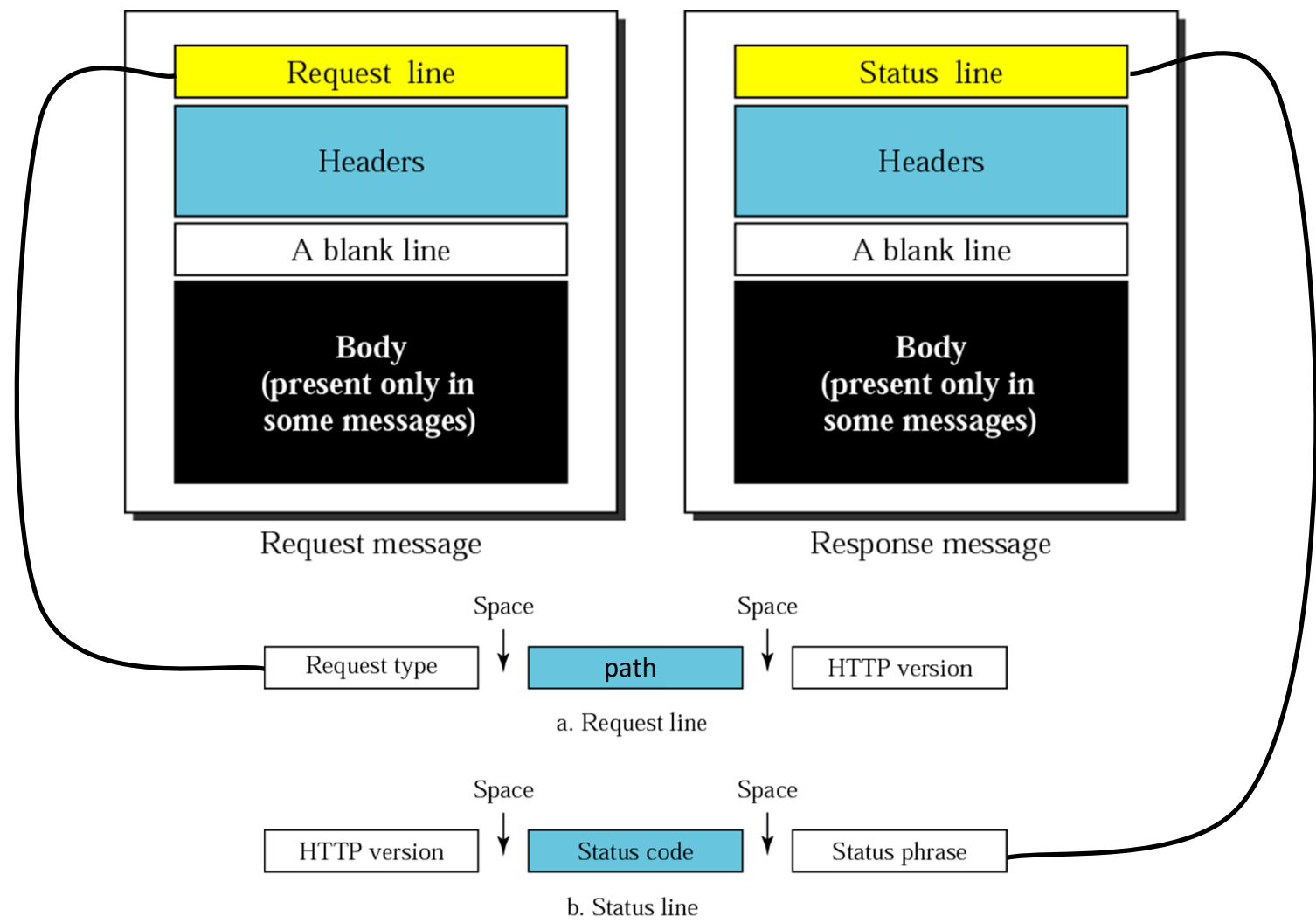
El servidor Web
procesa la solicitud
y **responde**

192.168.1.3	192.168.1.20	TCP	74 35216 → 80 [SYN] Seq=0
192.168.1.20	192.168.1.3	TCP	74 80 → 35216 [SYN, ACK]
192.168.1.3	192.168.1.20	TCP	66 35216 → 80 [ACK] Seq=1
192.168.1.3	192.168.1.20	HTTP	455 GET /mutillidae-master
192.168.1.20	192.168.1.3	TCP	66 80 → 35216 [ACK] Seq=1
192.168.1.20	192.168.1.3	HTTP	8992 HTTP/1.1 200 OK (text
192.168.1.3	192.168.1.20	TCP	66 35216 → 80 [ACK] Seq=3
192.168.1.3	192.168.1.20	TCP	66 35216 → 80 [FIN, ACK]
192.168.1.20	192.168.1.3	TCP	66 80 → 35216 [FIN, ACK]
192.168.1.3	192.168.1.20	TCP	66 35216 → 80 [ACK] Seq=3

El cliente y el servidor cierran
la conexión TCP (FIN,ACK)

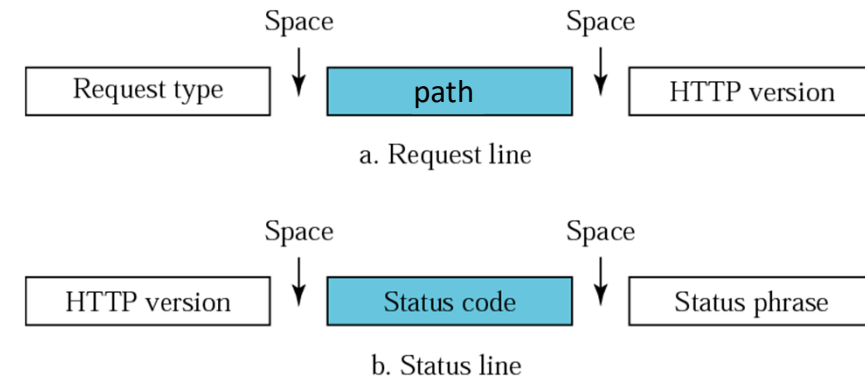
Si la conexión es persistente, el
cliente puede seguir enviando
peticiones

HTTP: Transacciones



HTTP: Transacciones

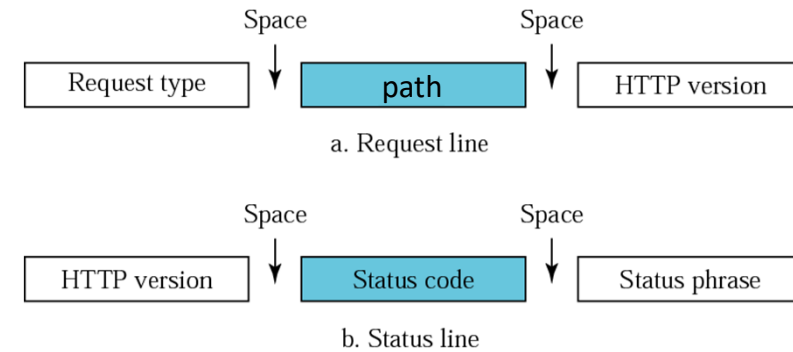
- Líneas de petición/estado y cabeceras en formato texto
- **Request type** : Tipos de mensajes de petición (métodos)
 - **GET**: Para pedir un documento al servidor
 - El servidor manda el documento en el cuerpo del mensaje
 - **HEAD**: Para pedir información sobre un documento
 - El mensaje de respuesta no contiene cuerpo, sólo cabecera
 - **POST**: Para enviar información desde el cliente al servidor
 - **PUT**: Para que el cliente proporcione un nuevo documento que debe ser almacenado en el servidor en la URL que se pasa
 - **COPY**: Para copiar un fichero de un path a otro en el servidor.
 - El origen está dado por la URL y el destino en la cabecera entity
 - **MOVE**: Para mover un fichero de un path a otro en el servidor.
 - El origen está dado por la URL y el destino en la cabecera entity
 - **DELETE**: Se utiliza para borrar un fichero del servidor
 - **OPTION**: Mensaje del cliente al servidor para pedir las opciones disponibles



HTTP: Transacciones

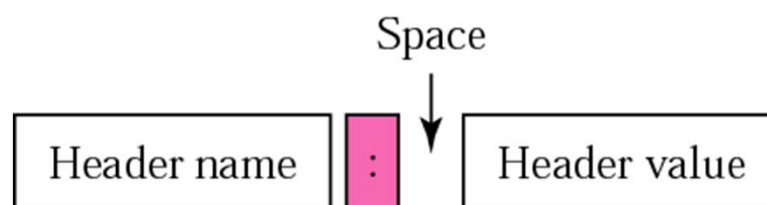
- **Status code y Status phrase**

- **Informativo:** el servidor ha recibido los encabezados de la petición, el cliente debe enviar el cuerpo del mensaje
 - 100 Continue
 - ...
- **Exito:** petición recibida correctamente, entendida y aceptada
 - 200 OK Respuesta estándar para peticiones correctas.
 - 202 Accepted
 - ...
- **Redirecciones:** el cliente tiene que tomar una acción más para completar la petición
 - 301 Moved Permanently
 -
- **Error:** La solicitud contiene sintaxis incorrecta o no puede procesarse
 - 400 Bad Request
 - 403 Forbidden
 - 404 Not Found
 - 405 Method Not Allowed
 - ...



HTTP: cabecera

- Sirven para el intercambio de información adicional entre cliente y servidor
 - Ejemplo: envíame el fichero comprimido con gzip
- Cada cabecera está constituida por una o más líneas de cabecera
 - Formato texto



- Las líneas de cabecera pueden ser de cuatro tipos
 - **general**: tanto en mensajes de petición como de respuesta
 - **de petición**: sólo en mensajes de petición
 - **de respuesta**: sólo en mensajes de respuesta
 - **entidad (entity)**: tanto en mensajes de petición como de respuesta

HTTP: cabecera petición

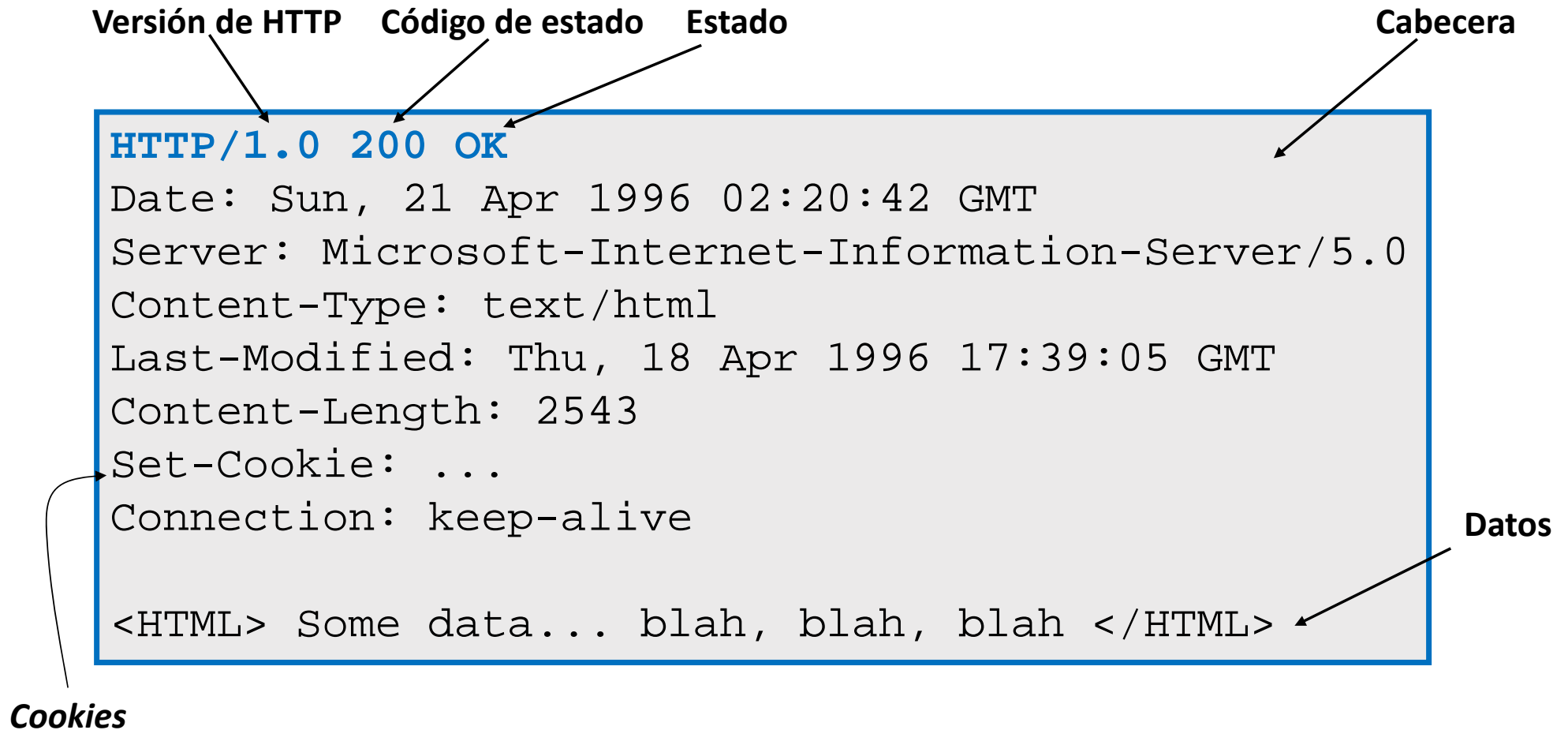


User-Agent: es el navegador del cliente

Referer : dirección de la pagina que nos ha llevado a la página actual

Connection: Keep-Alive indica conexión persistente

HTTP: cabecera respuesta



el servidor le envía las cookies y el cliente las almacena y las envía en posteriores peticiones POST

HTTP: URL

Protocolo **Nombre del host** **Recurso solicitado:
path y archivo**

`http://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos.html`

The diagram shows the URL `http://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos.html` with three main components identified by brackets and arrows: **Protocolo** (http://), **Nombre del host** (www.ccn-cert.cni.es), and **Recurso solicitado: path y archivo** (/informes/informes-ccn-cert-publicos.html).

`https://www.ccn-cert.cni.es/component/search?searchword=http&searchphrase=all&Itemid=784`

The diagram shows the URL `https://www.ccn-cert.cni.es/component/search?searchword=http&searchphrase=all&Itemid=784` with two main components identified by brackets and arrows: **path** (/component/search) and **query** (?searchword=http&searchphrase=all&Itemid=784).

La query siempre va precedida de ?

Protocolo HTTP

- HTTP es un protocolo sin estado, el servidor web no recuerda nada sobre conexiones previas, como consecuencia aparece:
 - La necesidad de **guardar el estado: cookies**
 - El concepto de **sesión**



SIGN IN

name

password

LOGIN

[forgot password?](#)

☐ **remeber me**

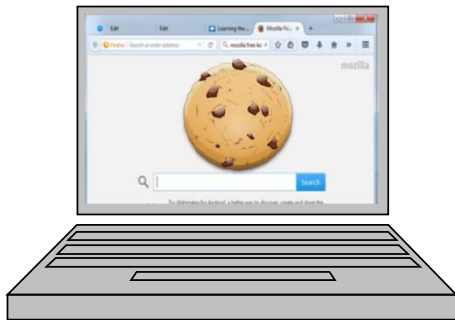
OR **REGISTER**

Cookies

■ Almacenan el estado en el cliente

- Una cookie HTTP es una pequeña porción de datos que el servidor envía al navegador del cliente para que la almacene y la envíe en cualquier http request futura a ese servidor
- Se usan principalmente para tres propósitos
 - **Gestión de sesiones** (login de usuario, carritos de la compra, ...)
 - Sirve para saber si dos solicitudes HTTP provienen del mismo navegador y en ese caso mantener al usuario identificado
 - **Personalización** (preferencias de usuario)
 - **Promocional** (analiza el comportamiento del usuario)
- **Cómo funcionan:**
 - El servidor envía una cabecera set-cookie en la respuesta a una solicitud HTTP
 - El navegador las almacena y las envía en cada solicitud a ese servidor dentro de la cabecera HTTP
 - Se puede limitar a dónde se envía la cookie, estableciendo restricciones a un dominio y ruta específicos
 - Se pueden crear con una fecha de expiración

Cookies



1

POST ...

2

Cabecera HTTP:

Set-cookie: NAME=VALUE ;

domain = (quién puede leer) ;

secure = (enviar solo sobre TLS);

expires = (cuándo expira) ;

Si expires=NULL dura solo esta sesión

3

POST ...

Cookie: NAME = VALUE

Servidor

