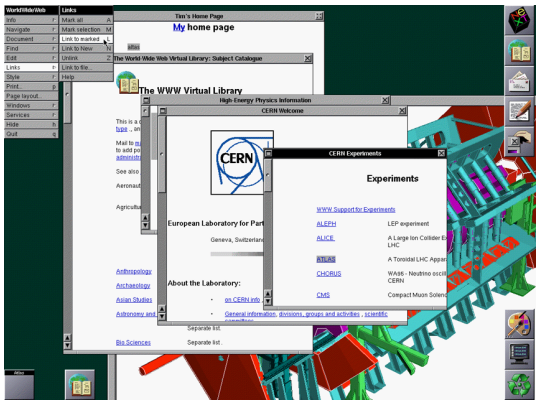
Redes de Datos II
Servicios TCP/IP IIa parte

Luis Marrone

LINTI–UNLP

19 de noviembre de 2021



CC BY NC SA

Contenidos

Estamos en:

Historia I

1965: El Instituto Tecnológico de Massachusetts (MIT) explora el concepto de email

1971: El ingeniero en computación Ray Tomlinson envía el primer mensaje de email, a través de la red:
"QWERTYUIOP"

1971: Tomlinson eligió la arroba, que en inglés se lee "at (en tal lugar)", para especificar el destinatario del mensaje: Fulano en tal lugar

1971: Acto seguido, se envió un mensaje a sí mismo(127.0.0.1) y dio inicio a la era del e-mail

Historia II

1982: Se publicaron las propuestas de correo electrónico de ARPANET como RFC 821 (protocolo de transmisión SMTP) y RFC 822 (formato de mensaje)

1984: CCITT elaboró su recomendación X.400, pero su excesiva complejidad, hace que no se utilice

1997: Microsoft compra Hotmail por alrededor de 400 millones de dólares

2004: Se presentan los emails multimedia tras el congreso MMS World celebrado en Viena

2007: Google lanza Gmail

Estamos en:

Características I

- ✓ Basado en el paradigma cliente – servidor
- ✓ Diferencia con los otros servicios. Se contempla el “delivery” de los mensajes
- ✓ Se recurre al concepto de “spooling”. El usuario coloca una copia del mensaje en el “spool” y luego el cliente en “background” envía los mensajes
- ✓ El cliente barre cada 30’(p. ej.)el spool para detectar nuevos mensajes

Características I

- ✓ Basado en el paradigma cliente – servidor
- ✓ Diferencia con los otros servicios. Se contempla el “delivery” de los mensajes
- ✓ Se recurre al concepto de “spooling”. El usuario coloca una copia del mensaje en el “spool” y luego el cliente en “background” envía los mensajes
- ✓ El cliente barre cada 30’(p. ej.)el spool para detectar nuevos mensajes

Características I

- ✓ Basado en el paradigma cliente – servidor
- ✓ Diferencia con los otros servicios. Se contempla el “delivery” de los mensajes
- ✓ Se recurre al concepto de “spooling”. El usuario coloca una copia del mensaje en el “spool” y luego el cliente en “background” envía los mensajes
- ✓ El cliente barre cada 30’(p. ej.)el spool para detectar nuevos mensajes

Características I

- ✓ Basado en el paradigma cliente – servidor
- ✓ Diferencia con los otros servicios. Se contempla el “delivery” de los mensajes
- ✓ Se recurre al concepto de “spooling”. El usuario coloca una copia del mensaje en el “spool” y luego el cliente en “background” envía los mensajes
- ✓ El cliente barre cada 30’(p. ej.)el spool para detectar nuevos mensajes

Características II

- ✓ El cliente acude al DNS para resolver el nombre del destino en una dirección IP y luego establece una conexión TCP con el servidor
- ✓ Los nombres del servidor son independientes de otros nombres dados a la misma máquina
- ✓ Es decir, el mail enviado a oboe.info.unlp.edu.ar puede ir a un lugar diferente que en el caso de telnet hecho con el mismo nombre
- ✓ El destino queda especificado con la dupla:
  (máquina destino, mailbox)

Características II

- ✓ El cliente acude al DNS para resolver el nombre del destino en una dirección IP y luego establece una conexión TCP con el servidor
- ✓ Los nombres del servidor son independientes de otros nombres dados a la misma máquina
- ✓ Es decir, el mail enviado a oboe.info.unlp.edu.ar puede ir a un lugar diferente que en el caso de telnet hecho con el mismo nombre
- ✓ El destino queda especificado con la dupla:
  (máquina destino, mailbox)

Características II

- ✓ El cliente acude al DNS para resolver el nombre del destino en una dirección IP y luego establece una conexión TCP con el servidor
- ✓ Los nombres del servidor son independientes de otros nombres dados a la misma máquina
- ✓ Es decir, el mail enviado a oboe.info.unlp.edu.ar puede ir a un lugar diferente que en el caso de telnet hecho con el mismo nombre
- ✓ El destino queda especificado con la dupla:
  (máquina destino, mailbox)

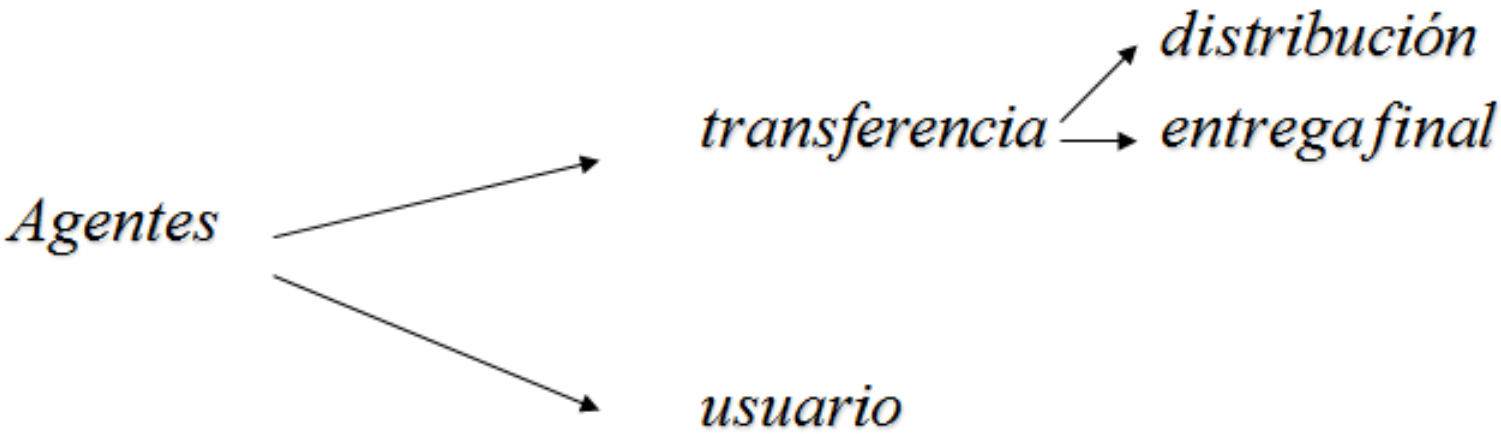
Características II

- ✓ El cliente acude al DNS para resolver el nombre del destino en una dirección IP y luego establece una conexión TCP con el servidor
- ✓ Los nombres del servidor son independientes de otros nombres dados a la misma máquina
- ✓ Es decir, el mail enviado a oboe.info.unlp.edu.ar puede ir a un lugar diferente que en el caso de telnet hecho con el mismo nombre
- ✓ El destino queda especificado con la dupla:
  (máquina destino, mailbox)

Arquitectura

Funciones del sistema de correo: edición de mensajes, transferencia y generación de informes

Subsistemas que lo forman: agentes de transferencia (generalmente "demonios")(**MTAs**) y los agentes de usuario

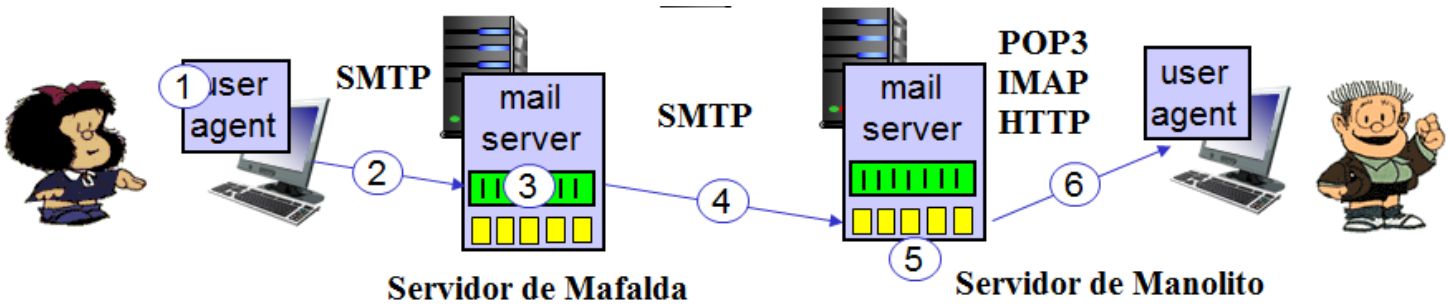


Agentes de transferencia

- ✓ de distribución: utilizando para ello el protocolo SMTP (Simple Mail Transfer Protocol) o SMTP extendido (ESMTP)

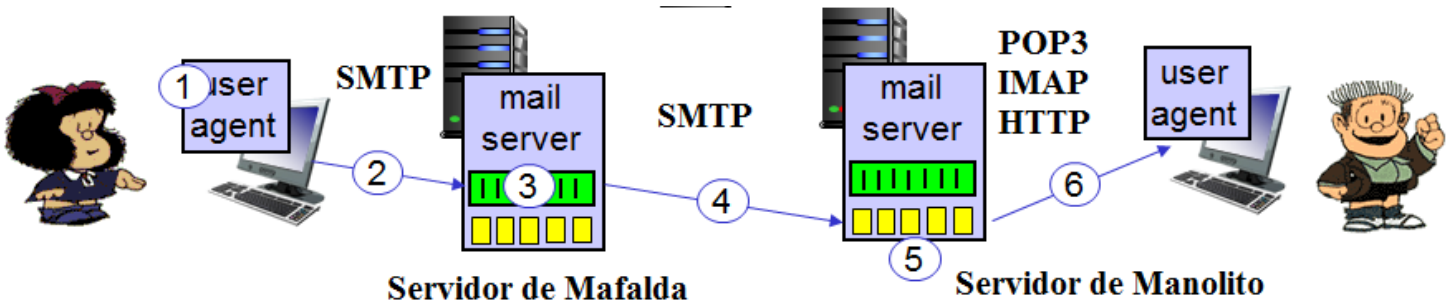
- ✓ de entrega final: que permita al usuario gestionar su correo a través de una máquina remota, utilizando los protocolos POP3 (Post Office Protocol), IMAP (Interactive Mail Access Protocol), o HTTP

Ciclo del email



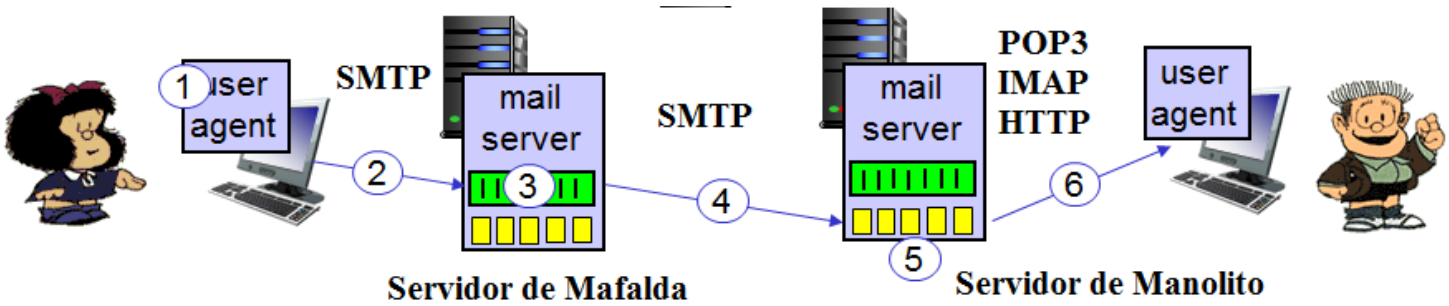
1. Mafalda utiliza su UA para componer el mensaje a manolito@quino.com
2. El UA de Mafalda envía el mensaje a su servidor por SMTP
3. El lado cliente de SMTP establece conexión TCP con el servidor de Manolito
4. SMTP envía el mensaje de Mafalda a través de la sesión TCP
5. El servidor de Manolito coloca el mensaje en la casilla de él
6. Manolito acude a su UA para leer el mensaje

Ciclo del email



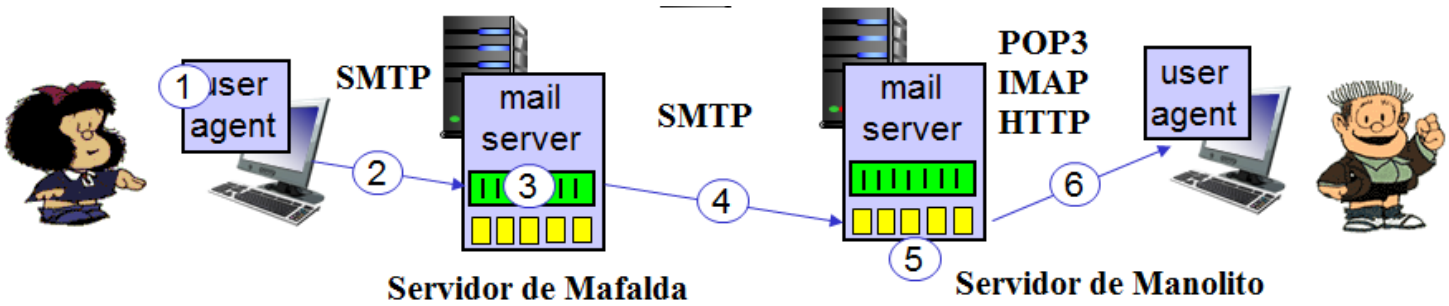
1. Mafalda utiliza su UA para componer el mensaje a manolito@quino.com
2. El UA de Mafalda envía el mensaje a su servidor por SMTP
3. El lado cliente de SMTP establece conexión TCP con el servidor de Manolito
4. SMTP envía el mensaje de Mafalda a través de la sesión TCP
5. El servidor de Manolito coloca el mensaje en la casilla de él
6. Manolito acude a su UA para leer el mensaje

Ciclo del email



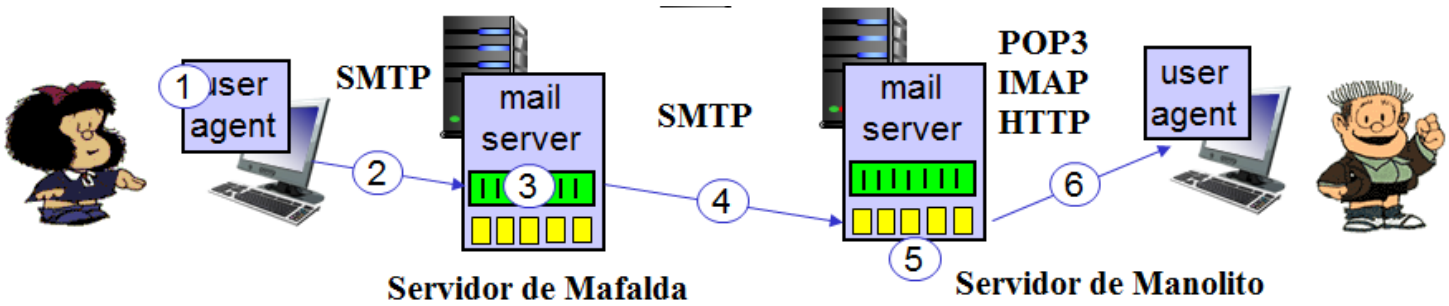
1. Mafalda utiliza su UA para componer el mensaje a manolito@quino.com
2. El UA de Mafalda envía el mensaje a su servidor por SMTP
3. El lado cliente de SMTP establece conexión TCP con el servidor de Manolito
4. SMTP envía el mensaje de Mafalda a través de la sesión TCP
5. El servidor de Manolito coloca el mensaje en la casilla de él
6. Manolito acude a su UA para leer el mensaje

Ciclo del email



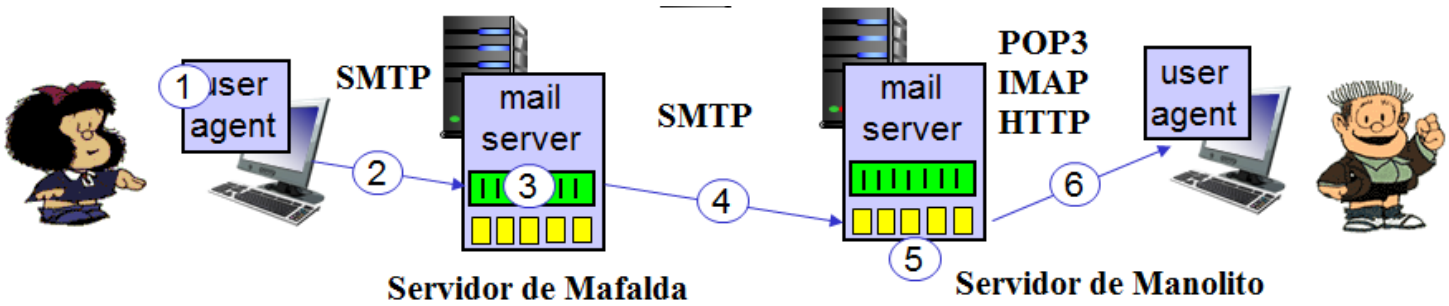
1. Mafalda utiliza su UA para componer el mensaje a manolito@quino.com
2. El UA de Mafalda envía el mensaje a su servidor por SMTP
3. El lado cliente de SMTP establece conexión TCP con el servidor de Manolito
4. SMTP envía el mensaje de Mafalda a través de la sesión TCP
5. El servidor de Manolito coloca el mensaje en la casilla de él
6. Manolito acude a su UA para leer el mensaje

Ciclo del email



1. Mafalda utiliza su UA para componer el mensaje a manolito@quino.com
2. El UA de Mafalda envía el mensaje a su servidor por SMTP
3. El lado cliente de SMTP establece conexión TCP con el servidor de Manolito
4. SMTP envía el mensaje de Mafalda a través de la sesión TCP
5. El servidor de Manolito coloca el mensaje en la casilla de él
6. Manolito acude a su UA para leer el mensaje

Ciclo del email



1. Mafalda utiliza su UA para componer el mensaje a manolito@quino.com
2. El UA de Mafalda envía el mensaje a su servidor por SMTP
3. El lado cliente de SMTP establece conexión TCP con el servidor de Manolito
4. SMTP envía el mensaje de Mafalda a través de la sesión TCP
5. El servidor de Manolito coloca el mensaje en la casilla de él
6. Manolito acude a su UA para leer el mensaje

Funcionamiento

- ✓ El sender abre una conexión TCP contra el port 25 en el receptor
- ✓ El receptor se identifica con “220 <domain> Service Ready”
- ✓ Caso contrario “421<domain> Service not Available”
- ✓ El sender se identifica con el comando Hello <domain>
- ✓ El receptor acepta la identificación con “250 OK”

Funcionamiento

- ✓ El sender abre una conexión TCP contra el port 25 en el receptor
- ✓ El receptor se identifica con “220 <domain> Service Ready”
- ✓ Caso contrario “421<domain> Service not Available”
- ✓ El sender se identifica con el comando Hello <domain>
- ✓ El receptor acepta la identificación con “250 OK”

Funcionamiento

- ✓ El sender abre una conexión TCP contra el port 25 en el receptor
- ✓ El receptor se identifica con “220 <domain> Service Ready”
- ✓ Caso contrario “421<domain> Service not Available”
- ✓ El sender se identifica con el comando Hello <domain>
- ✓ El receptor acepta la identificación con “250 OK”

Funcionamiento

- ✓ El sender abre una conexión TCP contra el port 25 en el receptor
- ✓ El receptor se identifica con “220 <domain> Service Ready”
- ✓ Caso contrario “421<domain> Service not Available”
- ✓ El sender se identifica con el comando Hello <domain>
- ✓ El receptor acepta la identificación con “250 OK”

Funcionamiento

- ✓ El sender abre una conexión TCP contra el port 25 en el receptor
- ✓ El receptor se identifica con “220 <domain> Service Ready”
- ✓ Caso contrario “421<domain> Service not Available”
- ✓ El sender se identifica con el comando Hello <domain>
- ✓ El receptor acepta la identificación con “250 OK”

Transferencia del email

- ✓ El comando mail identifica el generador del mensaje:

```
MAIL FROM:boris_vian@espuma.fr.
250 OK
```

- ✓ Por imposibilidad de ejecutar el comando mandará mensajes con prefijo 451-52/552

- ✓ Por errores en el comando lo hará con 421/500/501

Transferencia del email . . .

- ✓ A continuación el sender identifica todos los receptores del mail con el comando RCPT:

```
RCPT TO:<mrspeel@losvengadores.com>
250 OK
```

- ✓ Si el destino requiere forwarding contestará con 251
- ✓ Si el destino requiere forwarding pero el recepetor no lo hace contestará con 551 y el sender lo enviará directamente
- ✓ Si el destino es incorrecto devuelve 550, etc
- ✓ Este diálogo entre RCPT TO y vuelta de mensaje numérico se tendrá por cada destino especificado en el mail

Transferencia del email . . .

- ✓ El sender utiliza el comando DATA para enviar el mail:

```
DATA
354 Start mail input; end with <crlf>.<crlf>
```

- ✓ El sender envía el mensaje línea por línea

```
Abcdfdfdfdf
Dfdfdfdfdf
<crlf>.<crlf>
 250 OK
```

Transferencia del email . . .

- ✓ El sender envía el comando QUIT y se queda esperando respuesta
- ✓ Si el receptor tiene mensajes para enviar al servidor de Mafalda lo hace como en el caso anterior, si no
- ✓ Acto seguido inicia el cierre de la conexión TCP

Formato del Mensaje

- ✓ RFC 5322
- ✓ El mensaje consta de:
  - Encabezamiento
  - Cuerpo
- ✓ El encabezamiento consta de varias líneas iniciadas por una palabra clave, seguida de : y argumentos:

  `<keyword>:<argumentos>`
- ✓ El cuerpo está separado del encabezamiento por una línea en blanco

No se podía transmitir archivos ejecutables u otro tipo de objetos binarios. Si bien existían esquemas que convertían objetos binarios en texto como uuencode de UNIX no son estándar

SMTP sólo admite ASCII 7 bits con lo que no todo el alfabeto es posible transmitirlo , ñ, acentos, etc.. En los casos que se admiten conversiones no siempre son exitosas.

Multipurpose Internet Mail Extensions I

- ✓ MIME
- ✓ Es una extensión de 5322
- ✓ ¿Por qué una extensión?
  - Archivos ejecutables
  - Lenguajes
  - Multimedia
- ✓ Se definen nuevos campos en el header del mail (5). Proveen información acerca del cuerpo
  - Entre los campos tenemos:
    - MIME-Version : 1.0 Sirve para indicar que sigue RFCs
    - Content-Type: Describe al tipo de información que acarrea el mensaje
    - Content-Transfer-Encoding: Indica la codificación utilizada para representar el cuerpo del mensaje
    - Content-ID

Multipurpose Internet Mail Extensions II

- Content-Description: Provee un segundo nivel de descripción del cuerpo del mensaje. Útiles especialmente cuando el objeto no es leíble
- Los dos últimos son opcionales

✓ Se definen tipos de formato

✓ Se define la codificación para convertir el contenido del cuerpo en un formato aceptado y no alterado por el mail

Tipos MIME

Se definen types (7) y subtypes (14)

Type	Subtype
Text	Plain Richtext
Multipart	Mixed Parallel Alternative Digest
Message	RFC822 Partial External Body

En el texto plano tenemos que los caracterers son ASCII o ISO8859
En el Multipart se indica que el cuerpo contiene partes múltiples e independientes. El campo correspondiente incluye un parámetro “boundary” que indica el separador de las partes.

Mixed: son de diferente tipo.

Parallel: similar al anterior pero no importa el orden en que son enviadas al receptor.

Alternative: Se incluyen diferentes versiones del mismo mensaje

Digest: Las partes constituyen cada una un mensaje completo
Message

Rfc2822: El cuerpo es un mensaje. Pudiendo ser rfc2822 o MIME

Partial: Mensaje fragmentado

External-body: La información no está presente en el cuerpo, sino que contiene un puntero a la información.

Tipos MIME . . .

Type	Subtype
Image	jpeg gif
Video	mpeg
Audio	Basic
Application	PostScript octet stream

Codificación

- ✓ Q = “quoted-printable”, utilizada para poder completar el alfabeto latino, por ejemplo
- ✓ B = “Base-64”, utilizado para codificación de información binaria de modo de pasar sin alteraciones por los diferentes sistemas de correo
- ✓ En este sistema 3 bytes (24 bits) se dividen en 4 grupos de 6 bits que tienen su codificación NVT ASCII
- ✓ En la codificación B se definen 64 caracteres, el último asignado al signo “ = ” se lo utiliza como padding en el caso que los datos a codificar no resulten múltiplos de 24 bits

Ejemplo MIME I

```
From: profesor@unlp.edu.ar
To: alumno@post.unlp.edu.ar
MIME-Version: 1.0
Message-Id: <0123456789.AA00123@unlp.edu.ar>
Content-Type: multipart/mixer; boundary=abcdef
Subject: Mensaje de correo
```

Éste es el preámbulo, el agente de usuario lo ignora

```
--abcdef
Content-Type: text/richtext
```

Aquí va el mensaje de correo en texto

Ejemplo MIME II

```
--abcdef
Content-Type: message/external-body;
access-type=?anon-ftp?,
site=?linti.unlp.edu.ar?,
directory=?pub?,
name=?mensaje.snd?
--abcdef
content-type: audio/basic
content-transfer-encoding: base64
--abcdef
```

Estamos en:

1. Correo Electrónico
   - SMTP – Formato de mensaje
   - POP3
   - IMAP
2. Web – HTTP

Acceso al email

- ✓ Inicialmente, hasta los ’90 el usuario accedía al servidor de correo y con un reader que se ejecutaba en el servidor leía el correo
- ✓ Por otra parte es impracticable instalar un servidor en el usuario
- ✓ SMTP no es posible dado que incluye operaciones de extracción.
- ✓ Se dice que SMTP es un protocolo de “push”
- ✓ Protocolos de acceso utilizados:
  - POP3(Post Office Protocol version 3)
  - IMAP(Interactive Mail Access Protocol)
  - HTTP

Operación

El objetivo del POP3 es obtener correo electrónico del buzón remoto y almacenarlo en la máquina local del usuario para su lectura posterior

- ✓ Apertura de sesión TCP en port 110
- ✓ Fase de autorización(user y pass)
- ✓ Fase de transacción
  - Modalidad “download and delete”(list, retrieve, delete)
  - Modalidad “download and keep”(list, retrieve)
- ✓ Fase de actualización
  - Borra los mensajes, según la modalidad
  - Cierra la sesión (quit)

Ejemplo POP3 I

```
S: <wait for connection on TCP port 110>
C: <open connection>
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C:  APOP mrose c4c9334bac560ecc979e58001b3e22fb
S:  +OK mrose's maildrop has 2 messages (320 octets)
C:  STAT
S:  +OK 2 320
C:  LIST
S:  +OK 2 messages (320 octets)
S:  1 120
S:  2 200
S:  .
C:  RETR 1
S:  +OK 120 octets
S:  <the POP3 server sends message 1>
S:  .
C:  DELE 1
S:  +OK message 1 deleted
```

Ejemplo POP3 II

```
C:  RETR 2
S:  +OK 200 octets
S:  <the POP3 server sends message 2>
S:  .
C:  DELE 2
S:  +OK message 2 deleted
C:  QUIT
S:  +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

Estamos en:

1. Correo Electrónico
   - SMTP – Formato de mensaje
   - POP3
   - IMAP
2. Web – HTTP

Características

- ✓ Paradigma cliente servidor
- ✓ Encapsulado en TCP
- ✓ Port bien conocido: 143
- ✓ Asocia cada mensaje con una carpeta
- ✓ INBOX es la carpeta default
- ✓ Permite crear capetas y desplazar los emails entre ellas
- ✓ Permite descargar componentes parciales del mensaje

Otras Facilidades

- ✓ Pueden incorporar filtros o reglas cuando llega un correo electrónico
- ✓ Pueden reenviar (relay) a una dirección diferente, por ejemplo un teléfono movil con SMS, o a otro servidor de correo
- ✓ Permiten generar una contestación automática, por ejemplo cuando estamos de vacaciones: “Estoy de vacaciones. Regresaré el 15 de Agosto. Que tenga feliz día” Cuando activemos este mecanismo es mejor desuscribirse de las listas de correo, ya que inundaríamos la lista con esta contestación

RFCs(183 Nov. 2020) — email

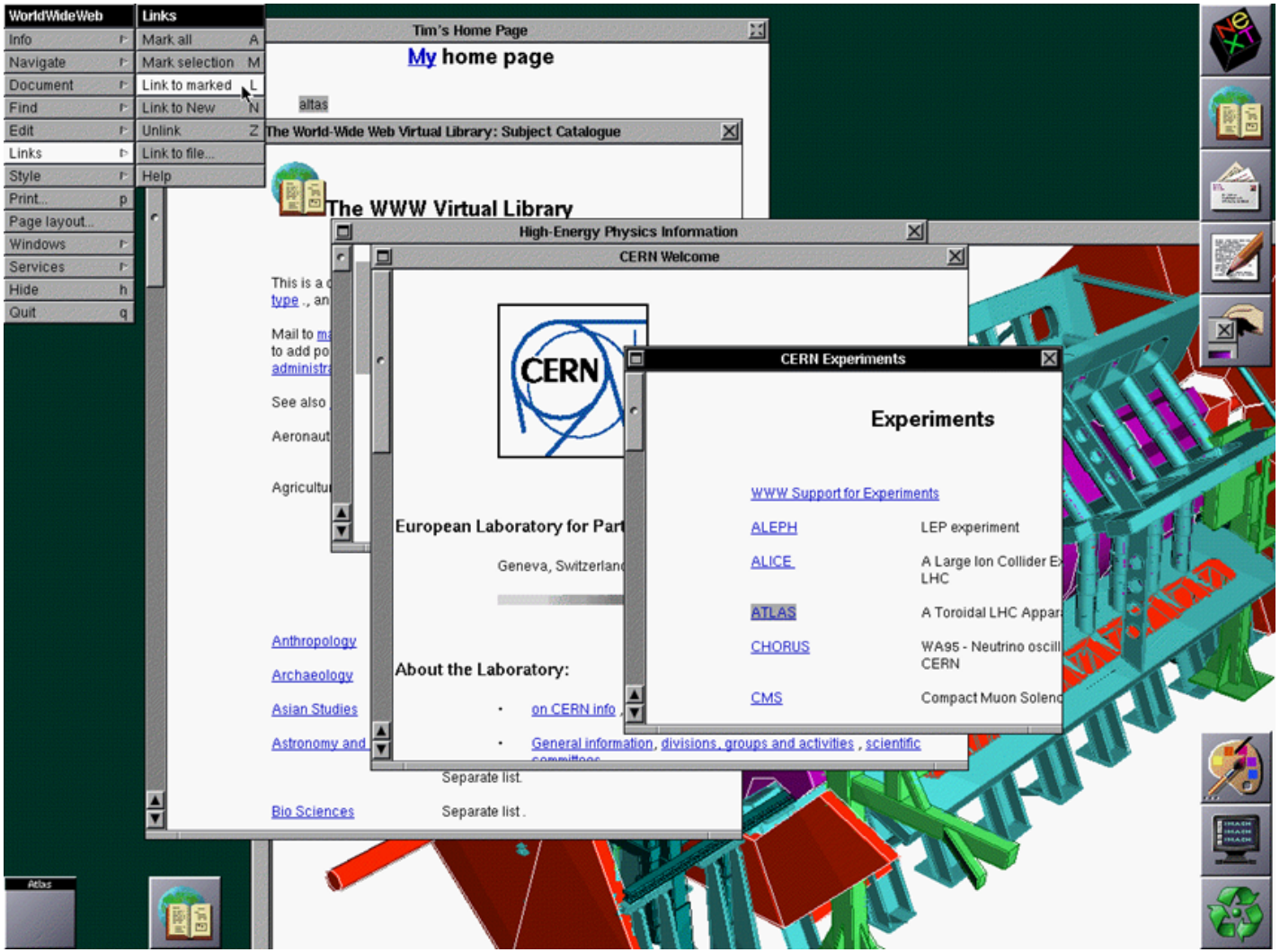
- ✓ RFC 5321: Simple Mail Transfer Protocol. Octubre 2008
  - Previas:821, 2821, STD 10
- ✓ RFC 6531: SMTP Extension for Internationalized Email. Feb. 2012
- ✓ RFC 1939: STD 53: Post Office Protocol - Version 3. Mayo 1996 Actualizada por RFC 2449, RFC 6186
- ✓ RFC 2449: POP3 Extension Mechanism. Noviembre 1998
- ✓ RFC 5034: POP3 SASL Authentication Mechanism. Julio 2007
- ✓ RFC 6856: Post Office Protocol Version 3 (POP3) Support for UTF-8. Marzo 2013
- ✓ RFC 1203: Interactive Mail Access Protocol: Version 3. Febrero 1991
- ✓ RFC 3501: Internet Message Access Protocol - Version 4rev1, Marzo 2003
- ✓ RFC 8514: IMAP SAVEDATE extension, Enero 2019

Estamos en:

1. Correo Electrónico
   - SMTP – Formato de mensaje
   - POP3
   - IMAP

2. Web – HTTP

Primer Web


WorldWideWeb
Info
Navigate
Document
Find
Edit
Links
Style
Print...
Page layout...
Windows
Services
Hide
Quit
Links
Mark all
Mark selection
Link to marked
Link to New
Unlink
Link to file...
Help
Tim's Home Page
My home page
altas
The World-Wide Web Virtual Library: Subject Catalogue
The WWW Virtual Library
High-Energy Physics Information
CERN Welcome
CERN
European Laboratory for Part
Geneva, Switzerlan
About the Laboratory:
on CERN info
General information, divisions, groups and activities , scientific
Anthropology
Archaeology
Asian Studies
Astronomy and
Bio Sciences
Separate list
Separate list .
CERN Experiments
Experiments
WWW Support for Experiments
ALEPH
LEP experiment
ALICE
A Large Ion Collider Ex
LHC
ATLAS
A Toroidal LHC Appar
CHORUS
WA95 - Neutrino oscill
CERN
CMS
Compact Muon Solenc

Historia I

1989: Aparece la primera propuesta “moderna” sobre el Web en el CERN (Centro Europeo de Investigación Nuclear) realizada por el físico Tim Berners-Lee

1993: Los directores del CERN declaran que la tecnología WWW será de libre uso y que no será necesario el pago de licencias

1993: Apareció el interfaz gráfico Mosaic y su autor Marc Andreessen abandonó el NCSA (National Center for Supercomputers Applications) para fundar Netscape Communications Corp

1994: El CERN y el MIT (Massachussets Institute of Technology) crean el W3C (WWW Consortium) para estandarizar servicios e incrementar la interoperabilidad

Historia II

1994: El MIT lleva la parte de EEUU y el centro francés INRIA la parte europea. http://www.w3.org

1995: Aparece el navegador Netscape

1996-1997: HTTP 1.1, HTML 3.0

1998: AOL (America On-Line) y Sun compran Netscape Comm. Corp

1999: XML, RDF

2000: Servicios Web (SOAP, WSDL, UDDI, etc.)

2015: HTTP 2

Servicio Web I

- ✓ Un Web browser es un programa de aplicación que el usuario invoca para acceder y mostrar una página Web
- ✓ El browser es el cliente que accede al servidor de Web para obtener una copia de la página
- ✓ La página web consiste de objetos que pueden ser:
  - archivo HTML
  - imagen JPEG
  - applet Java applet
  - archivo de audio
  - ...
- ✓ Podemos decir que la página Web consta de un archivo base HTML que incluye varios objetos referenciados.

Servicio Web II

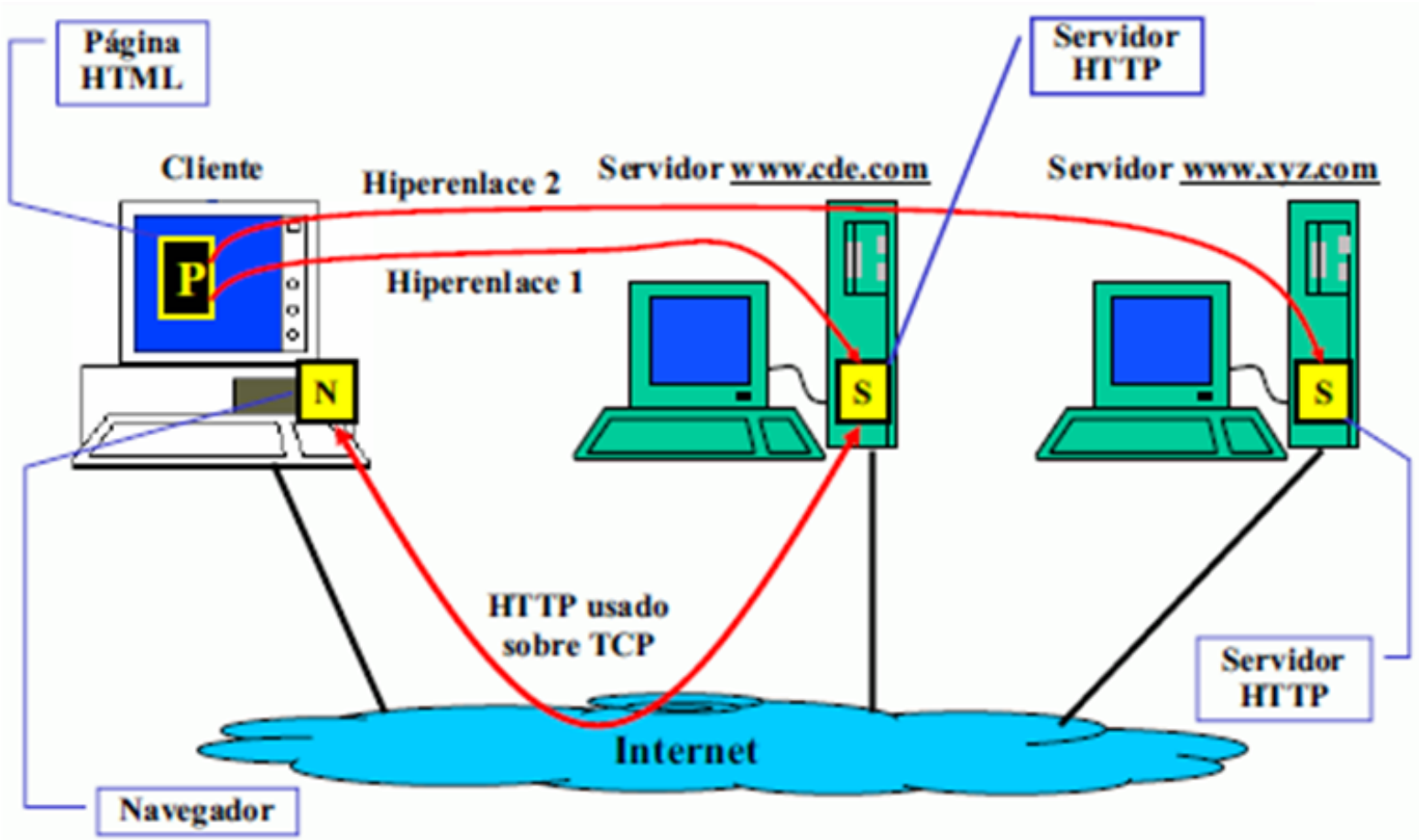
- ✓ Cada objeto es direccionado por una URL (Universal Resource Locator):

$$\underbrace{http}_{protocolo} : \underbrace{//www.linti.unlp.edu.ar}_{host\_name} / \underbrace{archivos/banner\_linti.jpg}_{path\_name}$$

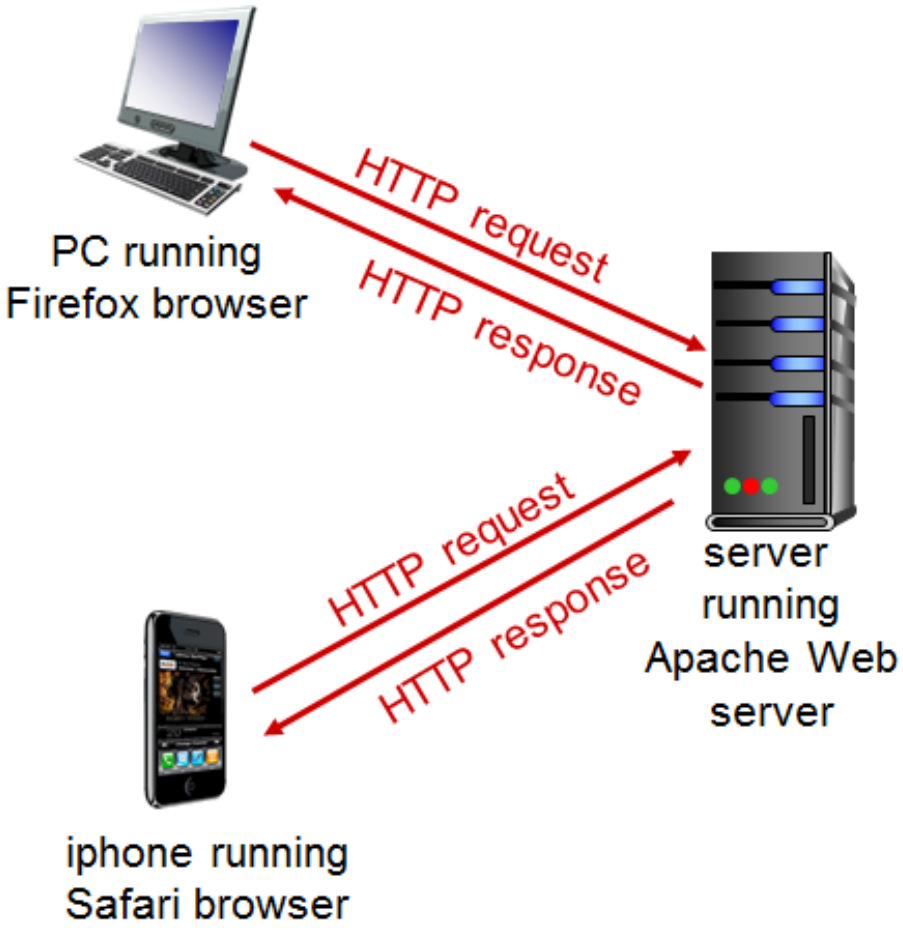
Arquitectura Web

- ✓ Las páginas que apuntan a otras se dice que usan hipertexto
- ✓ Las páginas Web se crean utilizando el lenguaje HTML (HyperText Markup Language) que está basado en etiquetas
- ✓ Para manejar otros tipos de información, el navegador utiliza visores externos adaptados a cada formato
- ✓ Estos visores se denominan conectores (plug-ins)
  Ejemplo: si pinchamos sobre un enlace a un documento PDF, se ejecuta el Acrobat Reader
- ✓ Actualmente el Web es un servicio interactivo y bidireccional gracias al uso de CGIs, Java, PHP/ASP, etc
- ✓ Los navegadores suelen tener cachés locales de disco para ahorrar tiempo en la descarga de páginas ya consultadas

¿Cómo funciona http?



¿Cómo funciona http?...



¿Cómo funciona http?...

Pasos para que el navegador descargue una página:

1. El navegador determina el URL asociado a un enlace de la página HTML. (http://www.w3.org/hypertext/WWW/TheProject.html)
2. El navegador pregunta al DNS por la dirección IP del servidor
3. El DNS responde 18.23.0.23 para www.w3c.org
4. El navegador abre una conexión TCP al puerto 80 de 18.23.0.23
5. El navegador envía el comando
   GET /hypertext/WWW/TheProject.html
6. El servidor Web envía el fichero TheProject.html
7. La conexión se libera (versiones iniciales de HTTP)
8. El navegador presenta en pantalla la página HTML
9. El navegador captura y presenta las imágenes y recursos que aparezcan en la página repitiendo este proceso, cuando dichos recursos están en otros servidores

HTTP

- ✓ El cliente inicia sesión TCP con el servidor en el port 80
- ✓ Se intercambian menmsajes http entre cliente y servidor
- ✓ HTTP es sin estado/memoria. No guarda información de pedidos anteriores
- ✓ Las sesiones/conexiones que establece pueden ser:
  - No persistentes(http 1.0)
  - Persistentes(http 1.1)

Conexiones http No Persistentes, http 1.0

- ✓ Transfiere sólo un objeto por cada sesión
- ✓ Por lo tanto transferir múltiples objetos requiere múltiples conexiones
- ✓ El retardo total es:(n es la cantidad de objetos)

$$\sum_{i}^{n} (2 \times RTT + \texttt{transferencia de objeto})_i$$

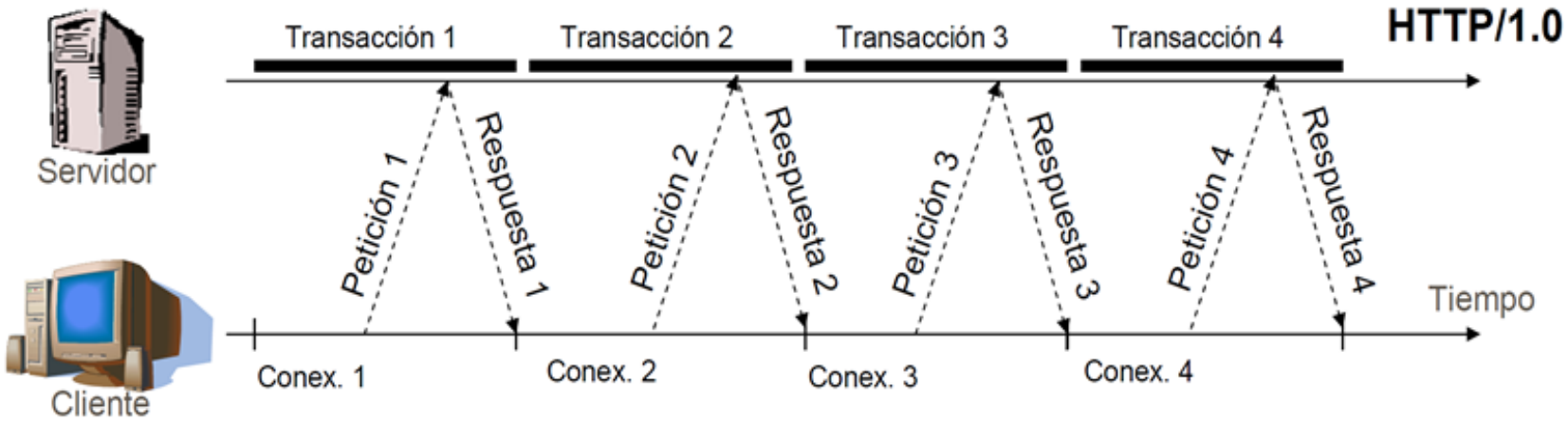
- ✓ Los “browsers” acuden a establecer sesiones en paralelo para reducir el retardo en obtener una página completa

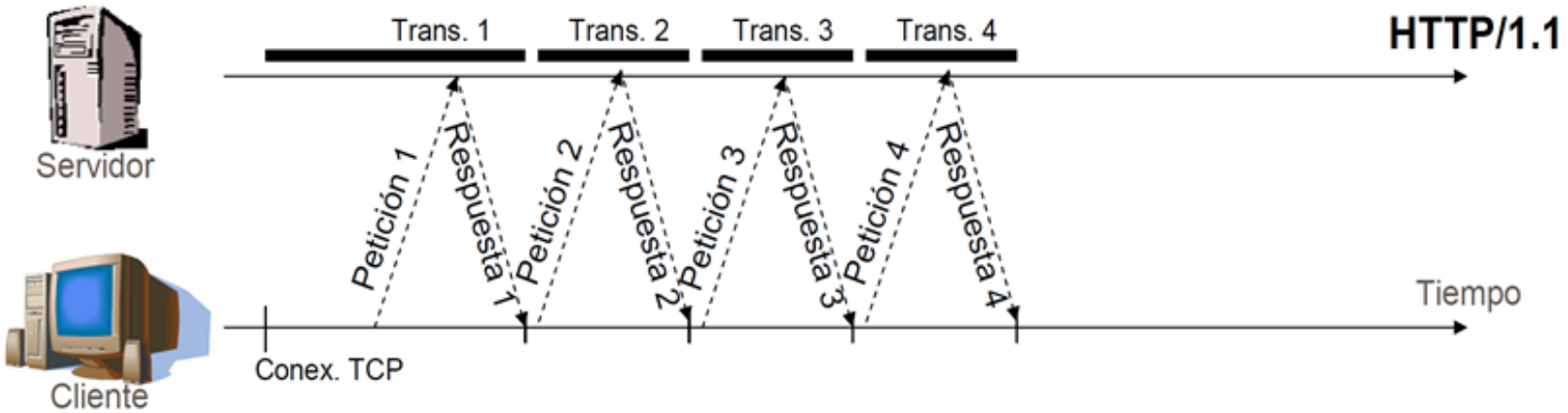
Conexiones http Persistentes, http 1.1

- ✓ Se transfieren múltiples objetos a través de una sola sesión TCP
- ✓ El servidor mantiene la sesión después de haber enviado la respuesta
- ✓ El cliente envía una solicitud no bien encuentra un objeto en la página que recibió
- ✓ El retardo total es:

$$2 \times RTT + \sum_{i}^{n} \texttt{transferencia de objeto}_i$$

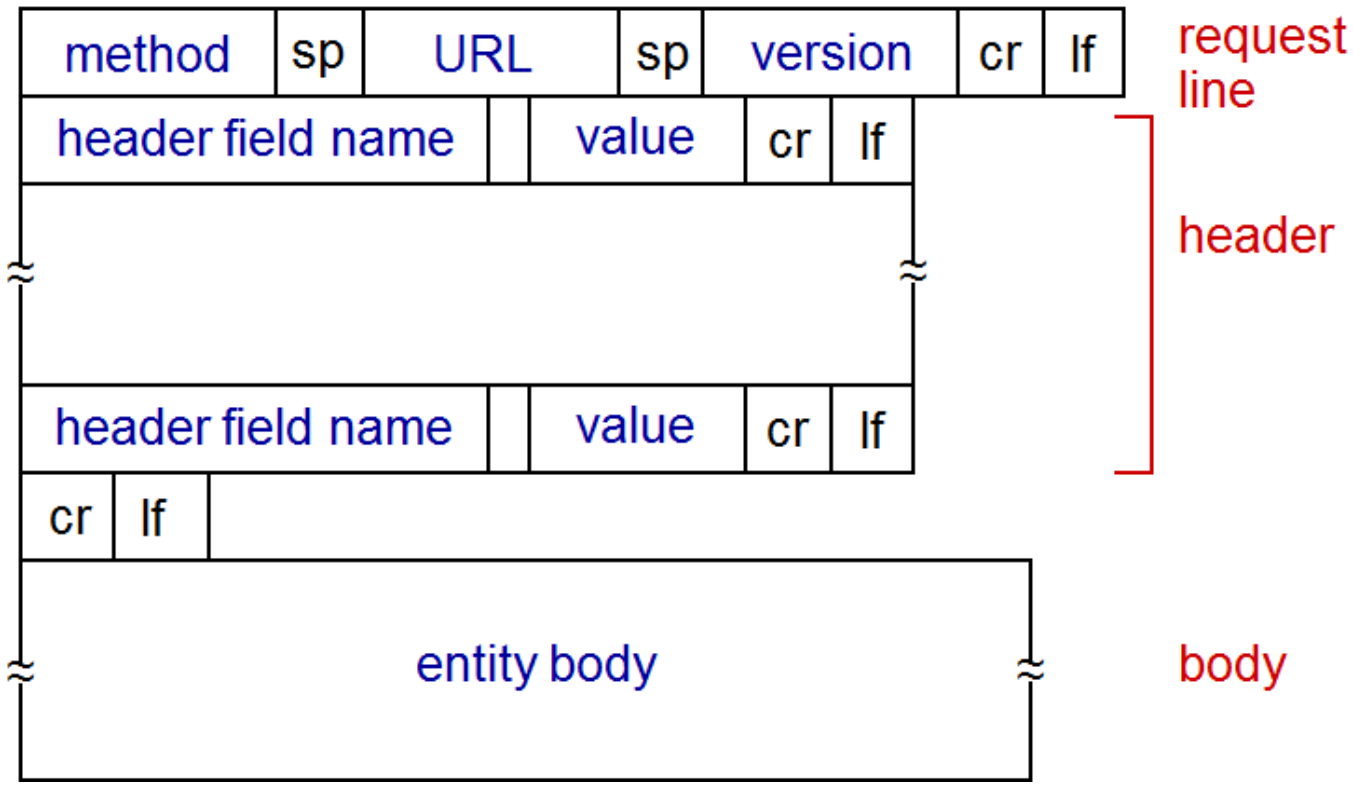
Comparativo





Mensaje Request

Mensaje Request : Formato general



Tipos de Métodos I

GET: Solicita un documento al servidor. Se pueden enviar datos en la URL

HEAD: Similar a GET, pero sólo pide las cabeceras HTTP

- Comprobar enlaces
- Consultar información sobre el archivo antes de solicitarlo

POST: Manda datos al servidor para su procesado

- Similar a GET, pero además envía datos en el cuerpo del mensaje
- La URL corresponde a un página dinámica que trata los datos enviados

PUT: Almacena el documento enviado en el cuerpo del mensaje

DELETE: Elimina el documento referenciado en la URL

Tipos de Métodos II

TRACE: Rastrea los intermediarios por los que pasa la petición

OPTIONS: Averigua los métodos que soporta el servidor

En una caché sólo se guardan las respuestas de las peticiones realizadas con GET y HEAD (POST no)

Método GET

Sintaxis:

- ✓ **GET <URL><VERSION>**
  Solicita el recurso nombrado en la URL
  Recurso estático o dinámico (con o sin parámetros)

- ✓ Variantes (para reducir el trafico en la red):
  - GET condicional
    Baja el recurso sólo bajo ciertas condiciones
    Añadiendo las cabeceras: **If-Modified-Since, If-Match, If-Range**, etc
  - GET parcial
    Baja sólo ciertas partes del recurso
    Añadiendo la cabecera:
    Range: bytes=...

Ejemplo GET

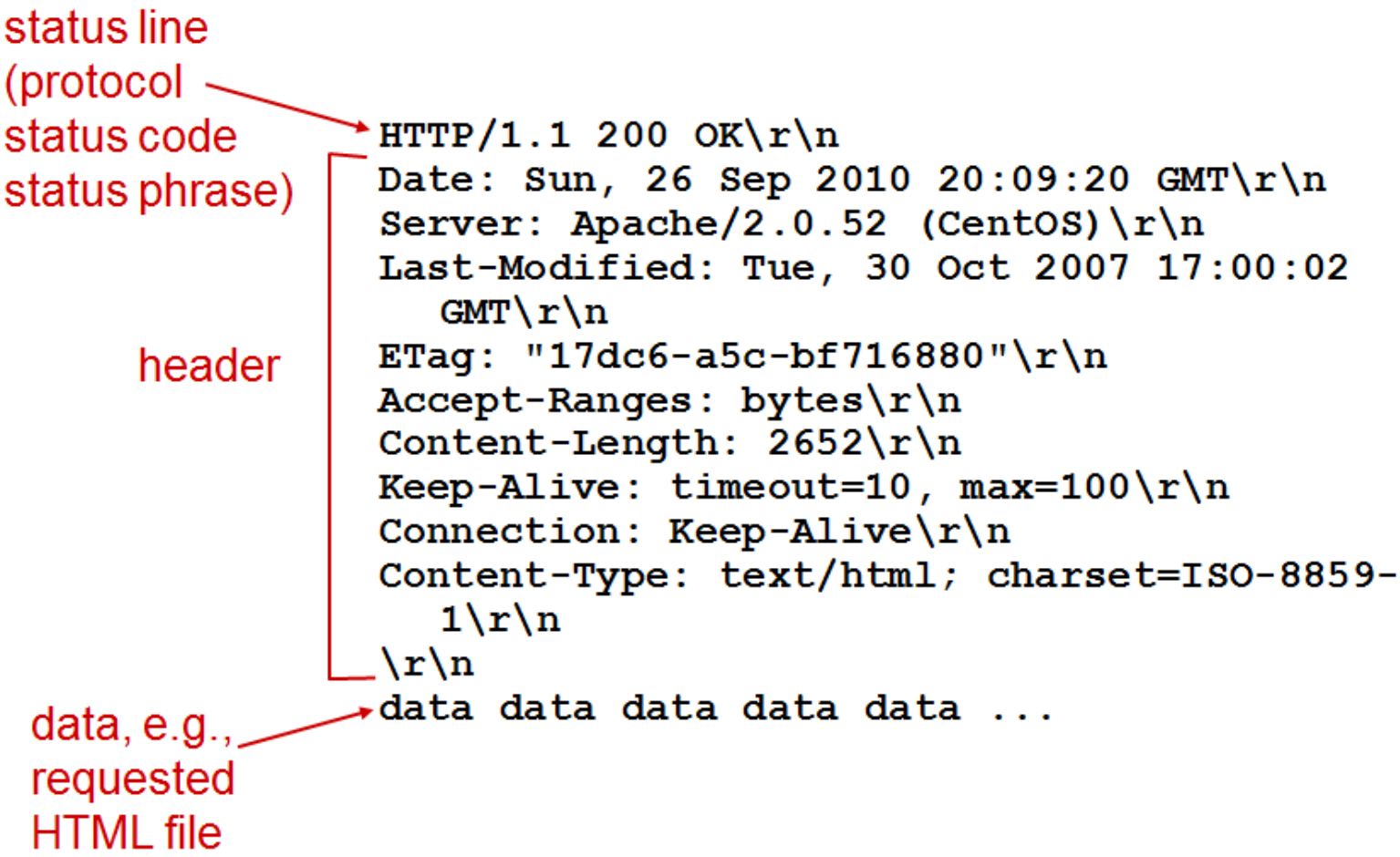
```
GET http://www.unlp.edu.ar/index.html HTTP/1.1
Host: www.unlp.edu.ar
If-Modified-Since: Fri, 22 May 2009 13:53:40 GMT

HTTP/1.0 304 Not Modified
Date: Thu, 21 May 2009 13:55:13 GM
Content-Type: text/html
Expires: Fri, 26 Jun 2009 13:55:13 GMT

GET /Default.htm HTTP/1.1
Host: www.microsoft.com
Range: bytes=0-80

HTTP/1.1 206 Partial content
Server: Microsoft-IIS/5.0
Content-Type: text/html
Content-Length: 81
Content-Range: bytes 0-80/19618
<HTML> ....
```

Mensaje Response



Cookies

- ✓ Las cookies son información que el navegador guarda en memoria o en el disco duro dentro de archivos texto, a solicitud del servidor
- ✓ Incluyen datos generados por el servidor, o datos introducidos en un formulario por el usuario, enviados al servidor y reenviados por éste al cliente
- ✓ HTTP es un protocolo sin estados (no almacena el estado de la sesión entre peticiones sucesivas)
- ✓ Las cookies pueden usarse para asociar estado
- ✓ Proporcionan una manera de conservar cierta información entre peticiones del cliente

Cookies – Usos

- ✓ Almacenar información importante para el servidor
- ✓ Constituyen una potente herramienta empleada por los servidores Web para almacenar y recuperar información acerca de sus visitantes
- ✓ Ejemplos de uso:
  - Guarda información de la sesión
  - Comercio electrónico
  - Carrito de la compra
  - Personalización de páginas
  - Idiomas
  - Seguimiento de las visitas a un Web
  - Carteles publicitarios
  - Almacenamiento del login y password

Almacenamiento de Cookies

- ✓ El hecho de ser almacenadas en el lado del cliente, libera al servidor de una importante carga
- ✓ El cliente devuelve la información al servidor en siguientes peticiones
- ✓ Tipos: cookies de sesión y cookies persistentes
- ✓ Las cookies persistentes son almacenadas en disco por el propio navegador
- ✓ Internet explorer. Un archivo para cada cookie: <identificador de usuario> @ <dominio.txt>

Cookie enviada por el servidor

- ✓ Cabecera HTTP: “Set-Cookie”
  - Cabecera incluida por el servidor en el mensaje de respuesta, cuando quiere enviar una cookie
  - Formato:

    “Set-Cookie:”
    “nombre=valor”: Nombre de la cookie y valor
    ;“expires=fecha”: Fecha de caducidad
    ;“path=camino”: Camino
    ;“domain=dominio”: Dominio
    ;“secure”: sólo se transmite sobre canales seguros (HTTPS)
- ✓ Ejemplo:
  Set-Cookie: unnombre=unvalor; expires=Mon, 30-Jan-2001 12:35:23 GMT; path=/dir; domain=mi.dominio.com; secure

Cookie enviada por el cliente

- ✓ Cabecera HTTP: “Cookie”
  Enviará todas las cookies en una única cabecera HTTP:
  Cookie: nombre1=valor1; nombre2=valor2;...

- ✓ Proceso:
  - Cuando un cliente solicita una URL, buscará en su lista de cookies aquellas que coincidan con ese dominio y con ese camino
  - Dentro de la cabecera “Cookie”, las cookies se ordenan de más a menos específica (según camino)
  - No se consideran las cookies caducadas (de hecho, se eliminan)

Cookies – Limitaciones

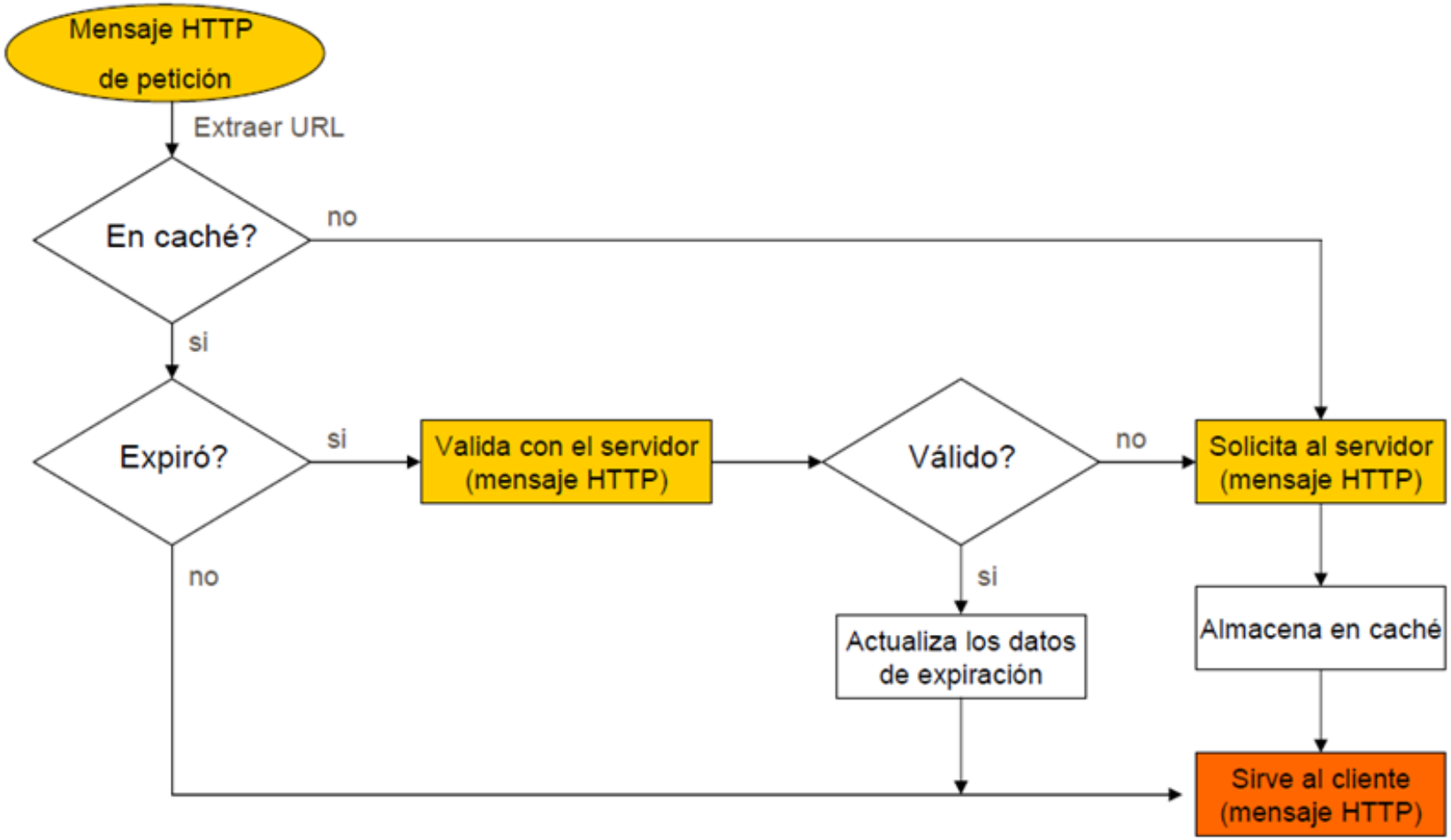
- ✓ Máximo de trescientas cookies en el disco
- ✓ Si llega la número 301, se borra la más antigua
- ✓ Tamaño máximo de 4 Kbytes por cookie (nombre y valor)
- ✓ Veinte cookies máximo por servidor o dominio
- ✓ Ninguna máquina que no encaje en el dominio de la cookie podrá acceder a ella

Cookies – Ejemplo

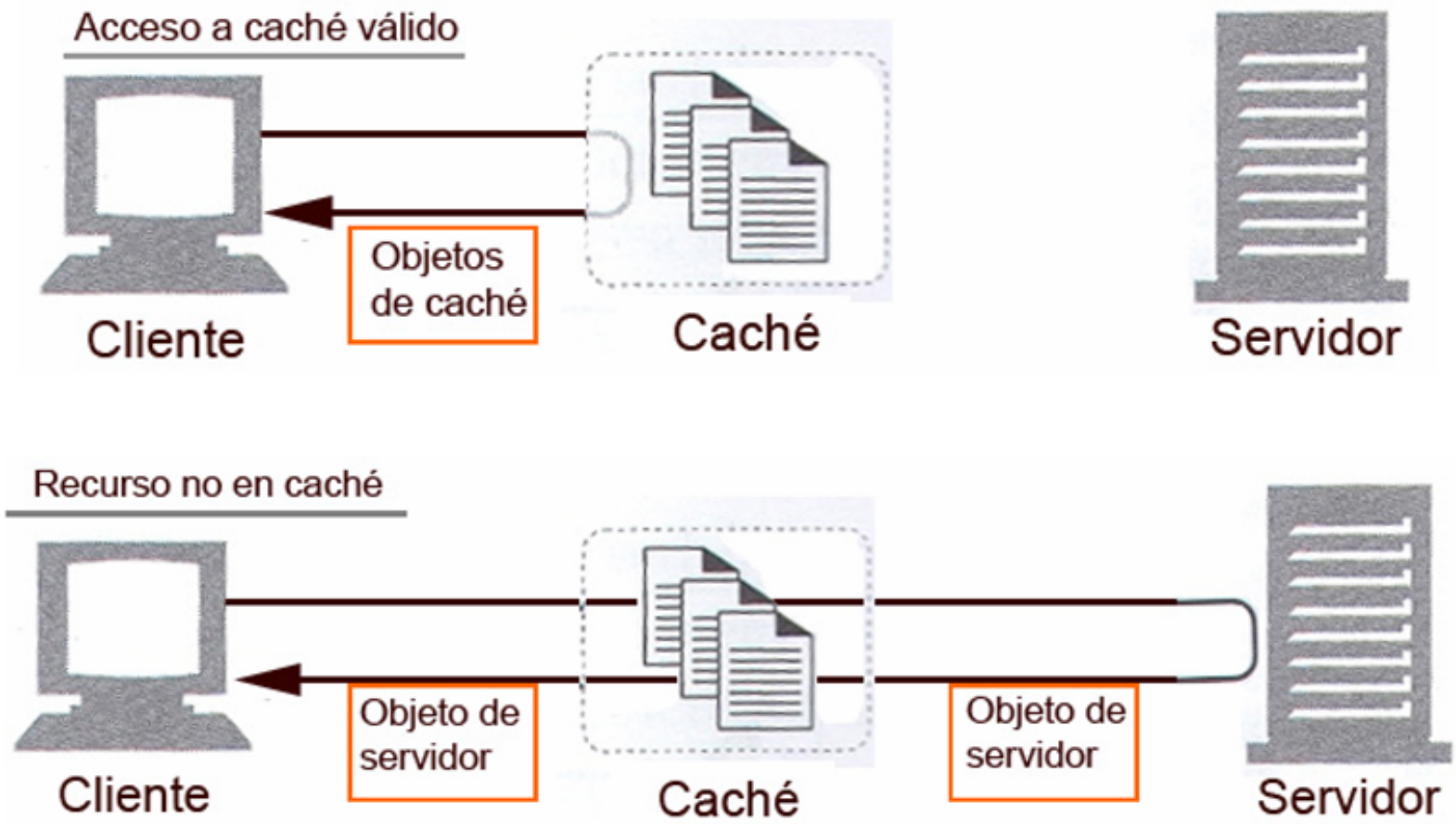
Uso de cachés

- ✓ Almacenan respuestas (susceptibles de ser guardadas) con la intención de reducir el tiempo de respuesta y la carga de la red
- ✓ Necesitan asegurar que los contenidos guardados en la caché sean equivalentes a los almacenados en el servidor
- ✓ Dos modelos:
  - Expiración (consistencia débil)
    Reduce las peticiones al servidor
  - Validación (consistencia fuerte)
    Reduce la cantidad de datos a transmitir
- ✓ Es imprescindible (para la caché) que el cliente identifique al servidor original
- ✓ Campo de la cabecera HTTP: Host

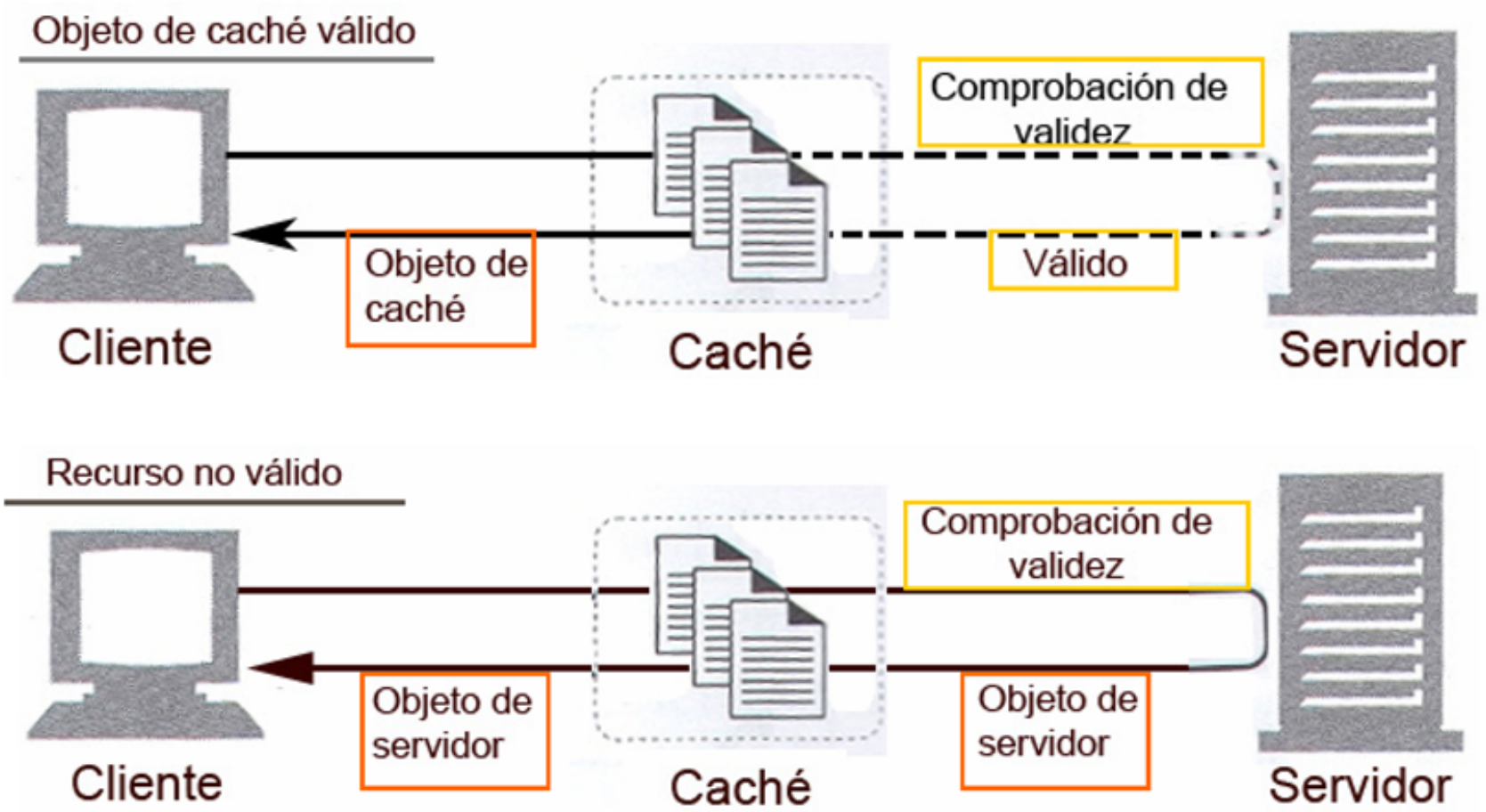
Funcionamiento de cachés



Validación de cachés



Validación de cachés . . .



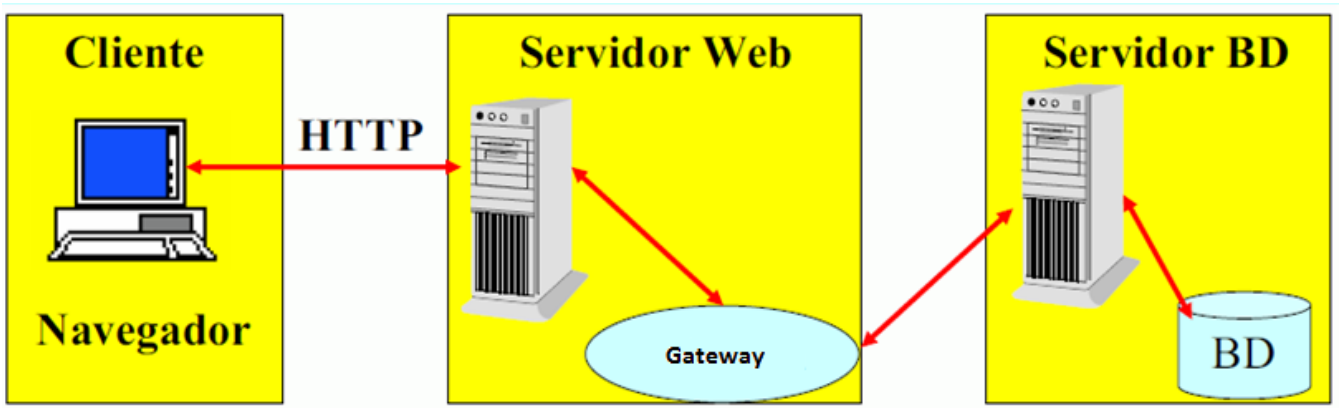
Aplicaciones WEB

- ✓ Las aplicaciones Web permiten construir sistemas de información que funcionan de manera similar a las bases de datos (BD)
- ✓ Permiten generar el contenido de consultas dinámicamente. Esto involucra generalmente BD y tecnologías de acceso a ellas como JDBC (Java Data Base Connectivity), ODBC (Open Data Base Connectivity) o Interfaces de Programación de Aplicaciones (APIs) propietarias
- ✓ La diferencia fundamental con las BD clásicas es que las aplicaciones Web usan como interfaz de usuario un navegador
- ✓ Además, las aplicaciones Web son escalables y siguen el paradigma cliente - servidor de 3 niveles

Aplicaciones WEB . . .

Se reconocen 3 niveles en la arquitectura de las aplicaciones WEB:

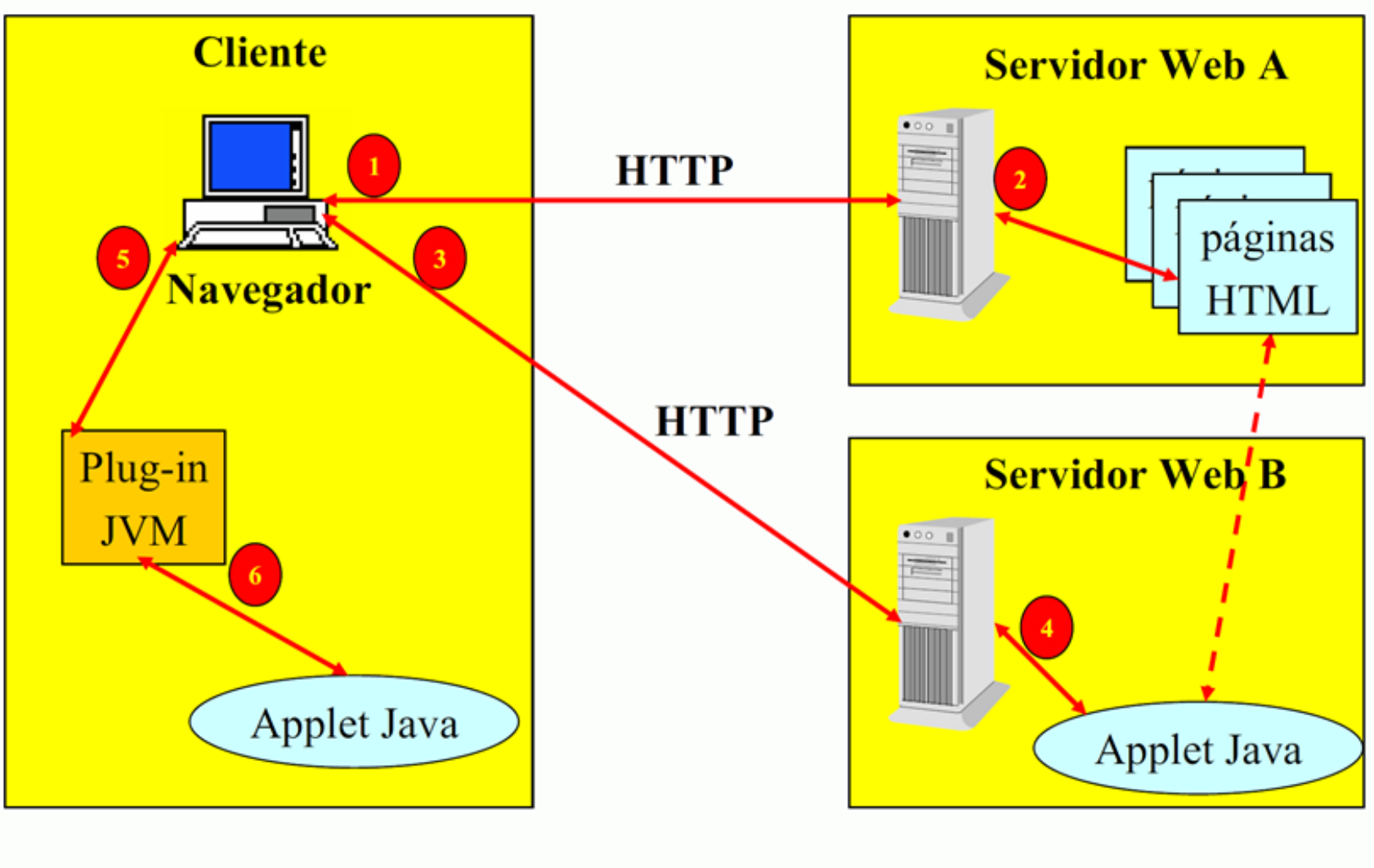
- ✓ El cliente (navegador)
- ✓ El servidor Web con algún tipo de gateway
- ✓ Un servidor (back-end), generalmente un motor de Base de Datos



Extensiones del cliente - Applets

- ✓ Los applets son programas Java que se envían desde el servidor al cliente y son ejecutados por la máquina virtual de Java (JVM)
- ✓ Dicha máquina virtual actúa como un plug-in para el navegador
- ✓ Debido a que es código que se descarga automáticamente por Internet al acceder a una página Web, sin el consentimiento expreso del usuario, dispone de restricciones de seguridad adicionales
- ✓ Los applets descargan al servidor de la ejecución de código y permiten la ejecución distribuida de aplicaciones Web

Applets en acción



Applets paso a paso

1. El navegador accede a una página web en el servidor A
2. El servidor A le entrega la página que apunta a un applet mediante la etiqueta:
   <APPLET CODE= http://www.servidorB.com/MiApplet.class WIDTH=100 HEIGHT=100# ></APPLET>
3. El navegador accede al servidor B donde se encuentra el applet. Normalmente (B = A)
4. El servidor B le entrega el applet al navegador
5. El navegador arranca la máquina virtual JVM (Java Virtual Machine) y le pasa el applet
6. La máquina virtual JVM ejecuta el applet que se muestra en la misma página Web o en una ventana distinta

http / 2

- Mayo 2015.
- Permite un uso eficiente de los recursos de red
- Reduce la latencia a través de compresión de headers
- Intercambio de datos concurrente en la misma conexión.

RFCs(145 Nov. 2021) – HTTP

Number	Title	Authors	Date	Status
RFC 1945	**Hypertext Transfer Protocol -- HTTP/1.0**	T. Berners-Lee, R. Fielding, H. Frystyk	May 1996	Informational
RFC 2616	**Hypertext Transfer Protocol -- HTTP/1.1**	R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee	June 1999	Draft Standard
RFC 2660	**The Secure HyperText Transfer Protocol**	E. Rescorla, A. Schiffman	August 1999	Experimental
RFC 2817	**Upgrading to TLS Within HTTP/1.1**	R. Khare, S. Lawrence	May 2000	Proposed Standard
RFC 7034	**HTTP Header Field X-Frame-Options**	D. Ross, T. Gondrom	October 2013	Informational
RFC 7089	**HTTP Framework for Time-Based Access to Resource States -- Memento**	H. Van de Sompel, M. Nelson, R. Sanderson	December 2013	Informational
RFC 7230	**Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**	R. Fielding, Ed., J. Reschke, Ed.	June 2014	Proposed Standard

RFCs – HTTP

Number	Title	Authors	Date	Status
RFC 7231	**Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**	R. Fielding, Ed., J. Reschke, Ed.	June 2014	Proposed Standard
RFC 7232	**Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests**	R. Fielding, Ed., J. Reschke, Ed.	June 2014	Proposed Standard
RFC 7233	**Hypertext Transfer Protocol (HTTP/1.1): Range Requests**	R. Fielding, Ed., Y. Lafon, Ed., J. Reschke, Ed.	June 2014	Proposed Standard
RFC 7234	**Hypertext Transfer Protocol (HTTP/1.1): Caching**	R. Fielding, Ed., M. Nottingham, Ed., J. Reschke, Ed.	June 2014	Proposed Standard
RFC 7235	**Hypertext Transfer Protocol (HTTP/1.1): Authentication**	R. Fielding, Ed., J. Reschke, Ed.	June 2014	Proposed Standard
RFC 7236	**Initial Hypertext Transfer Protocol (HTTP) Authentication Scheme Registrations**	J. Reschke	June 2014	Informational

RFCs – HTTP

Number	Title	Authors	Date	Status
RFC 7237	**Initial Hypertext Transfer Protocol (HTTP) Method Registrations**	J. Reschke	June 2014	Informational
RFC 7540	**Hypertext Transfer Protocol Version 2 (HTTP/2)**	M. Belshe, R. Peon, M. Thomson, Ed.	May 2015	Proposed Standard
RFC 7541	**HPACK: Header Compression for HTTP/2**	R. Peon, H. Ruellan	May 2015	Proposed Standard
RFC 7617	**The 'Basic' HTTP Authentication Scheme**	J. Reschke	September 2015	Proposed Standard
RFC 7694	**Hypertext Transfer Protocol (HTTP) Client-Initiated Content-Encoding**	J. Reschke	November 2015	Proposed Standard

RFCs – HTTP

Number	Files	Title	Authors	Date	Status
RFC 7711	ASCII, PDF	**PKIX over Secure HTTP (POSH)**	M. Miller, P. Saint-Andre	November 2015	Proposed Standard
RFC 7725	ASCII, PDF	**An HTTP Status Code to Report Legal Obstacles**	T. Bray	February 2016	Proposed Standard
RFC 7804	ASCII, PDF	**Salted Challenge Response HTTP Authentication Mechanism**	A. Melnikov	March 2016	Experimental
RFC 7807	ASCII, PDF	**Problem Details for HTTP APIs**	M. Nottingham, E. Wilde	March 2016	Proposed Standard
RFC 7838	ASCII, PDF	**HTTP Alternative Services**	M. Nottingham, P. McManus, J. Reschke	April 2016	Proposed Standard
RFC 7840	ASCII, PDF	**A Routing Request Extension for the HTTP-Enabled Location Delivery (HELD) Protocol**	J. Winterbottom, H. Tschofenig, L. Liess	May 2016	Proposed Standard
RFC 7975	ASCII, PDF	**Request Routing Redirection Interface for Content Delivery Network (CDN) Interconnection**	B. Niven-Jenkins, Ed., R. van Brandenburg, Ed.	October 2016	Proposed Standard

RFC 8942 HTTP Client Hints - 2021-02

draft-ietf-quic-http-34 Hypertext Transfer Protocol Version 3 (HTTP/3) - 2021-02-02

Esta obra está sujeta a la licencia Atribución-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0) de Creative Commons.

Para detalle de esta licencia visite https://creativecommons.org/licenses/by-nc-sa/4.0/