Tema 2. Nivel de aplicación

Introducción a las redes de ordenadores

Boni García

Curso 2017/2018

Índice de contenidos

Índice de contenidos

1. Introducción al nivel de aplicación

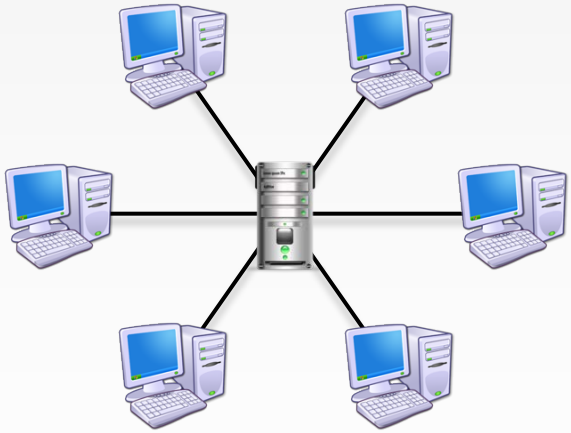
Servicios distribuidos

- Un **servicio distribuido** o **servicio telemático** es una aplicación distribuida, que consiste en varios procesos que se ejecutan en diferentes equipos terminales y que se comunican a través de una red de datos
- Características principales:
  - **Heterogeneidad** (elementos que componen un servicios distribuido pueden ser muy variados y diferentes)
  - Van a ocurrir situaciones de **concurrencia** (acceso simultáneo)
  - La red introduce **latencia** (tiempo de respuesta entre los componentes del servicio distribuido)

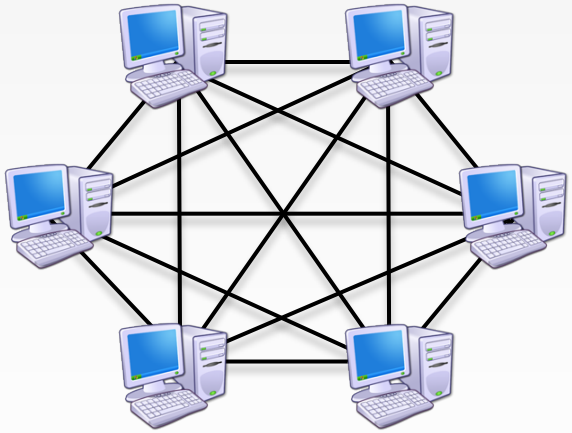
1. Introducción al nivel de aplicación

Modelos arquitectónicos de red

- El modelo arquitectónico identifica las funciones de los elementos individuales del sistema distribuido
- Tipos:



Cliente-servidor



Peer-to-peer (P2P)

1. Introducción al nivel de aplicación

Modelos arquitectónicos de red

- En el modelo **cliente-servidor**, los clientes son los elementos que necesitan servicios del recurso y el servidor es la entidad que poseen el recurso. Ejemplos de servicios distribuidos cliente-servidor:
  - Web: protocolo HTTP
  - Correo electrónico: protocolo de envío de correo electrónico (SMTP) y de recepción (POP, IMAP)
  - Transferencia de ficheros: protocolo FTP, SFTP
  - Resolución de nombres de dominio: protocolo DNS
  - Acceso a máquinas remotas: protocolos Telnet, SSH
- En el modelo **P2P** todos los equipos interactúan cooperativamente como iguales

Índice de contenidos

2. La Web

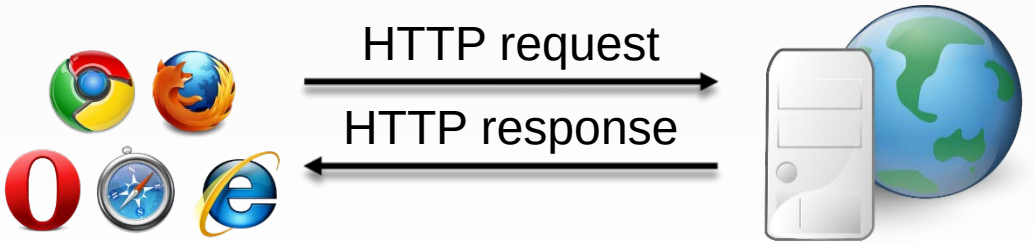
¿Qué es la Web?

- La **Web** (*World Wide Web*) es un servicio de distribución de contenidos hipertexto accesibles vía Internet
- Como ya sabemos, **Internet** es un conjunto descentralizado de redes de datos interconectadas que utilizan la familia de protocolos **TCP/IP** que interconecta cientos de millones de dispositivos (hosts o sistemas terminales) en todo el mundo
- Por lo tanto, no debemos confundir la Web con Internet
- Las **páginas web** son documentos escritos en lenguaje **HTML** (*HyperText Markup Language)* e interconectados a través de enlaces (*links*)

2. La Web

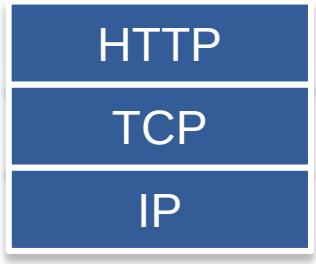
¿Qué es la Web?

- La Web está basada en un modelo **cliente-servidor**
- El protocolo de nivel de aplicación para comunicar clientes y servidores en la Web es **HTTP** (*HyperText Transfer Protocol*)
- Los **recursos web** se identifican con un nombre único llamado dirección web o **URL** (*Uniform Resource Locator*)
- Puerto por defecto para los servidores web: 80



Cliente = navegador

Servidor = servidor web



2. La Web

¿Qué es la Web?

- Un servidor web envía por HTTP los ficheros que tiene almacenados en su disco duro a los clientes que lo solicitan
- Puede servir cualquier tipo de fichero, aunque lo habitual son los ficheros que un navegador reconoce (html, jpg, png, pdf…)
- Cuando recibe una petición, devuelve el fichero del disco duro que se ajuste a la ruta indicada en la **URL**

http://www.miservidor.com:puerto/ruta/del/fichero/fichero.txt?clave=valor#fragmento

Esquema (protocolo)	Nombre del servidor	Puerto del servidor	Ruta del recurso	Nombre del recurso	Consulta	Ancla

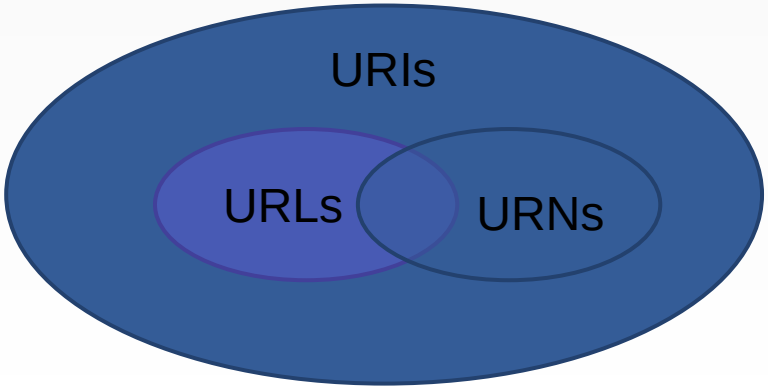
Dependiendo del servidor, las URLs pueden ser sensibles a mayúsculas (*case sensitive*). Ejemplo:
https://en.wikipedia.org/wiki/Iron_Maiden -- https://en.wikipedia.org/wiki/Iron_maiden

2. La Web

¿Qué es la Web?

- Las URLs son parte de un conjunto más grande llamado **URI** (*Uniform Resource Identifier*)
  - Las URLs (*Uniform Resource Locator*) son cadenas que sirven para localizar un recurso. Ejemplos URLs:
    - http://www.ietf.org/rfc/rfc2396.txt
    - mailto:john.doe@example.com
  - Las URNs (*Uniform Resource Name*) son cadenas que sirven para nombrar un recurso. Ejemplos URNs:
    - urn:ietf:rfc:2648
    - urn:issn:0167-6423y

http://www.w3.org/TR/uri-clarification/

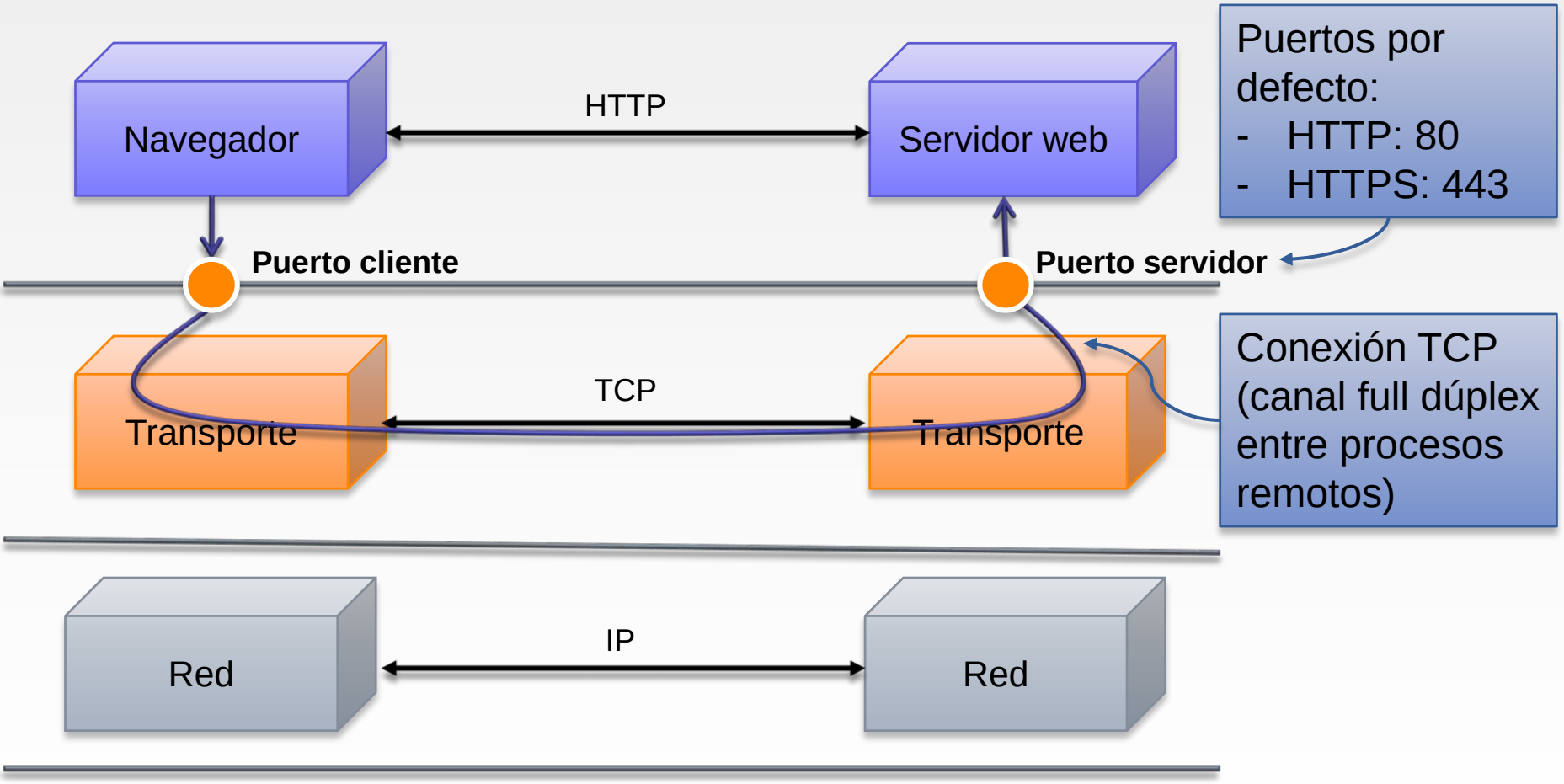


2. La Web

¿Qué es la Web?

- Navegadores más usados:
  - Google Chrome
  - Firefox
  - Internet Explorer → Edge
  - Safari
  - Opera
- Servidores web más usados:
  - Apache
  - Internet Information Server (IIS)

2. La Web


Navegador
HTTP
Servidor web
Puertos por defecto:
- HTTP: 80
- HTTPS: 443
Puerto cliente
Puerto servidor
Transporte
TCP
Transporte
Conexión TCP (canal full dúplex entre procesos remotos)
Red
IP
Red

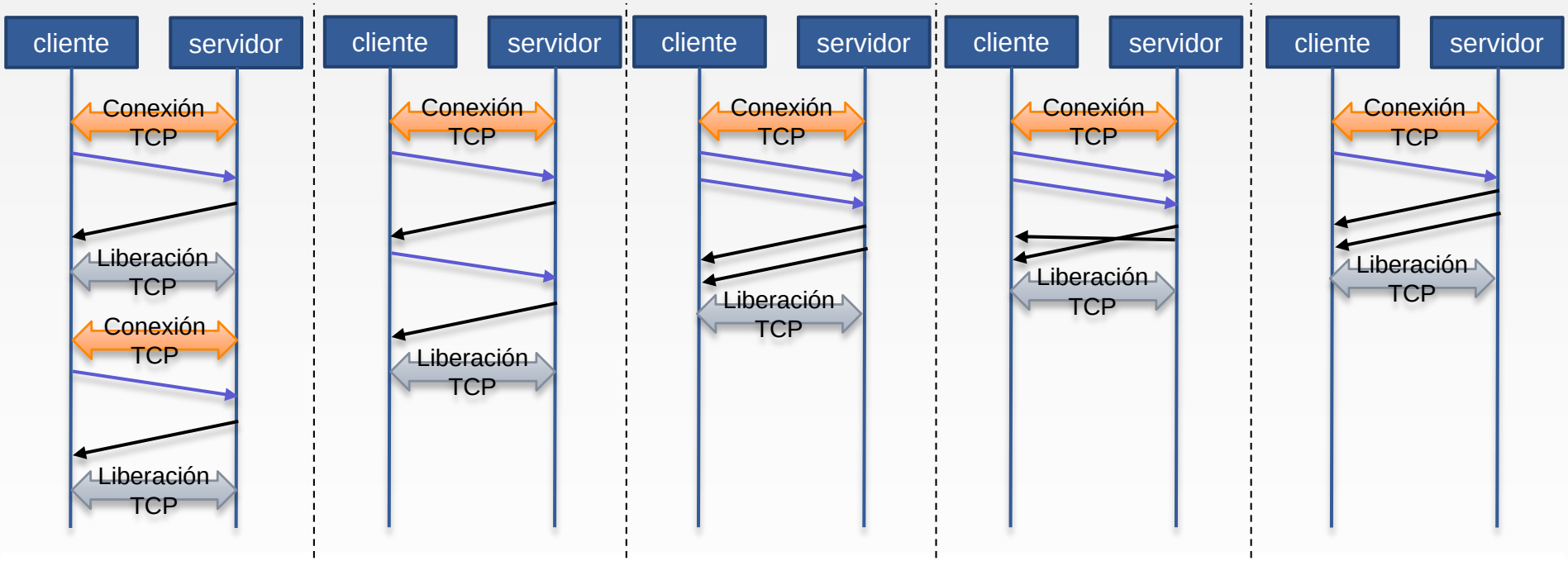
2. La Web

Versiones de HTTP

- HTTP/0.9 → Versión original. Actualmente obsoleta
- HTTP/1.0 (RFC 1945) → Versión antigua pero todavía se usa
  - Las conexiones son no-persistentes (se utilizarán múltiples conexiones TCP, una por cada objeto solicitado)
- HTTP/1.1 (RFC 2616) → Versión actual
  - Las conexiones son persistentes (el servidor mantiene abierta una conexión TCP para que las siguientes peticiones y respuestas se transmitan por esa conexión)
  - Permite peticiones sucesivas (*pipelining*), es decir, hacer varias peticiones al servidor sin esperar a la respuesta
- HTTP/2.0 (RFC 7540) → Soportado en navegadores desde 2015
  - Las respuestas se pueden procesar asíncronamente (*multiplexing*)
  - El servidor puede mandar recursos al cliente antes de exista petición (*push*)

2. La Web

Versiones de HTTP



HTTP/1.0 conexiones no permanentes

HTTP/1.1 conexiones permanentes

HTTP/1.1 *pipelining*

HTTP/2.0 *multiplexing*

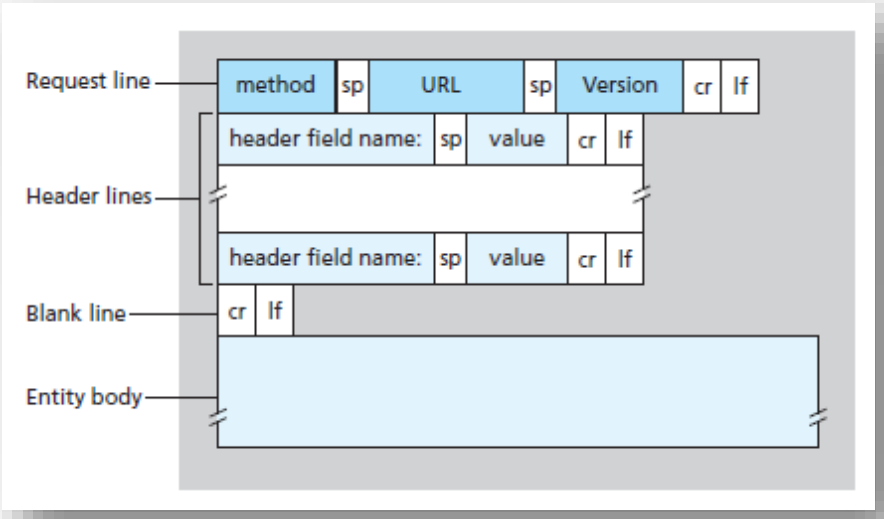
HTTP/2.0 *push*

2. La Web

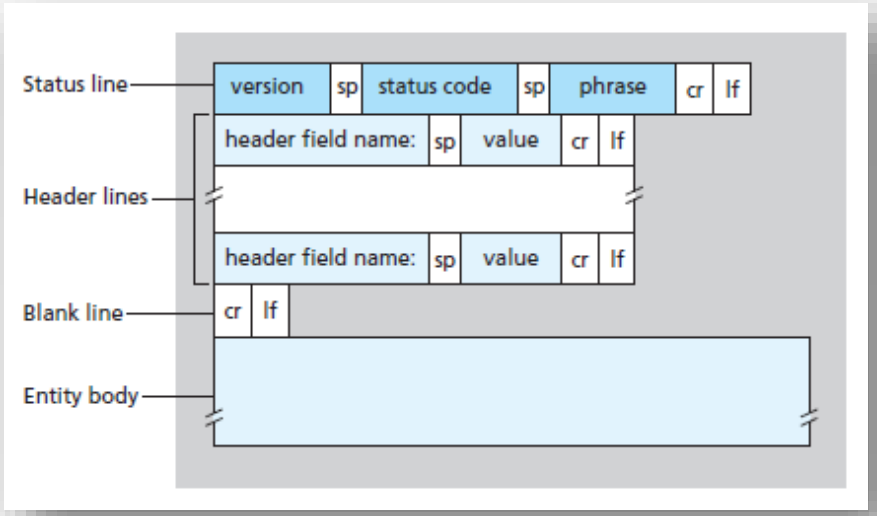
HTTP 1.1: Tipos de mensajes

- Hay dos tipos de mensajes:

sp=espacio
cr=retorno de carro
lf=salto de línea



peticiones (*request*)

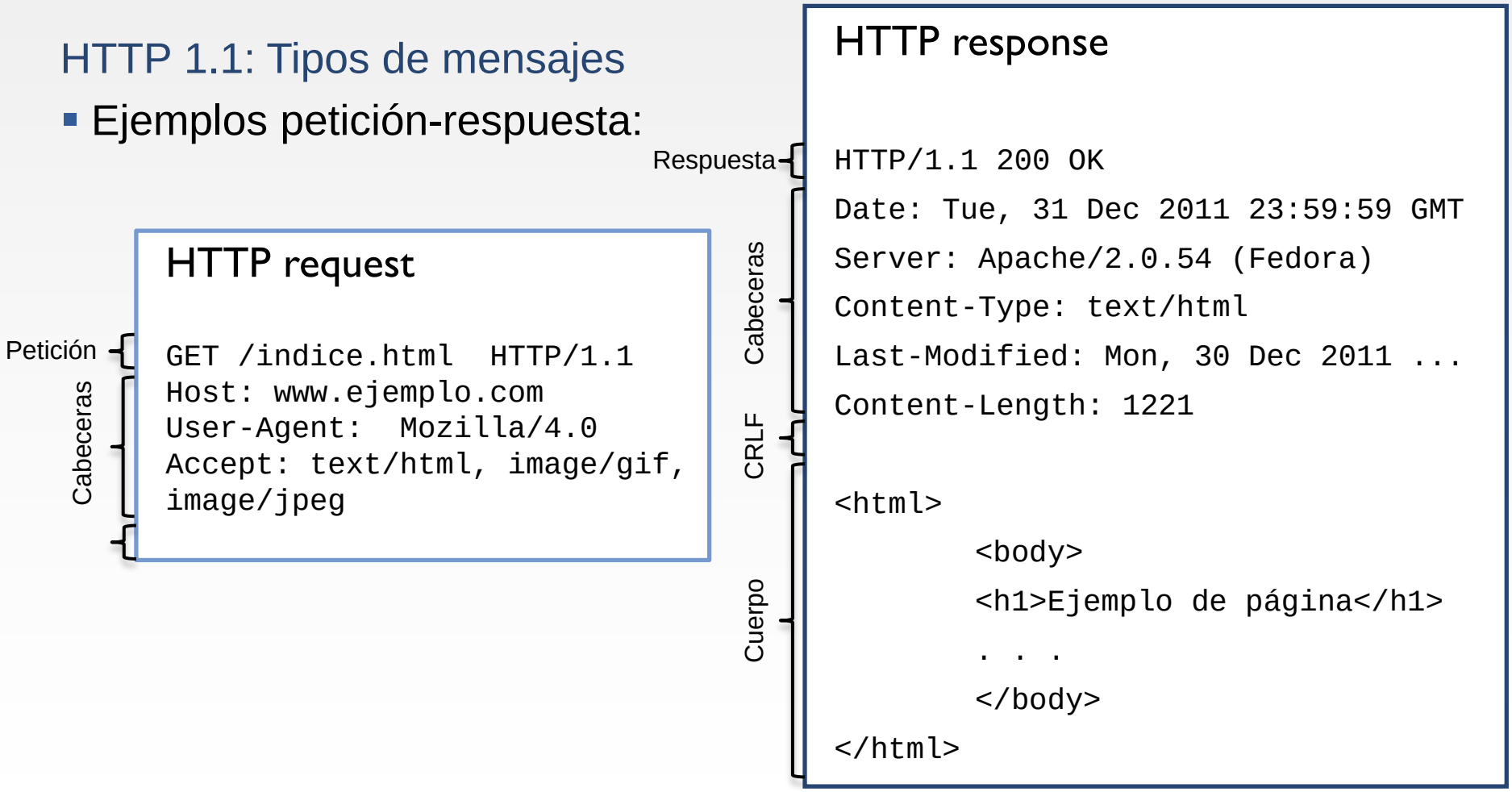


respuestas (*response*)

2. La Web

HTTP 1.1: Tipos de mensajes

- Ejemplos petición-respuesta:



2. La Web

HTTP 1.1: Métodos

- Los métodos en HTTP (algunas veces referido como “verbos”) indican la acción que desea que se efectúe sobre el recurso identificado
- HTTP 1.1 (RFC 2616) define 8 métodos:

- Hay una extensión a HTTP 1.1 (RFC 5789) que define un nuevo método:

2. La Web

HTTP 1.1: Métodos

Usados en servicios REST

- **GET**: Petición de un recurso determinado (URL)
- **POST**: Envío de datos que serán procesados por un recurso (URL)

Métodos principales

- PUT: Crea un recurso
- DELETE: Borra un recurso

Métodos normalmente deshabilitados en los servidores web

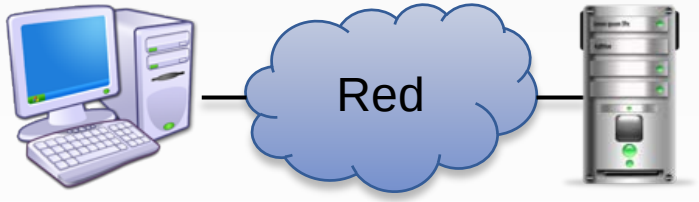
- PATCH: Solicita al servidor la modificación parcial de un recurso

- HEAD: Pide una respuesta idéntica a la que correspondería a una petición GET, pero sin el cuerpo de la respuesta. Esto es útil para conocer las cabeceras de la respuesta pero sin transportar todo el contenido
- TRACE: Solicita al servidor que envíe de vuelta en un mensaje de respuesta con la petición enviada (servicio de *echo*). Se utiliza con fines de comprobación y diagnóstico
- OPTIONS: Solicita al servidor los métodos admitidos para un determinado recurso. La respuesta se obtiene en la cabecera `Allow:`
- CONNECT: Se utiliza para indicar a un proxy web que establezca una conexión segura (TLS) con una máquina remota

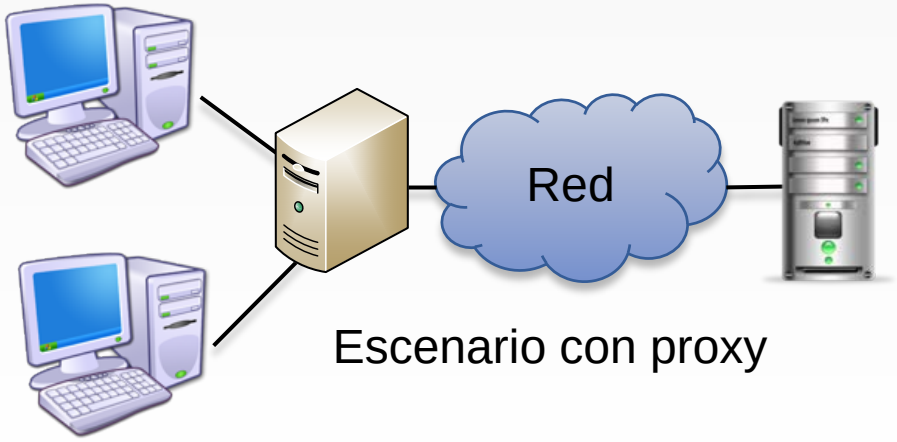
2. La Web

HTTP 1.1: Proxy web

- En redes, un *proxy* es un servidor que intercepta las conexiones de red hechas desde nodos clientes a otros servidores
- Principales razones del uso de proxy: monitorización, filtrado, mejora de rendimiento (caché)
- Un *proxy web* es un tipo de proxy que intercepta el tráfico HTTP



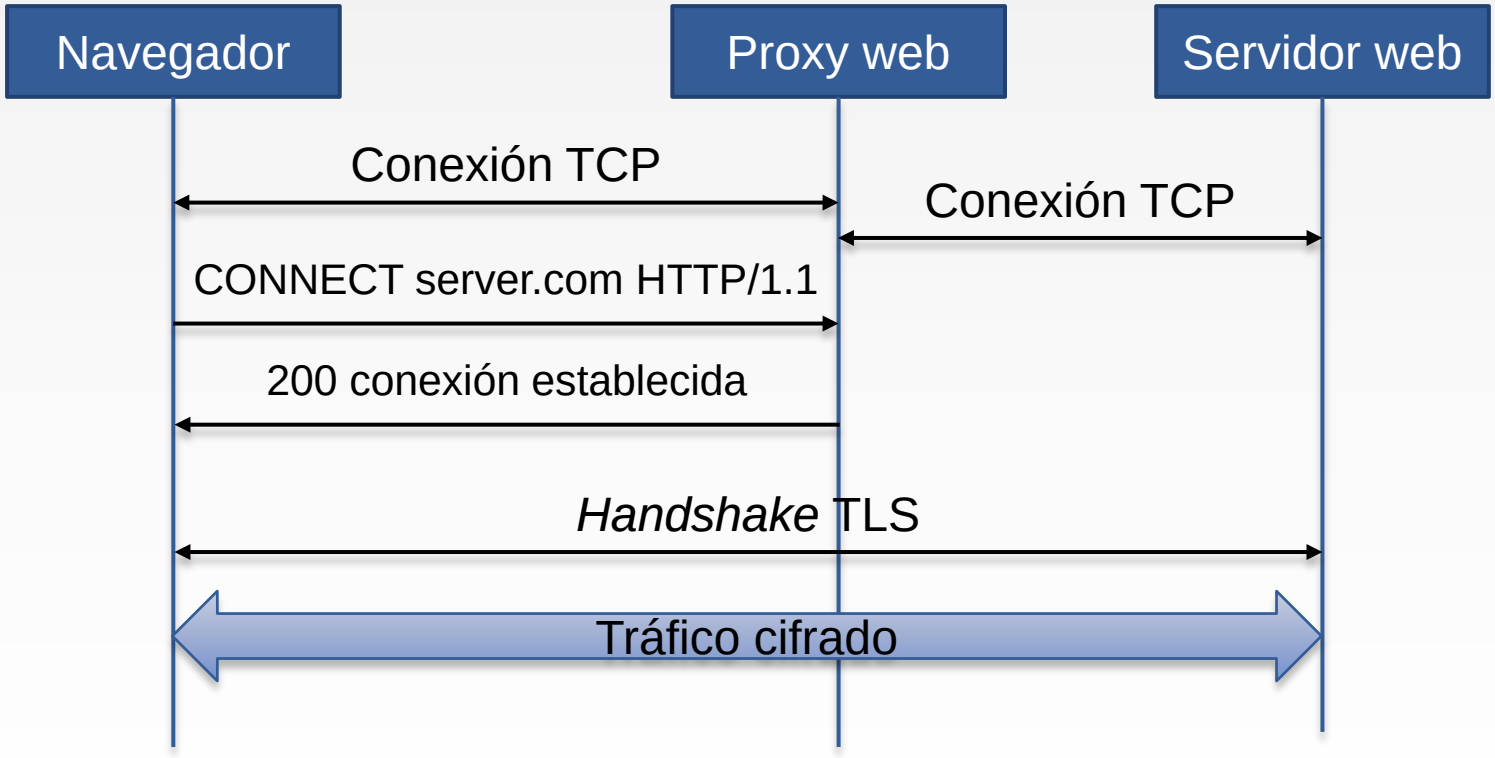
Escenario sin proxy



Escenario con proxy

2. La Web

HTTP 1.1: Proxy web



2. La Web

HTTP 1.1: Códigos de respuesta

- 1xx Respuesta informativa
  - 101 Cambio de protocolo
- 2xx Operación exitosa. Por ejemplo:
  - 200 OK
- 3xx Redirección. Por ejemplo:
  - 302 Movido temporalmente
  - 304 No modificado (usado como respuesta en un GET condicional → recurso en caché)
- 4xx Error por parte del cliente. Por ejemplo:
  - 404 No encontrado
- 5xx Error del servidor. Por ejemplo:
  - 500 Error interno

2. La Web

HTTP 1.1: Cabeceras

- Algunas de las cabeceras más comunes en las **peticiones**:

Cabecera	Descripción	Ejemplo
`Accept`	Determina el tipo de contenido o MIME	`Accept: text/plain`
`Accept-Charset`	Juego de caracteres de caracteres aceptable en la respuesta	`Accept-Charset: utf-8`
`Host`*	Nombre de dominio del servidor (permite *virtual hosting*)	`Host: en.wikipedia.org`
`If-Modified-Since`	*Get condicional*. Se usa para saber si un recurso ha cambiado desde una fecha determinada	`If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT`
`User-Agent`	Cadena que identifica al cliente	`User-Agent: Mozilla/5.0 …`

*cabecera obligatoria

2. La Web

HTTP 1.1: Cabeceras

- Algunas de las cabeceras más comunes en las **respuestas**:

Cabecera	Descripción	Ejemplo
`Date`	Fecha en que fue mandada la respuesta	`Date: Tue, 15 Nov 1994 08:12:31 GMT`
`Server`	Tipo de servidor	`Server: Apache/2.4.1 (Unix)`
`Content-Type`	Tipo MIME del cuerpo de respuesta	`text/html; charset=UTF-8`
`Content-Length`	Tamaño del cuerpo en bytes	`Content-Length: 348`
`Last-Modified`	Respuesta al *GET condicional*. Fecha en la que fue modificado el recurso	`Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT`
`Allow`	Métodos válidos para un recurso	`Allow: GET, HEAD`

2. La Web

HTTP 1.1: Tipos MIME

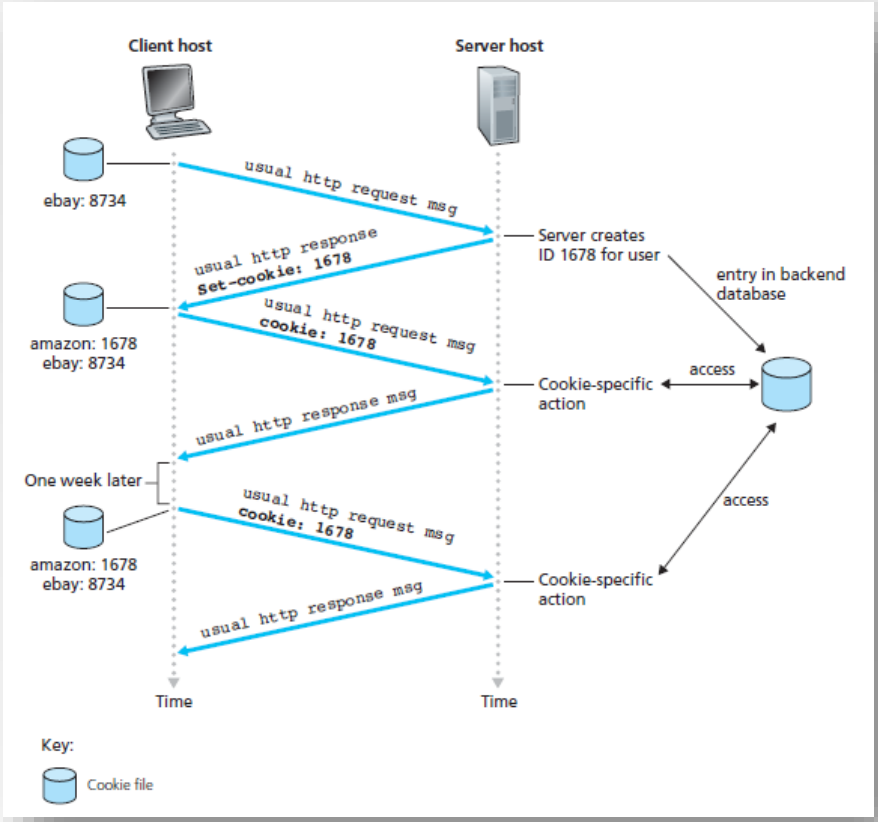
- MIME (*Multipurpose Internet Mail Extensions*) indica el tipo de medio que representa el contenido del mensaje

Tipo	Extensión(es)	Descripción
`text/plain`	`.txt`	Texto plano
`text/html`	`.html .htm`	Página web
`image/jpeg`	`.jpg .jpeg`	Imagen JPEG
`image/gif`	`.gif`	Imagen GIF
`image/png`	`.png`	Imagen PNG
`application/pdf`	`.pdf`	Archivo PDF
`audio/mpeg3`	`.mp3`	Audio en formato MP3
`video/mpeg`	`.mpg .mpeg`	Vídeo en formato MPEG

2. La Web

HTTP 1.1: Cookies

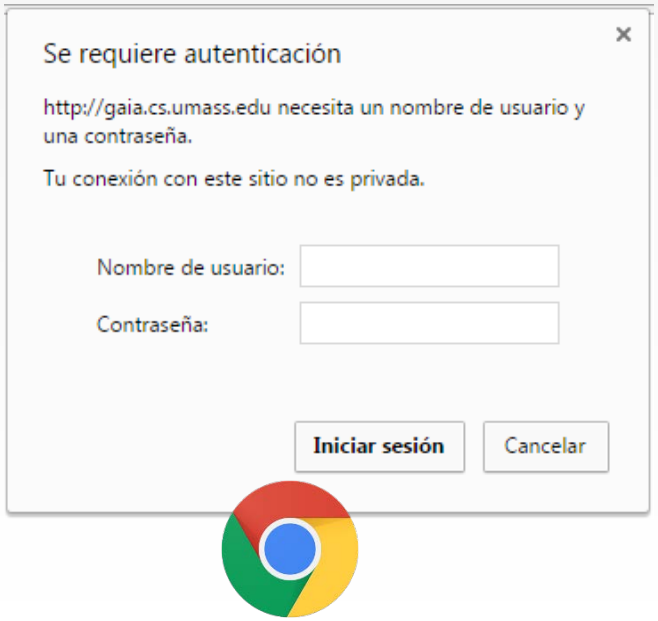
- HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre clientes
- Para guardar información sobre el estado de los clientes, HTTP usa el mecanismo de **cookies**
- La tecnología de cookies tiene 4 componentes:
  - Cabecera `Set-cookie` en respuesta
  - Cabecera `Cookie` en petición
  - Fichero de cookie almacenado en el navegador
  - Base de datos de cookies en el servidor web

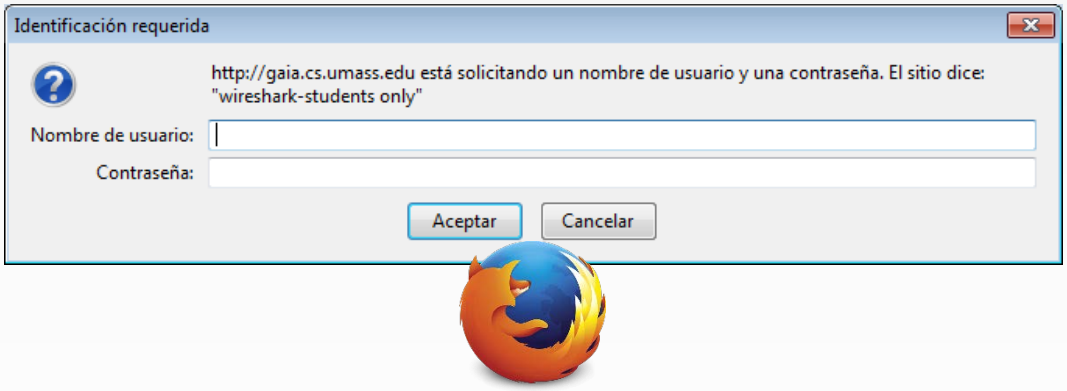


2. La Web

HTTP 1.1: Autenticación básica

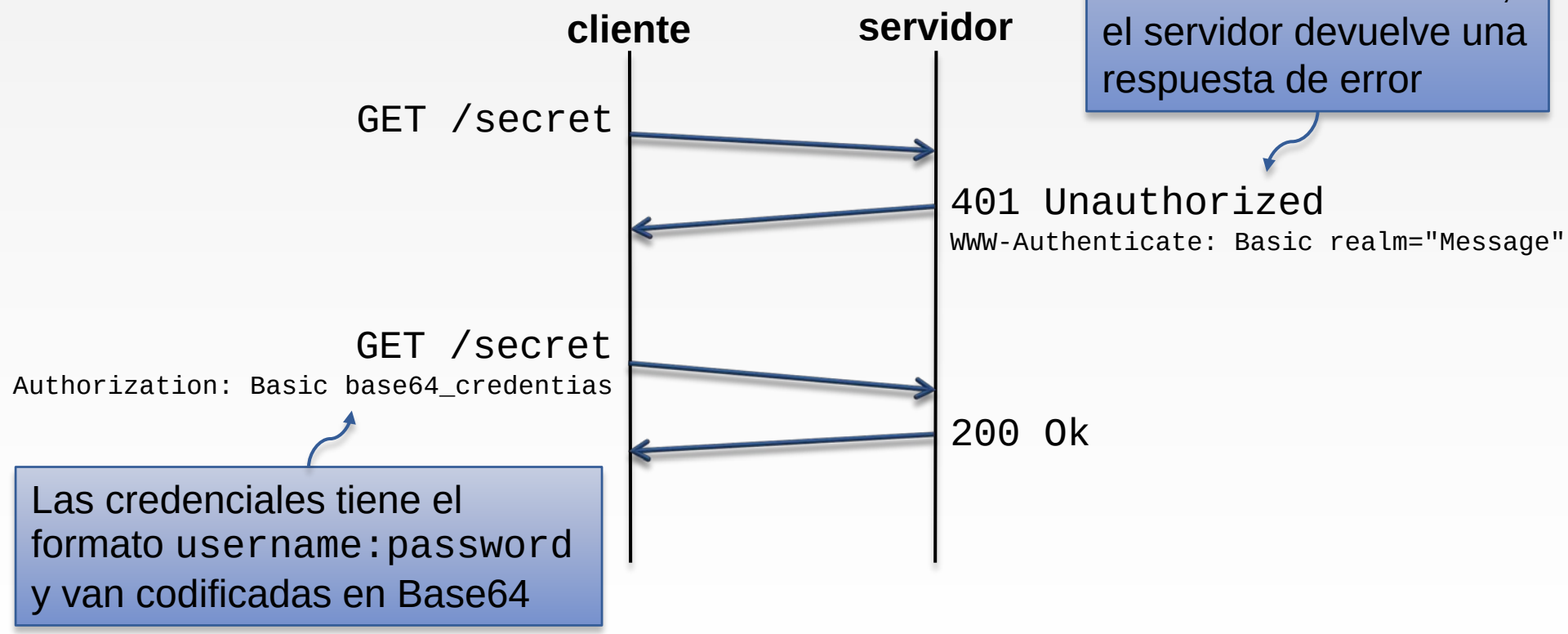
- HTTP proporciona es la capacidad de autenticación (esto es, confirmar la identidad del cliente)
- El navegador pedirá autenticación (nombre de usuario y contraseña):





2. La Web

HTTP 1.1: Autenticación básica



Al hacer una petición a un recurso protegido sin enviar las credenciales, el servidor devuelve una respuesta de error

Las credenciales tiene el formato `username:password` y van codificadas en Base64

2. La Web

HTTP 1.1: Autenticación básica

- **Base64** es sistema de codificación de binario a texto usado en el protocolo de correo electrónico SMTP
- La autenticación básica en HTTP reutiliza esta norma para codificar las credenciales que se usan para autenticarse frente al servidor
- Base64 usa un alfabeto de 64 caracteres consistente en los caracteres en mayúscula y minúscula de la tabla ASCII (`A-Z`, `a-z`), los numerales (`0-9`) y los símbolos + y `/`
- Ejemplo de codificador-decodificador online de Base64: https://www.base64encode.org/

2. La Web

HTTP 1.1: Autenticación básica

- Ejemplo de codificación en Base64:

h	o	l	a

h	o	l	a
01101000	01101111	01101100	01100001

a	G	9	s	Y	Q
011010	000110	111101	101100	011000	010000

a	G	9	s	Y	Q	=	=

1. Palabra a codificar

2. Se usa la tabla ASCII para codificar cada carácter

3. Se usa la tabla de índices de Base64

4. Resultado final (siempre va a ser múltiplo de 4 caracteres)

2. La Web

HTTP 1.1: Autenticación básica

- Tabla ASCII (menos caracteres no imprimibles):

Binario	Carácter	Binario	Carácter	Binario	Carácter	Binario	Carácter	Binario	Carácter
00100000	espacio	00110011	3	01000110	F	01011001	Y	01101100	l
00100001	!	00110100	4	01000111	G	01011010	Z	01101101	m
00100010	"	00110101	5	01001000	H	01011011	[	01101110	n
00100011	#	00110110	6	01001001	I	01011100	\	01101111	o
00100100	$	00110111	7	01001010	J	01011101	]	01110000	p
00100101	%	00111000	8	01001011	K	01011110	^	01110001	q
00100110	&	00111001	9	01001100	L	01011111	_	01110010	r
00100111	'	00111010	:	01001101	M	01100000	`	01110011	s
00101000	(	00111011	;	01001110	N	01100001	a	01110100	t
00101001	)	00111100	<	01001111	O	01100010	b	01110101	u
00101010	*	00111101	=	01010000	P	01100011	c	01110110	v
00101011	+	00111110	>	01010001	Q	01100100	d	01110111	w
00101100	,	00111111	?	01010010	R	01100101	e	01111000	x
00101101	-	01000000	@	01010011	S	01100110	f	01111001	y
00101110	.	01000001	A	01010100	T	01100111	g	01111010	z
00101111	/	01000010	B	01010101	U	01101000	h	01111011	{
00110000	0	01000011	C	01010110	V	01101001	i	01111100	|
00110001	1	01000100	D	01010111	W	01101010	j	01111101	}
00110010	2	01000101	E	01011000	X	01101011	k	01111110	~

2. La Web

HTTP 1.1: Autenticación básica

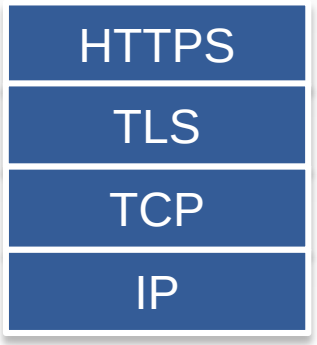
- Tabla de índices Base64:

Decimal	Binario	ASCII	Decimal	Binario	ASCII	Decimal	Binario	ASCII	Decimal	Binario	ASCII
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/

2. La Web

HTTPS

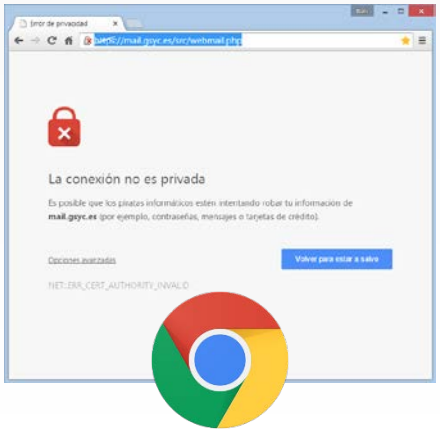
- HTTPS (*Hypertext Transfer Protocol Secure*) es la versión segura de HTTP
- Con HTTPS se consigue que la información sensible (claves, etc) no pueda ser interceptada por un atacante, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar
- TLS (*Transport Layer Security*) es un protocolo que proporcionan cifrado sobre conexiones TCP
- Puerto por defecto para servidores web que usen HTTPs: 443

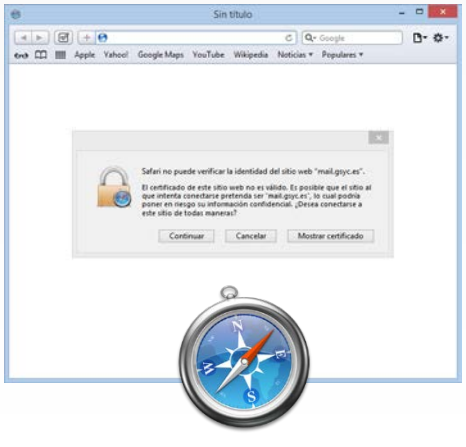


2. La Web

HTTPS

- Un servidor web seguro debe contar con un **certificado** emitido por una **autoridad de certificación** (CA, *Certificate Authority*)
- Los navegadores tienen una lista de CAs conocidas
- Al recibir un certificado no valido muestra una alerta de seguridad al usuario. Esto ocurre cuando:
  - El certificado firmado por una CA no conocida
  - El certificado ha caducado

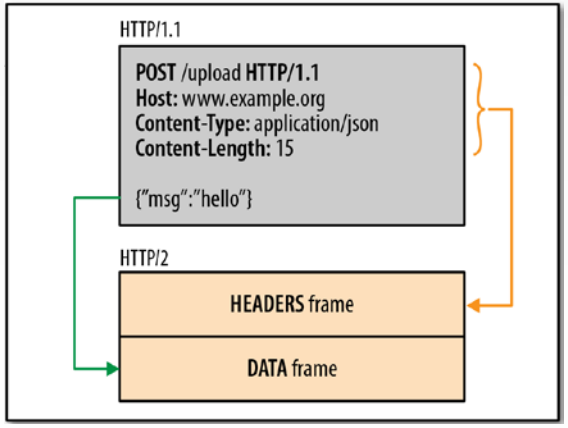




2. La Web

HTTP 2

- Introduce mejoras encaminadas a disminuir la latencia:
  - Multiplexación de peticiones (las respuestas se pueden procesar asíncronamente)
  - Servicio *push* (el servidor puede mandar recursos antes de que exista una petición explícita por parte del cliente)
  - Compresión de cabeceras (mediante un mecanismo llamado HPACK, definido en la RFC 7541)
  - HTTP 2 conversa toda la semántica de HTTP 1.1 (verbos, cabeceras, respuestas) pero el protocolo pasa a ser binario en lugar de textual



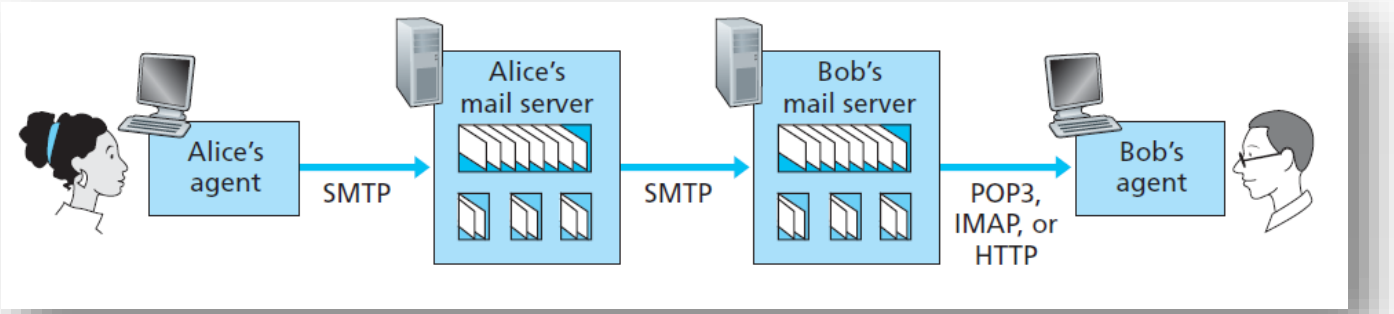
Índice de contenidos

3. Correo electrónico

- El correo electrónico (**e-mail**) es servicio **asíncrono** para enviar y recibir mensajes digitales sobre Internet
  - En un sistema de comunicaciones síncrono el transmisor debe coordinarse con el receptor antes del envío de datos. Ejemplos: teléfono, videollamada
  - En un sistema de comunicaciones asíncrono no hay coordinación temporal entre emisor y receptor. Otros ejemplos: correo postal, mensajes de texto
- Es un servicio basado en una arquitectura **cliente-servidor**
  - Cliente: *Mail User Agent* (MUA)
  - Servidor: *Mail Transfer Agent* (MTA)
- Direcciones de correo electrónico: usuario@servidor.tld

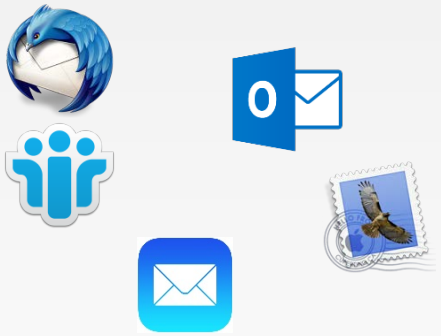
3. Correo electrónico

- Protocolos:
  - Envío de correo y comunicación entre servidores: **SMTP**
  - Recepción de correo: **IMAP** y **POP**
- Hay un buzón de correo (**mailbox**) por cada usuario → base de datos de mensajes



3. Correo electrónico

- Algunos ejemplos de clientes de correo electrónico:
  - Thunderbird
  - Microsoft Outlook
  - IBM Notes
  - Apple Mail
  - iOS Mail



- Algunos ejemplos de clientes web (*webmail*):
  - Gmail
  - Office365

- Algunos ejemplos de servidores de correo:
  - Sendmail
  - Postfix

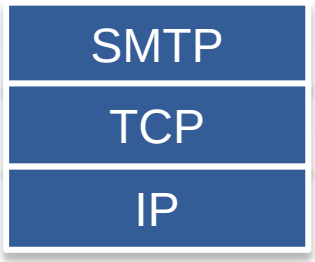
3. Correo electrónico

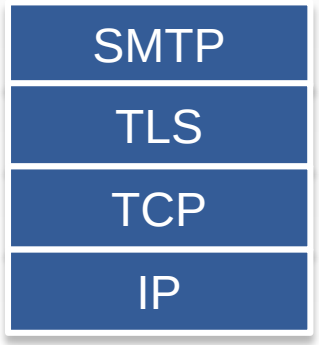
- Normalmente, los servidores de correo requieren autenticación previa al envío de correos electrónicos
- Puede ocurrir que esta autenticación no sea necesario, en ese caso se dice que el servidor es ***open relay***
- Esta configuración era la forma habitual de los servidores de correo en el inicio de Internet
- Hoy día estos servidores suelen estar en la lista negra debido a su uso como **spam**
  - Spam = correo no deseado
  - Lista negra = lista de servidores no admitidos

3. Correo electrónico

SMTP

- SMTP = *Simple Mail Transfer Protocol* (RFC 2821)
- Protocolo usado para enviar correo electrónico por los clientes y para el intercambio de mensajes entre servidores
- Puertos por defecto:
  - Conexión MUA-MTA sin TLS: 465
  - Conexión MUA-MTA con TLS: 587
  - Conexión MTA-MTA: 25

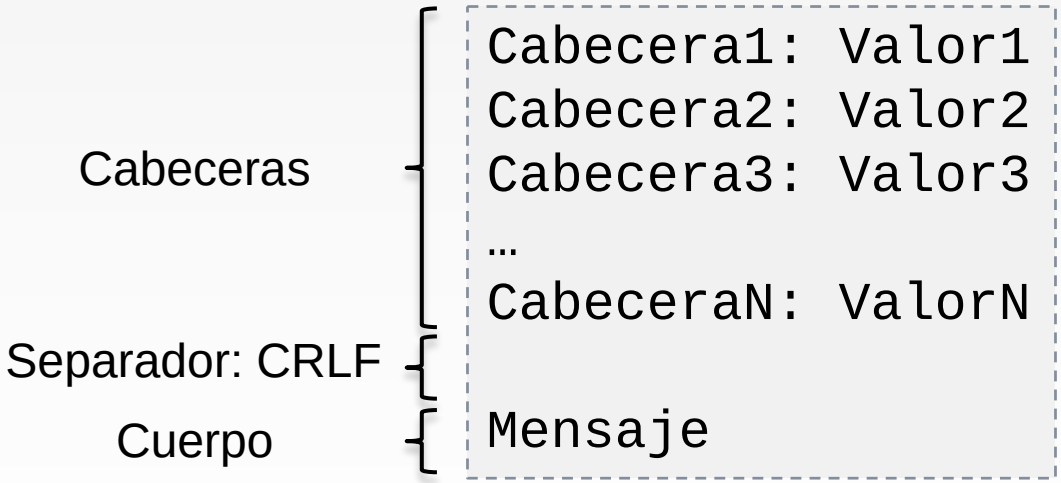




3. Correo electrónico

SMTP

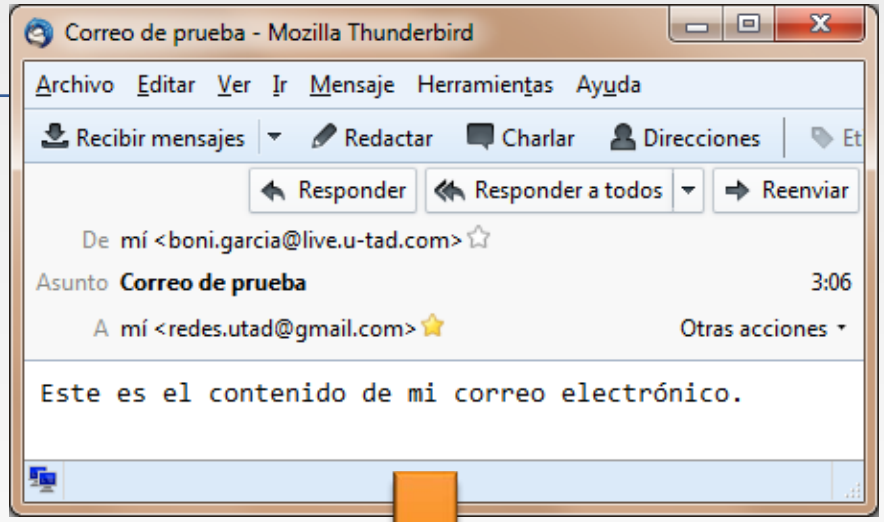
- SMTP es un protocolo **basado en texto**, por lo tanto los mensajes
- El formato de los emails está definido en la RFC 5322:



3. Correo electrónico

SMTP

- Ejemplo real de correo electrónico:




```
Message-ID: <54348E21.5030003@live.u-tad.com>
Date: Wed, 08 Oct 2014 03:06:41 +0200
From: =?UTF-8?B?Qm9uaSBHYXJjw61h?= <boni.garcia@live.u-tad.com>
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:31.0) Gecko/20100101 Thunderbird/31.1.2
MIME-Version: 1.0
To: redes.utad@gmail.com
Subject: Correo de prueba
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 8bit

Este es el contenido de mi correo electrónico.
```

Los protocolos de recepción (IMAP o POP) añadirán cabeceras extra al mensaje recibo

3. Correo electrónico

SMTP

- Algunas de las cabeceras SMTP más importantes:

Cabecera	Tipo	Descripción
`From:`	Recomendada	Dirección de e-mail del remitente
`To:`	Recomendada	Dirección de e-mail del destino
`Subject:`	Recomendada	Asunto
`Cc:`	Opcional	Dirección de e-mail del destino (copia)
`Bcc:`	Opcional	Dirección de e-mail del destino (copia oculta)
`User-Agent:`	Opcional	Cadena de texto que identifica al cliente
`Message-ID:`	Opcional	Identificador único del correo electrónico
`Date:`	Opcional	Fecha en la que fue enviado por el cliente
`Reply-to:`	Opcional	Dirección de respuesta. Si no aparece, será `From:`

3. Correo electrónico

SMTP

- MIME = Multi-Purpose Internet Mail Extensions (RFCs 2045, 2046, 2047, 4288, 4289 y 2049)
- Es una extensión a SMTP para permitir enviar/recibir contenidos diferentes a ASCII (audio, video, imágenes, …)
- Cabeceras MIME:

Cabecera	Tipo	Descripción
`MIME-Version:`	Opcional	Versión MIME: `1.0`
`Content-Type:`	Opcional	Tipo MIME
`Content-Disposition:`	Opcional	Nombre de los ficheros adjuntos
`Content-Transfer-Encoding:`	Opcional	Tipo de codificación. Suele ser `8bit` o `base64`

3. Correo electrónico

SMTP

- Los adjuntos se implementan mediante **mensajes *multiparte***
  - Cada parte tiene sus propias cabeceras

Definición frontera

Parte 1

Parte 2

Los contenido nos textuales van codificados en Base64

```
Message-ID: <54347AC1.3090703@live.u-tad.com>
Date: Wed, 08 Oct 2014 01:44:01 +0200
From: =?UTF-8?B?Qm9uaSBHYXJjw61h?= <boni.garcia@live.u-tad.com>
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:31.0) Gecko/20100101 Thunderbird/31.1.2
MIME-Version: 1.0
To: redes.utad@gmail.com
Subject: Prueba
Content-Type: multipart/mixed; boundary="------------030308000808080505060200"

This is a multi-part message in MIME format.
--------------030308000808080505060200
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 8bit

Hola, qué tal?

--------------030308000808080505060200
Content-Type: image/jpeg; name="utad.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="utad.jpg"

/9j/4QjJRXhpZgAASUkqAAgAAAAMAAABAwABAAAAKgMAAAEBAwABAAAAbgEAAAIBAwADAAAA
ngAAAAYBAwABAAAAAgAAABIBAwABAAAAAQAAABUBAwABAAAAAwAAABoBBQABAAAApAAAABsB
.....
8LLJhjPmGP8A/Kg/yh/6luD/AJGT/wDVTL/5Sz/zv9y1/lcfd/spfrTL/lU35ef4f/w9+ho/
0N9a+vfU/Um4/WOPD1OXPn9nanLjlf53Lx8fF6q4en0p/LQ4eGtvi//Z
--------------030308000808080505060200--
```

3. Correo electrónico

SMTP

- Algunos comandos SMTP de petición:

Comando	Descripción	Ejemplo
HELO	Comando inicial	`HELO <nameserver>`
EHLO	Equivalente a HELO, pero obtiene como respuesta las extensiones de SMTP que implementa el servidor (por ejemplo: AUTH)	`EHLO <nameserver>`
AUTH	Autenticación	`AUTH LOGIN`
MAIL	Identifica el correo origen	`MAIL from: <boni.garcia@live.u-tad.com>`
RCPT	Identifica el correo destino	`RCPT to: redes.utad@gmail.com`
DATA	Inicio del mensaje a enviar	`DATA`
QUIT	Cierra la conexión con el servidor	`QUIT`

3. Correo electrónico

SMTP

- Algunos comandos SMTP de respuesta:

Código	Descripción
221	Servidor cierra conexión
250	Acción completada de forma correcta
235	Autenticación correcta
334	Comienzo de autenticación en base64
354	Respuesta a DATA. Comienzo del mensaje a enviar. Se acaba el mensaje del email enviando <CRLF>.<CRLF> (↲.↲)
535	Autenticación incorrecta

3. Correo electrónico

SMTP

- Ejemplo real de envío de mensaje con SMTP:

```
HELO smtp
250 smtp.gmail.com at your service
AUTH LOGIN
334 VXNlcm5hbWU6
<base64_loging>
334 UGFzc3dvcmQ6
<base64_password>
235 2.7.0 Authentication successful
MAIL from: <redes.utad@gmail.com>
250 2.1.0 OK 184sm6880451wmv.35 - gsmtp
RCPT to: <boni.garcia@live.u-tad.com>
250 2.1.5 OK 184sm6880451wmv.35 - gsmtp
DATA
354  Go ahead 184sm6880451wmv.35 - gsmtp
From: boni.garcia@live.u-tad.com
To: redes.utad@gmail.com
Subject: Esto es una prueba

Hola!
.
250 2.0.0 OK 1506271877 184sm6880451wmv.35 - gsmtp
QUIT
221 2.0.0 closing connection 184sm6880451wmv.35 – gsmtp read:errno=0
```

En azul las peticiones (comandos desde el cliente)

3. Correo electrónico

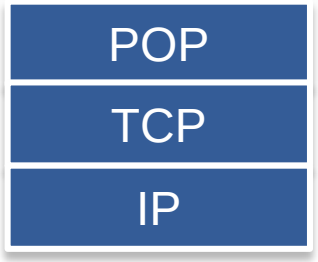
SMTP

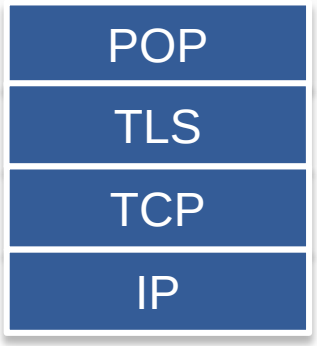
- *Dot-stuffing* (relleno de puntos): Mecanismo usado por SMTP para conseguir que la línea de finalización de correo (<CRLF>.<CRLF>) sea única.
- Antes en enviar un correo, el cliente comprueba el primer carácter de cada línea. Si es un punto, añade otro punto adicional
- Cuando un correo es recibido por un servidor SMTP, se comprueba también cada línea.
- Si hay varios puntos al inicio de una línea, elimina uno de los puntos. Si es un punto único, entonces es el final del correo

3. Correo electrónico

POP

- POP = *Post Office Protocol* (RFC 1939)
- Protocolo basado en texto usado para leer correo electrónico
- Principal diferencia respecto a IMAP: POP se descarga el correo en local y no se almacena en el servidor
- Puertos por defecto:
  - Sin TLS: 110
  - Con TLS: 995





3. Correo electrónico

POP

- Algunos comandos POP de petición:

Comando	Descripción	Ejemplo
USER	Identificación de usuario	`USER micorreo@midominio↵`
PASS	Contraseña de usuario	`PASS micontraseña↵`
STAT	Devuelve el número de mensajes en el buzón	`STAT↵`
LIST	Muestra todos los mensajes en el buzón	`LIST↵`
RETR	Descarga un mensaje del buzón	`RETR número_correo↵`
TOP	Descarga cabecera y un número de líneas del mensaje	`TOP número_correo número_líneas↵`
QUIT	Cierra la conexión con el servidor	`QUIT↵`

3. Correo electrónico

POP

- Ejemplo real de lectura de mensaje con POP:

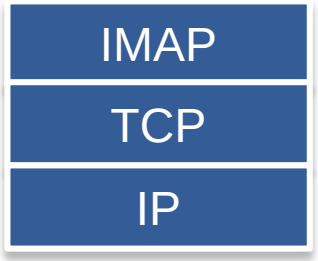
En azul las peticiones (comandos desde el cliente)

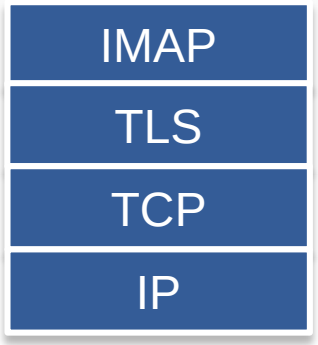
```
USER boni.garcia@live.u-tad.com
+OK
PASS <password>
+OK User successfully logged on.
LIST
+OK 52 3890769
1 14357
2 17457
3 171603
.
RETR 1
+OK
Date: Fri, 19 Sep 2014 19:42:04 +0200
From: <remitente@dominio>
To: <destino@dominio>

Estimados profesores/as:

Bla bla bla
.
QUIT
+OK Microsoft Exchange Server 2013 POP server signing off.
read:errno=0
```

3. Correo electrónico

IMAP

- IMAP = *Internet Mail Access Protocol* (RFC 3501)
- Protocolo basado en texto usado para leer correo electrónico
- Principal diferencia respecto a POP: IMAP almacena el correo en el servidor
- Puertos por defecto:
  - Sin TLS: 143
  - Con TLS: 993





3. Correo electrónico

IMAP

- Algunos comandos IMAP de petición:

Comando	Descripción	Ejemplo
LOGIN	Identificación de usuario	`a1 LOGIN usuario@dominio passwd`
LIST	Lista los buzones existentes	`a2 LIST "" %`
EXAMINE	Examina buzón de correo (determinado por su nombre)	`a3 EXAMINE INBOX`
SELECT	Selecciona buzón de correo (determinado por su nombre)	`a4 SELECT INBOX`
FETCH	Lee un correo (determinado por su número)	`a5 FETCH 1 BODY[]`
LOGOUT	Cierra la conexión con el servidor	`a6 LOGOUT`

3. Correo electrónico

IMAP

- Ejemplo real de lectura de mensaje con IMAP:

```
a1 LOGIN boni.garcia@live.u-tad.com <password>
a1 OK LOGIN completed.
a2 LIST "" %
* LIST (\Marked \HasNoChildren) "/" INBOX
...
a2 OK LIST completed.
a3 EXAMINE INBOX
* 52 EXISTS
* 0 RECENT
...
a3 OK [READ-ONLY] EXAMINE completed.
a4 SELECT INBOX
* 52 EXISTS
* 0 RECENT
...
a4 OK [READ-WRITE] SELECT completed.
a5 FETCH 1 BODY[]
* 1 FETCH (BODY[] {4609}
Date: Fri, 19 Sep 2014 19:42:04 +0200
From: <remitente@dominio>
To: <destino@dominio>

Estimados profesores/as:

Bla bla bla
.
a5 OK FETCH completed.
a6 LOGOUT
* BYE Microsoft Exchange Server 2013 IMAP4 server signing off.
a6 OK LOGOUT completed.
```

En azul las peticiones (comandos desde el cliente)

Cada comando en IMAP lleva un identificador llamado tag. Cada respuesta se asocia con ese tag

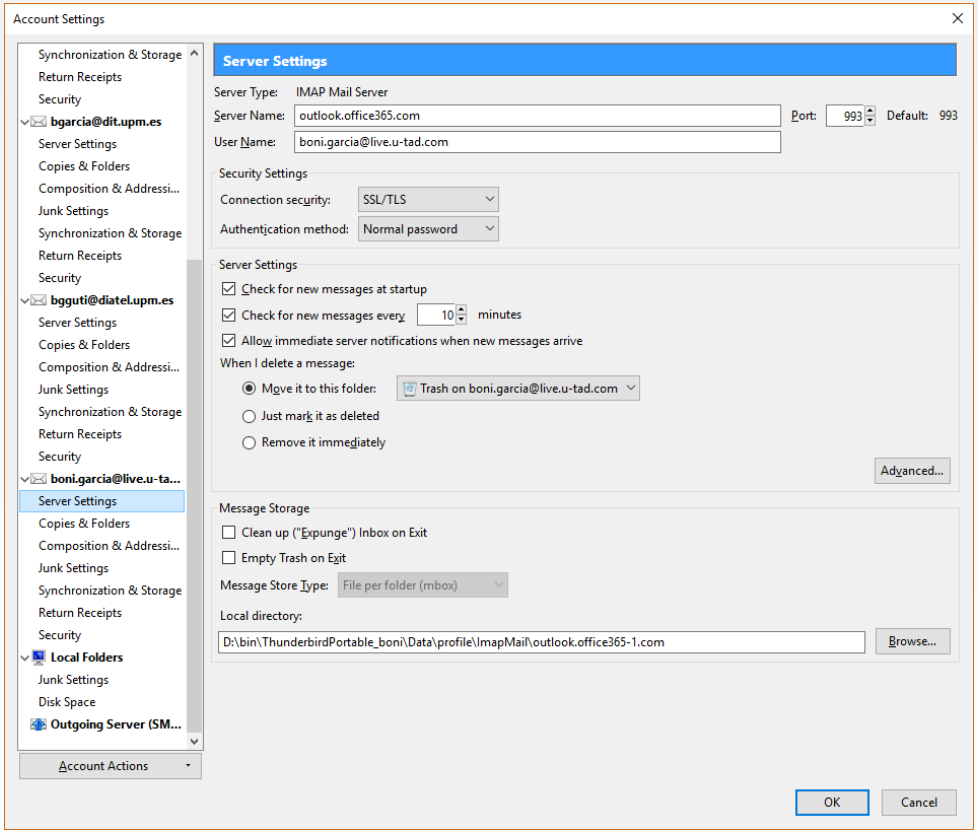
3. Correo electrónico

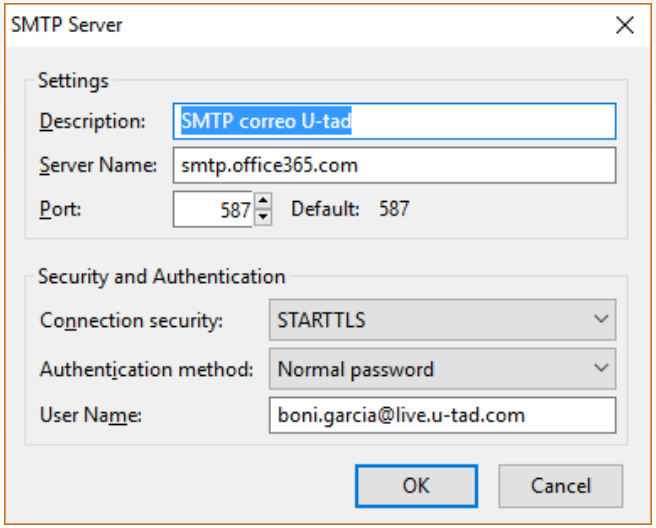
STARTTLS

- STARTTLS es una extensión para los protocolos basados en texto (como SMTP, POP, e IMAP)
  - STARTTLS para IMAP y POP está definido en el RFC 2595
  - STARTTLS para SMTP está definido en el RFC 3207
- Permite la negociación de parámetros de seguridad para pasar de una conexión TCP no segura a una cifrada (TLS) por el mismo puerto
  - Permite que la conexión entre servidores de correo sea segura
- Los principales proveedores del mundo Internet soportan STARTTLS:
  - Amazon, Google, Microsoft, Facebook, Dropbox…

3. Correo electrónico

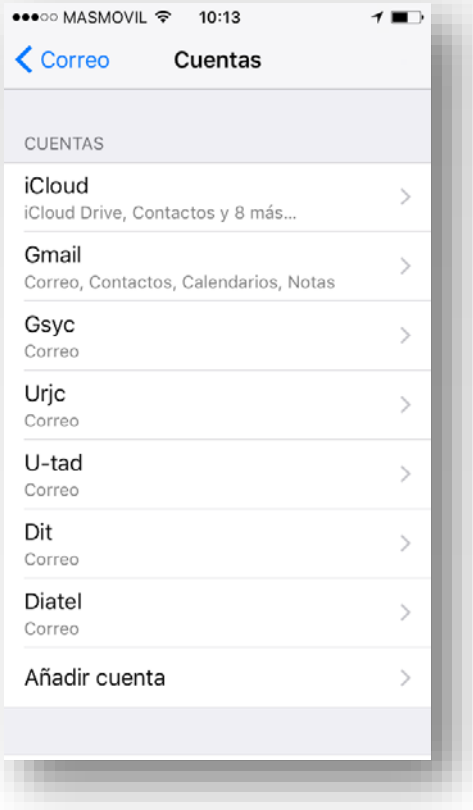
Configuración manual de clientes de correo

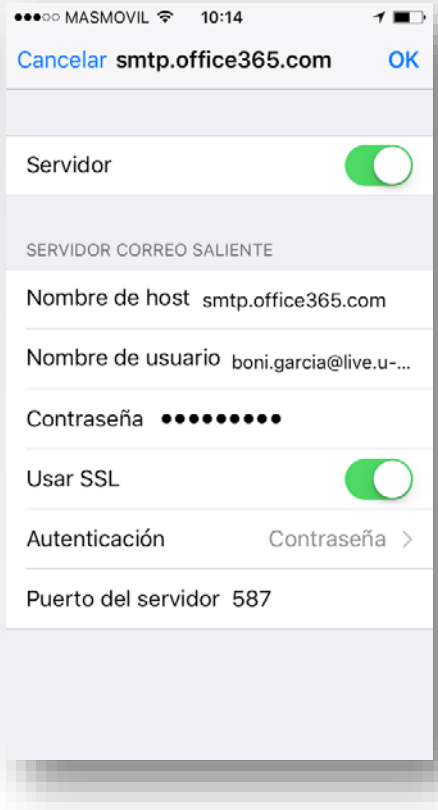




3. Correo electrónico

Configuración manual de clientes de correo





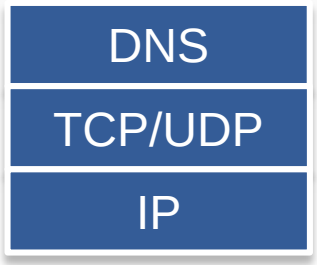
Índice de contenidos

4. Sistema de nombres de dominio

- DNS = *Domain Name System* (RFCs 1034, 1035)
- DNS es un protocolo de aplicación **cliente-servidor** cuya función más importante es traducir (**resolver**) nombres inteligibles para los humanos (**nombres de dominio**) en direcciones IP y viceversa
- El servidor de DNS más usado es BIND (*Berkley Internet Name Domain*), instalado en sistemas UNIX
- Los clientes de DNS se conocen como (*resolver*). Por ejemplo, la herramienta de línea de comando `nslookup` (o como alternativa en sistemas Linux/Unix, `dig`)
- DNS se instrumenta mediante una base de datos de distribuida en los servidores

4. Sistema de nombres de dominio

- DNS es usado normalmente por otros protocolos de aplicación como HTTP, SMTP, y FTP
- En las consultas de los clientes DNS se usa UDP
  - Puerto por defecto de destino de peticiones: 53
- Hay un caso un caso puntual en el que se emplea TCP, la transferencias de zonas, que realiza entre servidores



4. Sistema de nombres de dominio

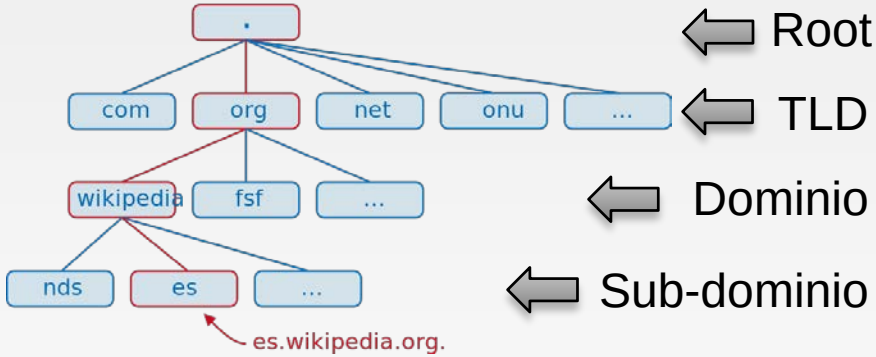
Servicios proporcionados por DNS

1. **Resolución** de nombres:
   - Resolución directa: dado un nombre de dominio, obtener su dirección IP
   - Resolución inversa: dada una dirección IP, obtener su nombre de dominio
2. **Alias**. Pseudónimo para nombres de dominio. Por ejemplo, un dominio llamado midominio.es podría tener un alias www.midominio.es (ambos nombres de dominio apuntan a la misma dirección IP)
3. **Distribución de carga**. DNS puede ser usado para balancear carga a servidores replicados (DNS *Round Robin*). Esto es útil para servidores especialmente cargados (por ejemplo: servidores de correo, servidores web)
   - Round Robin es un método para seleccionar los elementos en un grupo comenzando por el primer elemento de la lista hasta llegar al último sucesivamente

4. Sistema de nombres de dominio

Nombres de dominio

- Estructura jerárquica en árbol
- El nivel superior de esta jerarquía se llama simplemente raíz (***root***) se representa con un punto (.)



- **TLD** (*Top Level Domain*): Dominio de nivel superior. Identifica el tipo de dominio (`.com`, `.org`, ...)
- Después de TLD encontramos el **dominio** y sus **sub-dominios**, que están administrados por la misma entidad
- FQDN (*Fully Qualified Domain Name*): consiste en la concatenación de todas las etiquetas de un camino incluyendo el punto al final
  - En el ejemplo: es.wikipedia.org. El punto final (*trailing dot*) se suele omitir
  - Longitud máxima de un FQDN = 255 caracteres
  - Profundidad máxima del árbol de la jerarquía de dominios = 127 niveles
  - Máxima longitud de cada nodo del árbol de la jerarquía de dominios = 63 caracteres

4. Sistema de nombres de dominio

Tipos de TLDs

- **Geográficos** (ccTLD). Usados por un país o territorio independiente. Tienen dos letras. Por ejemplo: `.es`, `.us`, `.de`, `.fr`, `.uk`, `.jp`, …
  - Segundo nivel (SLD). Organizaciones dentro de un país: `.co.uk`, `.co.jp`, …
- **Genéricos** (gTLD). Usados por una clase particular de organización. Tienen tres o más letras. Por ejemplo: `.com` (comercial), `.org` (inicialmente organizaciones sin ánimo de lucro, hoy sin limitación), `.net` (inicialmente para infraestructuras de red, hoy día sin limitación)…
- **Patrocinados** (sTLD): Existen reglas para optar al dominio. Por ejemplo: `.edu` (fines educativos), `.int` (organismos internacionales),…
- De **infraestructura**. En este grupo hay un único TLD: `.arpa`. Se usa en la resolución inversa (*Address and Routing Parameter Area*)

https://www.iana.org/domains/root/db

4. Sistema de nombres de dominio

Organismos involucrados en el DNS

- Administración servidores raíz: **ICANN** (*Internet Corporation for Assigned Names and Numbers*)
  - www.icann.org
- Registro servidores TLD: **IANA** (*Internet Assigned Numbers Authority*)
  - www.iana.org
- Dominios españoles: **Red.es** (entidad pública dependiente del Ministerio de Energía, Turismo, y Agencia Digital)
  - www.dominios.es
  - Lista completa de agentes registradores (de forma genérica llamados *registrars*) de dominio `.es`:

    http://www.dominios.es/dominios/es/agentes-registradores/todos-los-agentes-registradores

4. Sistema de nombres de dominio

Tipos de servidores de DNS

- Servidores **raíz** (*root servers*). Hay 13 servidores raíz (etiquetados desde A hasta M) replicados a través del mundo
- Servidor de dominio superior (**TLD**). Servidor para cada una de las zonas `.com`, `.es`, `.net` , etc.
- Servidores **autoritativos**: Devuelven la dirección que se busca sólo si es una dirección de su zona de autoridad (espacio de nombres que gestionan). Si no es así, devolverá una lista de servidores a los que preguntar. Hay dos tipos:
  - Primario (*master*): Copia principal de la información de zona
  - Secundario (*slave*): Replica del primario
- Servidores **no autoritativos** (también conocidos como servidores locales): No son capaces de realizar la resolución de nombre por sí mismos y realizan peticiones recursivas. Para mejorar el rendimiento, mantienen cachés con las peticiones resueltas

4. Sistema de nombres de dominio

Servidores raíz

- Los servidores raíz almacenan una lista de los nombres de dominio y direcciones IP de todos los servidores TLD
- Cada operador utiliza equipos redundantes de servidores raíz para ofrecer un servicio fiable
- Se localiza el servidor DNS más cercano (*anycast*)

http://www.root-servers.org/

4. Sistema de nombres de dominio

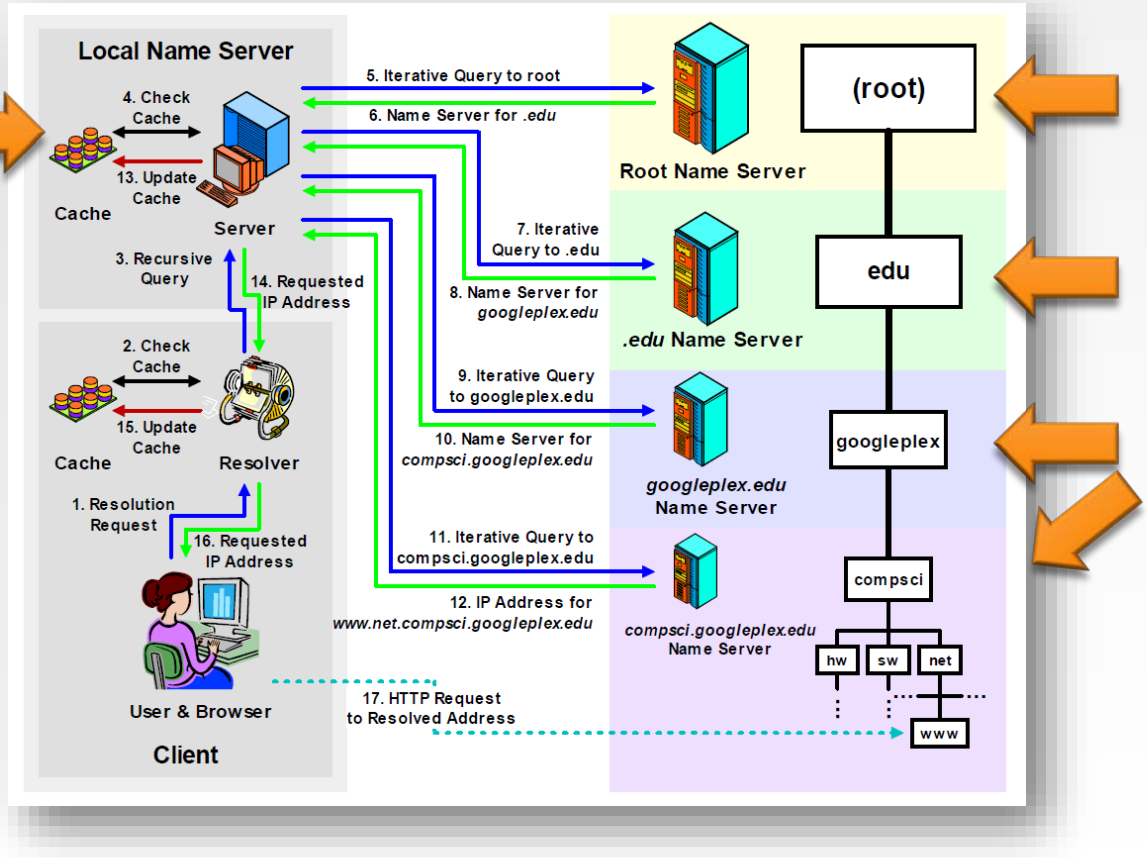
Servidores raíz

Hostname	Dirección IP	Operador
a.root-servers.net	198.41.0.4	VeriSign, Inc.
b.root-servers.net	192.228.79.201	University of Southern California (ISI)
c.root-servers.net	192.33.4.12	Cogent Communications
d.root-servers.net	199.7.91.13	University of Maryland
e.root-servers.net	192.203.230.10	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4	US Department of Defence (NIC)
h.root-servers.net	128.63.2.53	US Army (Research Lab)
i.root-servers.net	192.36.148.17	Netnod
j.root-servers.net	192.58.128.30	VeriSign, Inc.
k.root-servers.net	193.0.14.129	RIPE NCC
l.root-servers.net	199.7.83.42	ICANN
m.root-servers.net	202.12.27.33	WIDE Project

4. Sistema de nombres de dominio

Funcionamiento DNS

Servidor no autoritativo



Servidor raíz

Servidor TLD

Servidores autoritativos

4. Sistema de nombres de dominio

Registros DNS

- Cada registro en la base de datos DNS se llama RR (*Resource Record*)
- Hay diferentes tipos (TYPE) de registro RR, y almacenan diferente información:
  - `A`: Nombre de host y dirección IPv4
  - `AAAA`: Nombre de host y dirección IPv6
  - `NS`: Nombre de host y servidor de DNS que sabe resolver ese nombre
  - `CNAME`: Alias de un host
  - `MX`: Servidor de correo electrónico
  - `PTR`: Registro para traducción inversa. Se insertan en la base de datos como el dominio especial `in-addr.arpa`.
  - `SOA`: Define los parámetros globales de una zona. En este registro se establece el TTL (tiempo de vida del recurso). Cada vez que un servidor DNS da una respuesta, lo hace adjuntando un TTL

4. Sistema de nombres de dominio

El estándar IDN

- Inicialmente, los nombres de dominio eran cadenas alfanuméricas (con '-' como único símbolo permitido)
- IDN (*Internationalized Domain Name*) es una extensión a DNS que permite (desde 2005) que un nombre de dominio contenga caracteres no-ASCII (incluso emojis)
  - Ejemplos: http://canción.com/, http://pequeñin.com/, https://i❤.ws/
- En IDN, en lugar de redefinir la infraestructura DNS existente, lo que se hace con los nombre de dominio no-ASCII es convertirlo a una forma basada en ASCII llamada **Punycode** (RFC 3492)
  - Ejemplo: españa.es = xn--espaa-rta.es
- Conversor online de Punycode: http://punycode.es/
- En la práctica no se están usando masivamente este tipo de dominios

4. Sistema de nombres de dominio

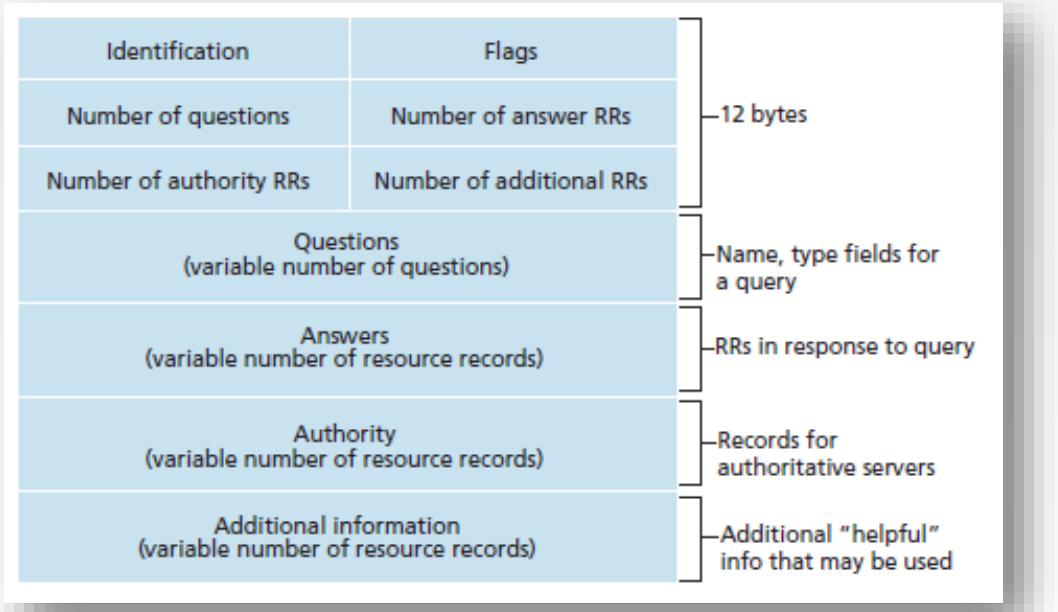
Transferencia de zonas

- La transferencia de zonas es el proceso mediante el que el contenido de servidor autoritativo se copia desde un servidor principal a un servidor secundario
- Los mensajes que se intercambian en este proceso van por **TCP**
- Se producirá una transferencia de zona durante cualquiera de los siguientes escenarios:
  - Al iniciar el servicio DNS en el servidor DNS secundario
  - Cuando caduca el tiempo de actualización
  - Cuando hay los cambios en el archivo de zona principal

4. Sistema de nombres de dominio

Mensajes DNS

- Hay dos tipos: peticiones y respuestas
- Ambos tipos de mensajes tienen la misma estructura:

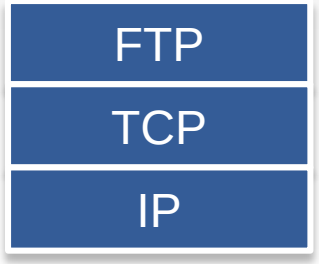


Índice de contenidos

5. Transferencia de ficheros

FTP

- FTP = *File Transfer Protocol* (RFCs 959, 2428), es un protocolo cliente-servidor de transferencia de archivos en redes IP
- Los puertos por defecto para servidores FTP son:
  - 20 TCP (puerto de datos)
  - 21 TCP (puerto de control)
- Ejemplos de servidores FTP:
  - Filezilla server, Xlight FTP Server (Windows)
  - vsftpd (*very secure FTP daemon*) (Linux)
- Ejemplo de clientes FTP:
  - Filezilla (multiplataforma)
  - Comando `ftp` en la línea de comandos



5. Transferencia de ficheros

FTP

- El intercambio de ficheros cliente-servidor puede ser bidireccional (clientes pueden subir y bajar ficheros)
- Los cliente normalmente tienen que autenticares frente a los servidores previamente a poder intercambiar ficheros
- Existe una modalidad en la que los clientes no necesitan autenticación llamada **FTP anónimo**:
  - Usuario: *anonymous*
  - Contraseña: cualquier valor
- FTP tiene dos modos de funcionamiento: **activo** y **pasivo**
- La mayoría de navegadores web actuales tiene soporte para FTP. La URL para acceder a un servidor FTP mediante navegador es:

```
ftp://[<user>[:<password>]@]<host>[:<port>]/<url-path>
```

5. Transferencia de ficheros

FTP

- Las **peticiones** FTP más usadas son:
  - `USER username`: Identificación
  - `PASS password`: Contraseña
  - `LIST`: Listado del directorio remoto
  - `CWD`: Cambiar directorio actual
  - `RETR filename`: Obtener fichero remoto
  - `STOR filename`: Copiar fichero a remoto
  - `DELE filename`: Elimina fichero remoto
  - `PORT`: Envío del puerto cliente para recibir datos (FTP activo)
  - `PASV`: Petición de envío de datos mediante FTP pasivo
  - `QUIT`: Terminar una sesión

5. Transferencia de ficheros

FTP

- Las **respuestas** FTP más comunes son:
  - `220 Ready`: Mensaje inicial del servidor (servidor listo)
  - `331 Need pasword`: Recibido nombre usuario, se necesita contraseña
  - `230 Greeting`: Contraseña válida, servidor lista para transferencias
  - `430 Invalid username or password`: Contraseña o nombre de usuario no válido
  - `150 File status ok`: Comienzo de transferencia de datos
  - `226 Transfer complete`: Fin de transferencia de datos
  - `227 Entering passive mode`: Cambio a modo pasivo

5. Transferencia de ficheros

FTP

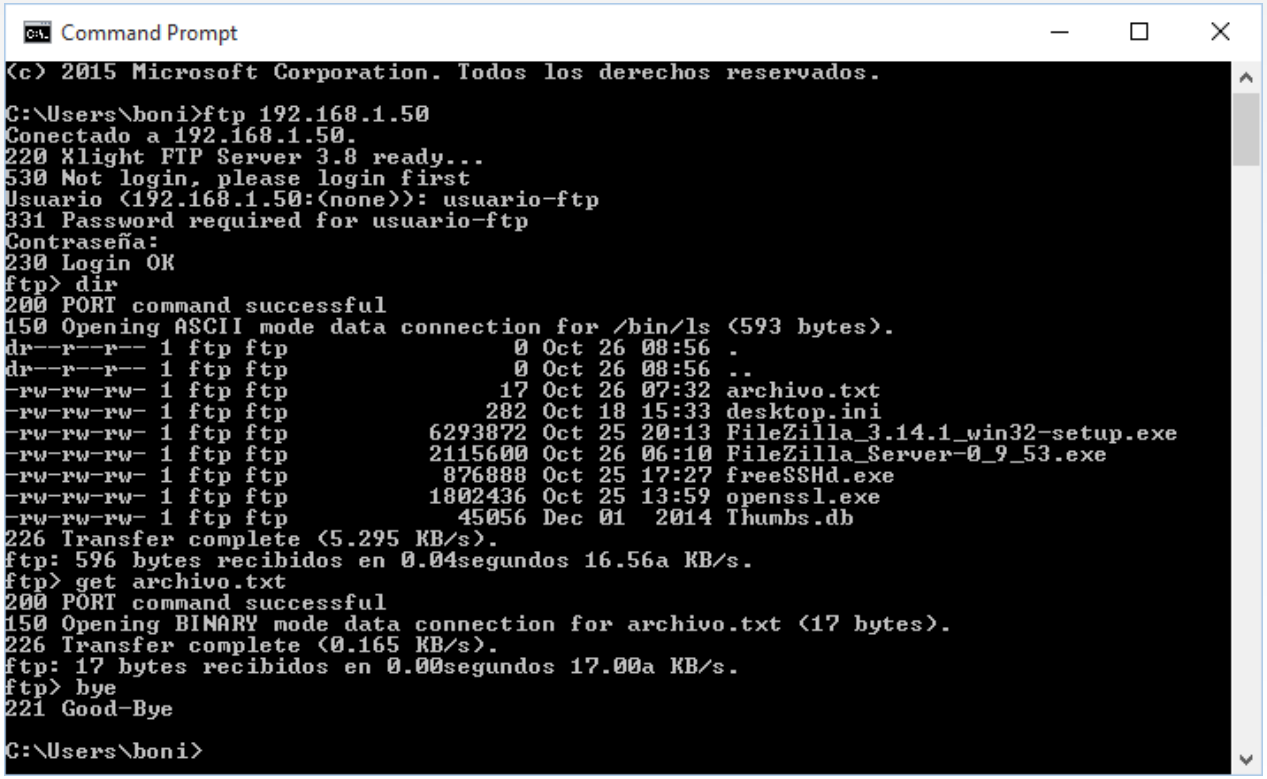
- Sintaxis de comandos de la herramienta `ftp` de línea de comandos:
  - `ftp <server>`: Inicio de sesión
  - `dir`: Obtener un listado de ficheros en el directorio actual
  - `ls`: Obtener un listado de ficheros en el directorio actual
  - `cd <folder>`: Cambiar de directorio
  - `get <file>`: Copiar fichero remoto a local
  - `del <file>`: Eliminar fichero remoto
  - `put <file>`: Copiar fichero local a remoto
  - `bye`: Terminar sesión
  - `help`: Obtiene un listado de todos los comandos posibles

http://www.tcpipguide.com/free/t_FTPSampleUserandInternalCommandDialog.htm

5. Transferencia de ficheros

FTP

- Ejemplo de interacción con servidor FTP a través de línea de comandos

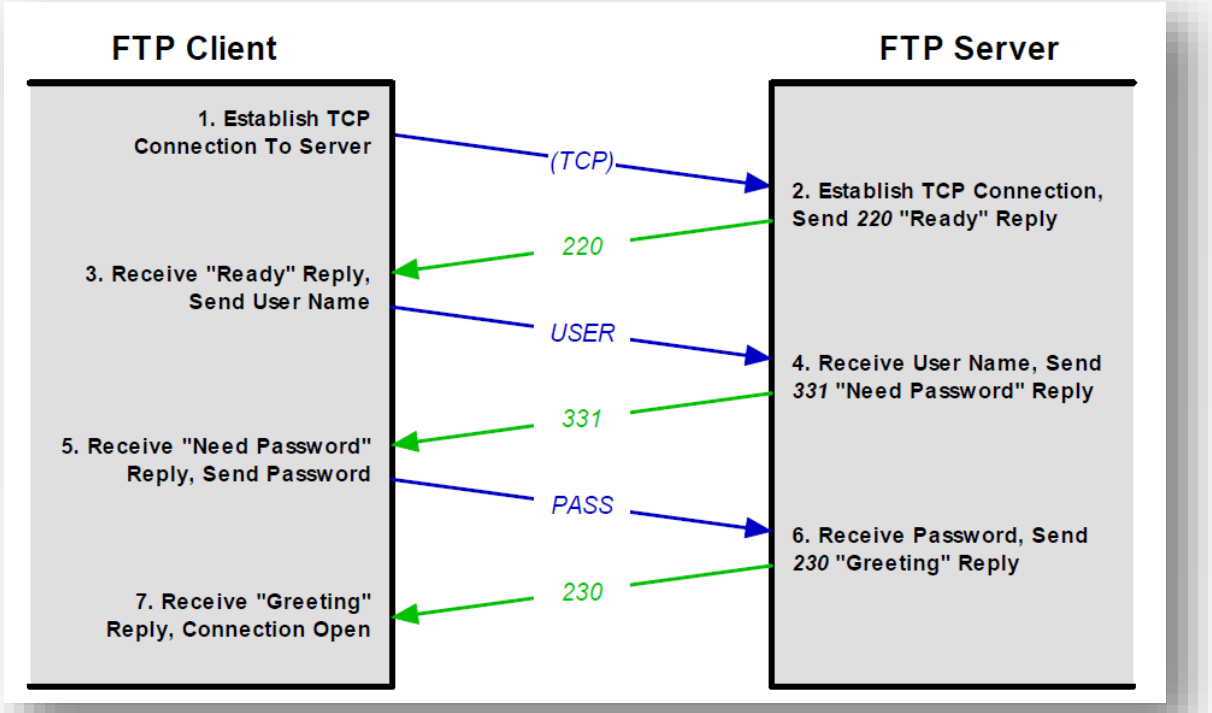


5. Transferencia de ficheros

FTP

- Establecimiento de conexión y autenticación de usuario:

El cliente se conecta desde un puerto aleatorio no privilegiado (>1024) al puerto de control del servidor (21)



5. Transferencia de ficheros

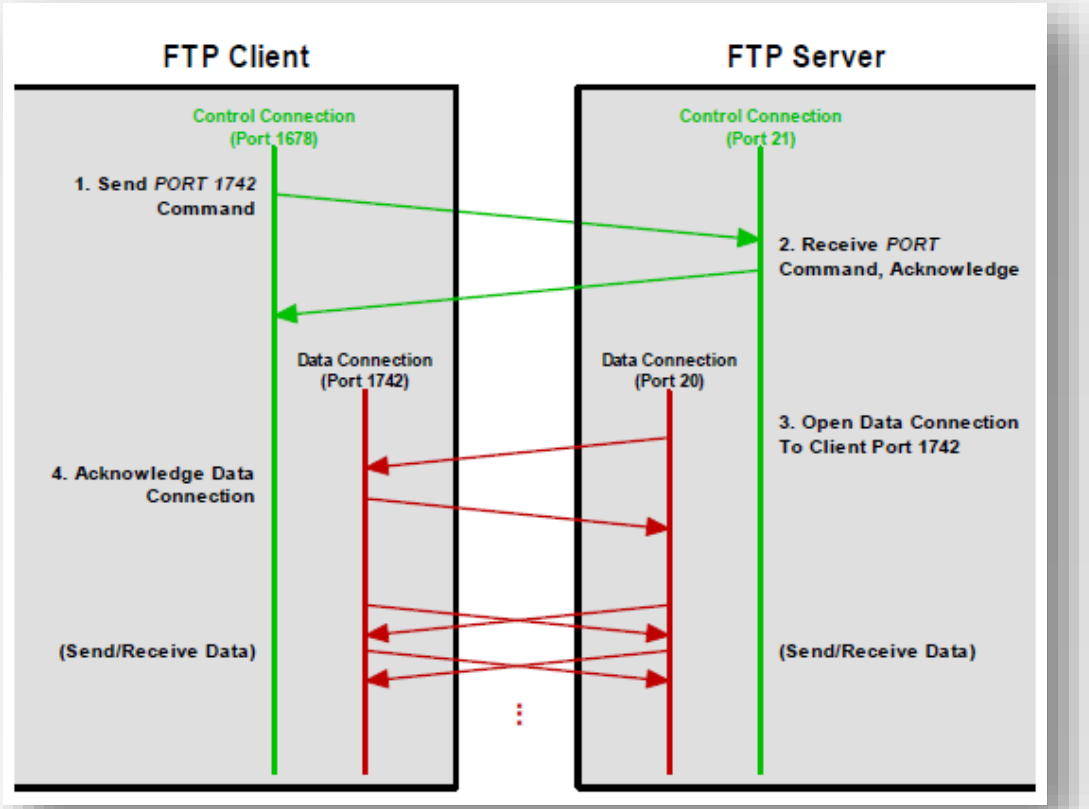
FTP

- FTP activo:

Además del puerto cliente usado para la conexión con el servidor, el cliente usa otro puerto no privilegiado (>1024) para recibir los datos

El cliente envía ese puerto usando el comando PORT al servidor

El servidor abre una conexión con ese puerto desde su puerto de datos (20)



5. Transferencia de ficheros

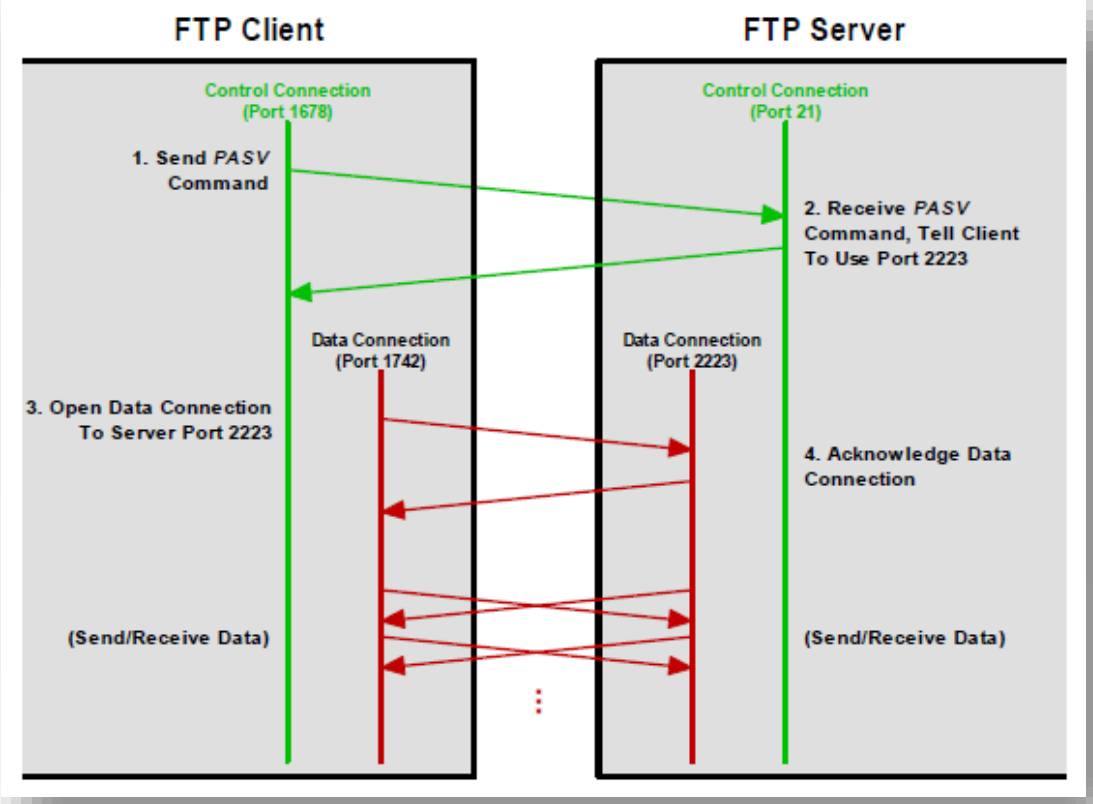
FTP

- FTP pasivo:

El cliente envía el comando PASV para iniciar el modo pasivo, y el servidor selecciona un puerto que manda como respuesta a este comando

En este modo es el cliente el que abre la conexión de transferencia de datos

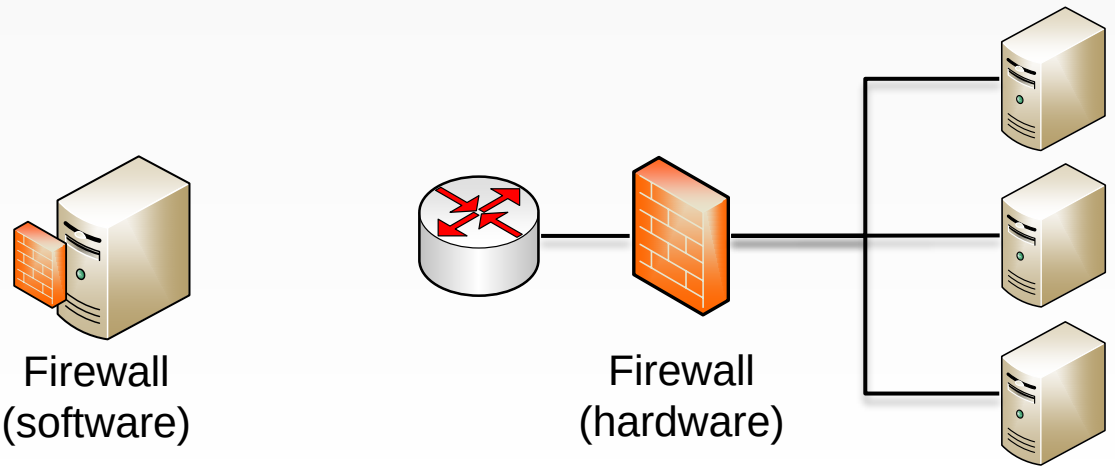
La razón de ser de este modo es debido a las restricciones impuestas por **Firewalls**



5. Transferencia de ficheros

Firewall

- Un firewall (cortafuegos) en un dispositivo de seguridad de red que controla el tráfico entrante y saliente en una red o host
- Tipos de firewall:
  - Software: Más sencillos y económicos (usado para usos domésticos)
  - Hardware: Dispositivos dedicados, más complejos (usado normalmente en empresas)



5. Transferencia de ficheros

Firewall

- Un firewall tiene un conjunto de reglas que permite bloquear o limitar el tráfico entrante y saliente a la red/equipo terminal
  - También es posible limitar el tráfico en base a direcciones IP
- Por ejemplo, `ufw` (*Uncomplicated Firewall*) es un firewall software para Linux. Para instalarlo en Ubuntu usamos el siguiente comando:

```
sudo apt-get install ufw
```

- Su interfaz gráfica se instala en Ubuntu con el siguiente comando

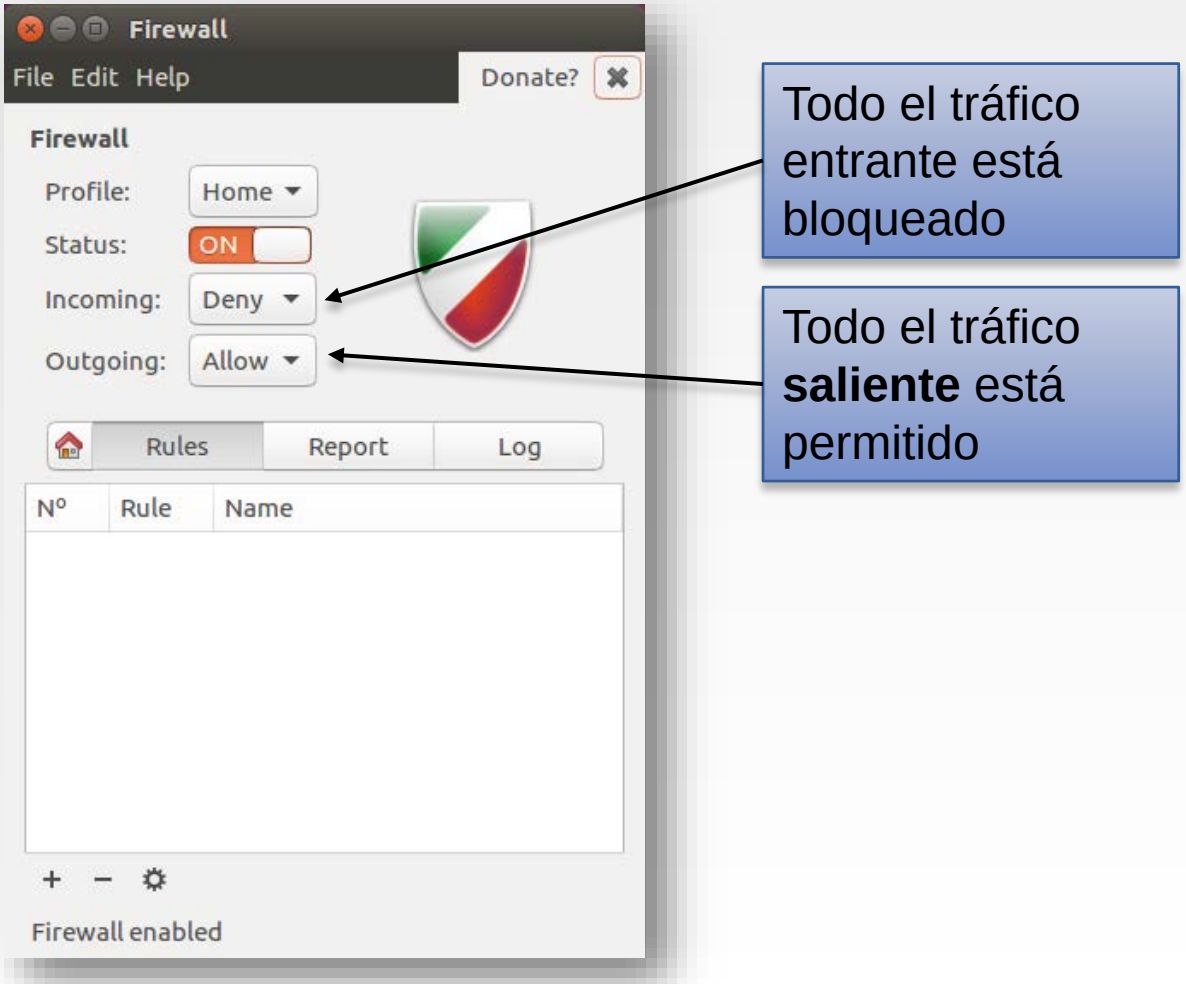
```
sudo apt-get install gufw
```

- Más información en https://help.ubuntu.com/community/UFW

5. Transferencia de ficheros

Firewall

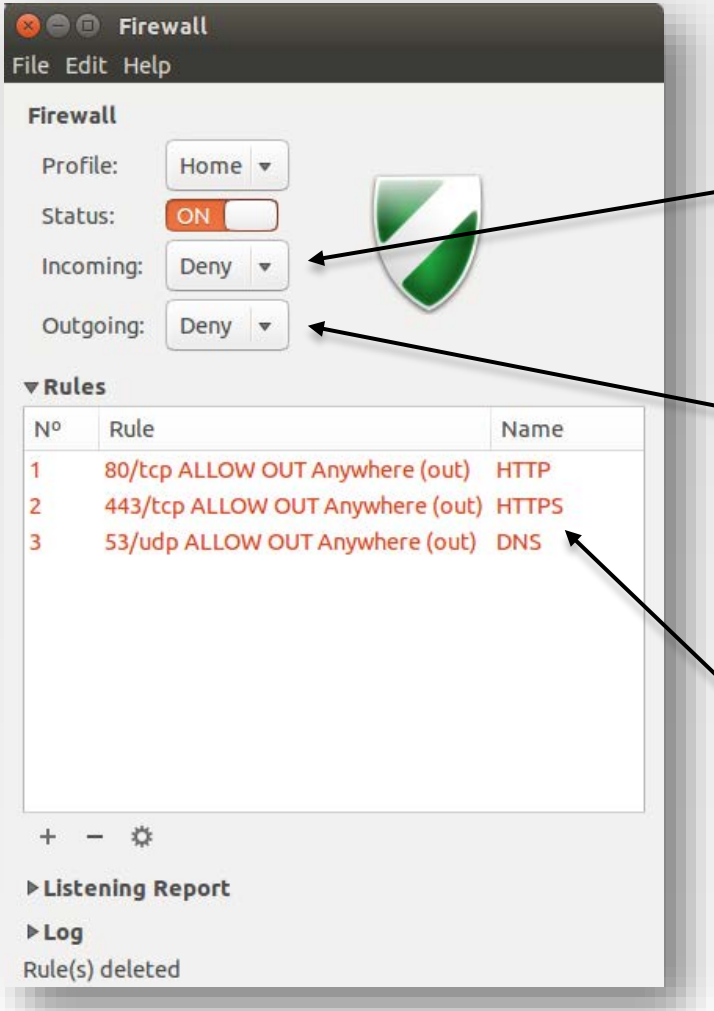
- Ejemplo:



5. Transferencia de ficheros

Firewall

- Ejemplo:



Todo el tráfico entrante está bloqueado

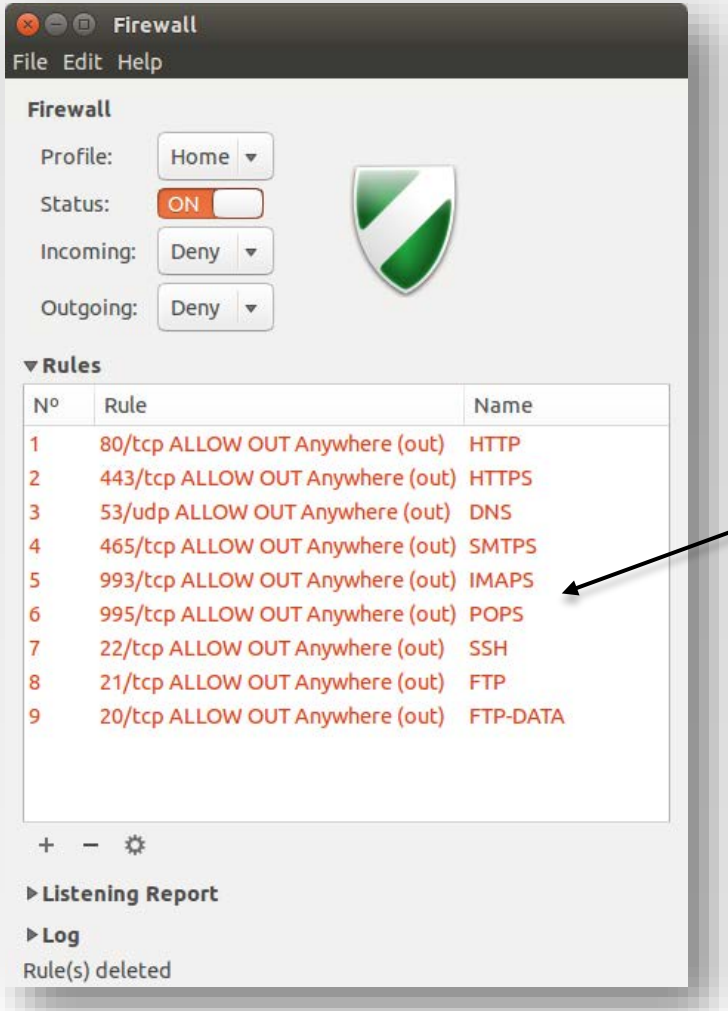
Todo el tráfico **saliente** está bloqueado

Hay algunas excepciones al tráfico saliente: **HTTP**, **HTTPS**, y **DNS**

5. Transferencia de ficheros

Firewall

- Ejemplo:



En esta configuración además se permite el tráfico saliente hacia los protocolos de correo electrónico sobre TLS (SMTPS, IMAPS, POPS), FTP y SSH

5. Transferencia de ficheros

FTP seguro

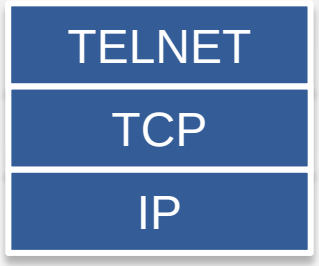
- El principal problema de FTP es la **seguridad** (no ofrece cifrado de la transferencia).
- Las alternativas para proporcionar seguridad a FTP son dos:
  - **SFTP** (*SSH File Transfer Protocol*). Diseñado originalmente por el IETF como extensión de SSH (*Secure SHell*), en la actualidad es un protocolo independiente
  - **FTPS** (FTP SSL). FTP sobre TLS (extensión a FTP para usar una conexión de transporte cifrada usando TLS)

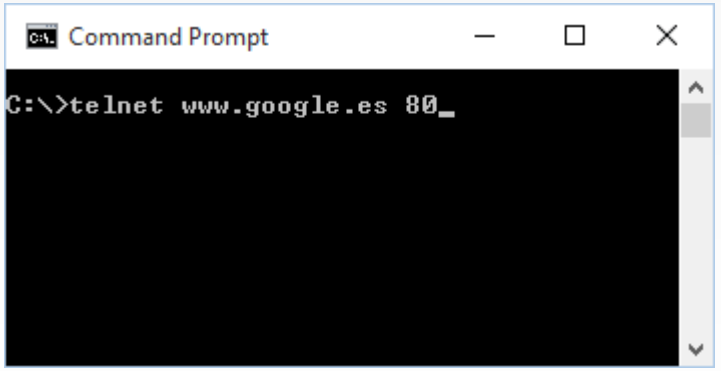
Índice de contenidos

6. Acceso a máquinas remotas

Telnet

- Protocolo de aplicación cliente-servidor para controlar de forma remota un host.
- RFCs 854 y 855
- Puerto TCP en servidor: 23
- Problema: seguridad (credenciales se envían como texto plano)
- Solución: Uso de SSH en su lugar
- Por esta razón no se suele usar Telnet para gestionar de formar remota un host
  - Sí podríamos usar Telnet para operaciones de depuración (establecer conexiones TCP en un determinado puerto)

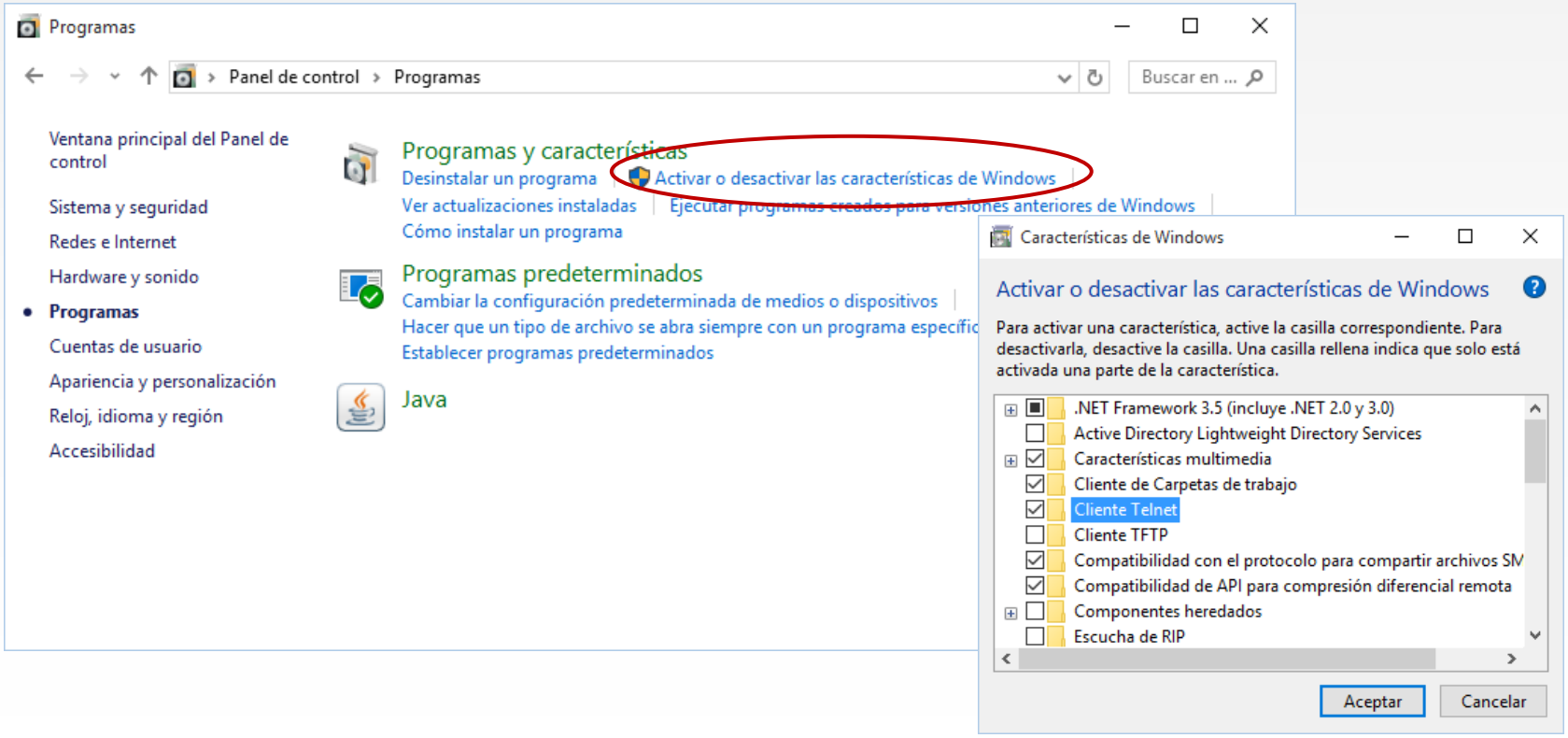




6. Acceso a máquinas remotas

Telnet

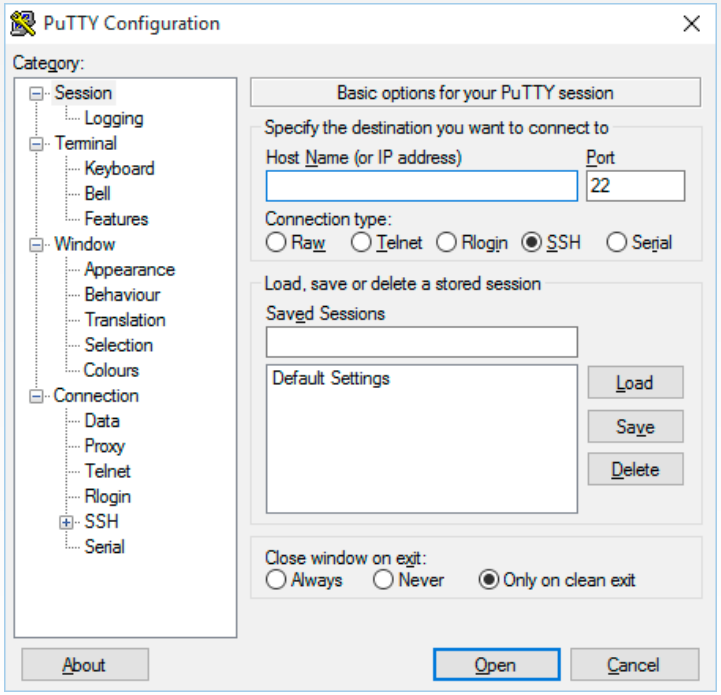
- Para usar Telnet en Windows hay que activarlo (por defecto no lo está):



6. Acceso a máquinas remotas

Telnet

- Alternativa: **putty** (cliente Telnet, SSH, conexiones TCP *raw*)

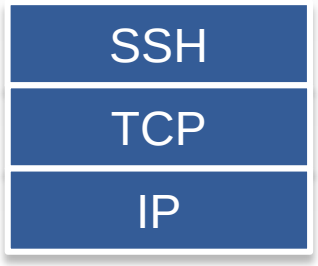


http://www.putty.org/

6. Acceso a máquinas remotas

SSH

- SSH (*Secure SHell*) es un protocolo cliente-servidor para acceder a máquinas remotas de forma segura
- El puerto por defecto de los servidores SSH es el 22
- Existen aplicaciones de consola de comandos en sistemas Unix/Linux que hace uso del protocolo SSH:
  - `ssh`, cliente SSH para acceder a máquinas de forma remota mediante consola de comandos (Shell)
  - `scp` (*Secure CoPy*) es una aplicación de línea de comandos para transferir archivos sobre SSH



6. Acceso a máquinas remotas

SSH

- Al conectarnos a un servidor SSH por primera vez, éste nos envía lo que se conoce como ***fingerprint***
- Se trata de una función resumen (*hash*) de la clave pública del servidor
- Sirve para advertir de posibles ataques de suplantación de intermediario (*man-in-the-middle*)