

# Redes de computadoras

Capa de aplicación

Las diapositivas están basadas en en libro:

“Redes de Computadoras – Un enfoque descendente”  
de James F. Kurose & Keith W. Ross

# Capa de Aplicación

## Temario

Aplicaciones de red

Web y HTTP

FTP

E-Mail

DNS

P2P

Programación de sockets

# ¿Qué es una aplicación de red?

**Programas que ejecutan en equipos conectados en red**

## **Ejemplos:**

- Servidor Web IIS
- Web browser Mozilla Firefox

**En el núcleo de la red  
no hay aplicaciones de usuario.**

# Aplicaciones de red

## Arquitecturas de aplicaciones

- Cliente servidor
- P2P (peer to peer)
- Híbridas cliente servidor / P2P

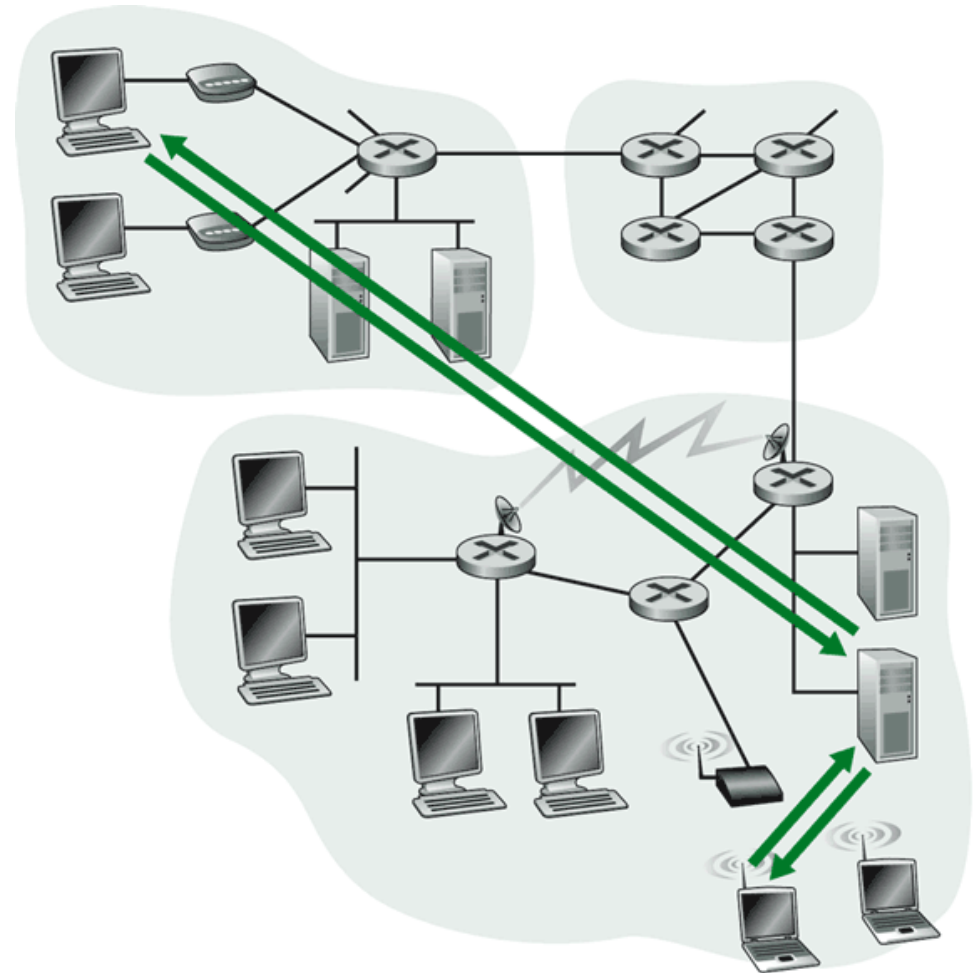
# Arquitectura Cliente servidor

## Servidor

- Equipo de alta disponibilidad
- Dirección IP fija
- Granjas de servidores para escalar

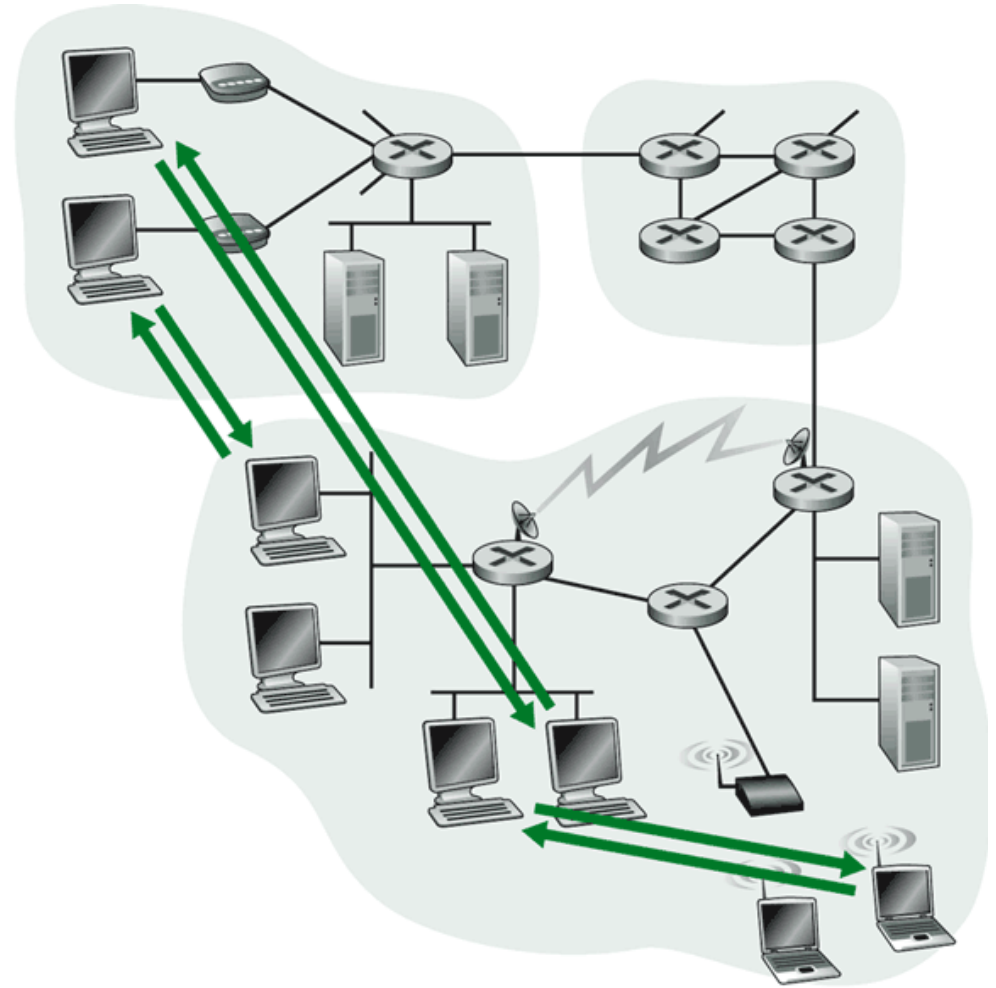
## Cliente

- Se comunica con el servidor
- Uso del servicio intermitentemente
- Dirección IP dinámica
- No se comunica con otros clientes



# Arquitectura P2P

- Servidor de disponibilidad variable
- Se comunican directamente sistemas finales diversos
- Los “peer” se conectan intermitentemente y pueden tener IP dinámica.
- De muy alta escalabilidad pero difícil de administrar



# Arquitectura Híbrida

## Skype

- Aplicación VoIP (Voz sobre IP)
- Servidor centralizado, encuentra las direcciones de los peer remotos.
- Conexión cliente-cliente directa (sin intervención del servidor)

## Mensajería Instantánea

- Conversaciones entre usuarios es P2P
- Servicio centralizado: Presencia de clientes, detección, localización.
  - Usuario se registra con un servidor central
  - Se conecta con el servidor para encontrar contactos

# Comunicación de procesos

## Proceso: Programa ejecutándose en un equipo (host)

- Proceso cliente: proceso que inicia la comunicación
- Proceso servidor: proceso que espera la comunicación de un proceso cliente

En un mismo equipo, los procesos utilizan comunicación inter-procesos (definida por el sistema operativo)

En diferentes equipos los procesos usan intercambio de mensajes.



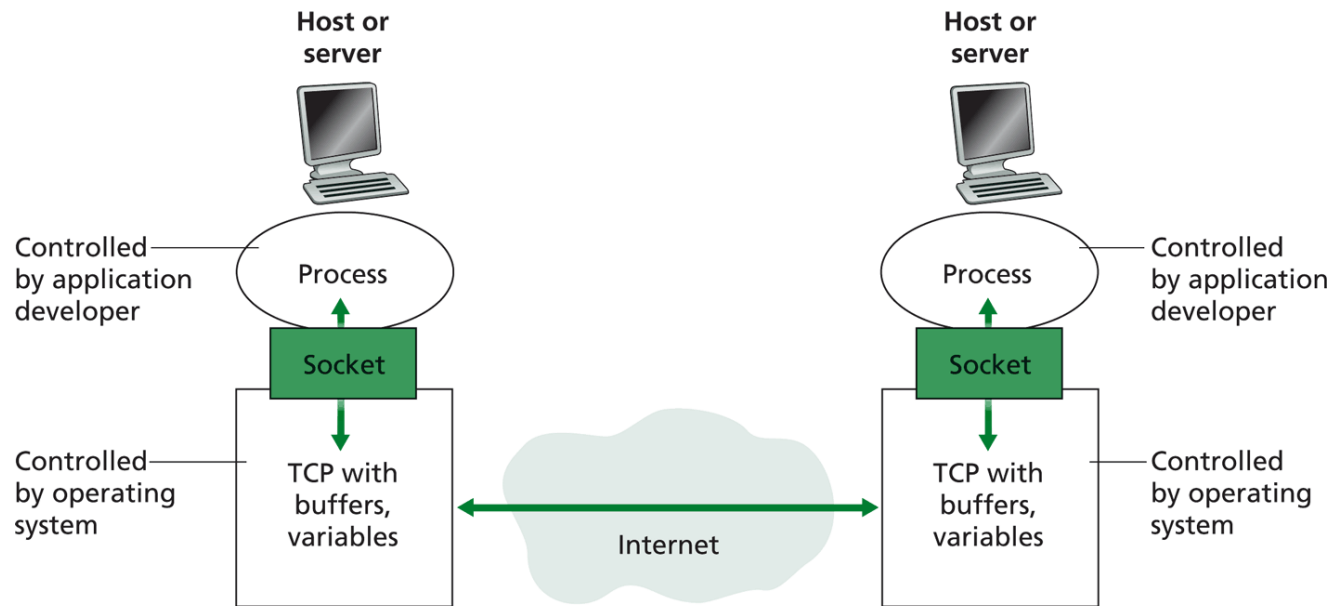
# Sockets

- Los procesos envían-reciben mensajes a través del socket
- El socket se puede pensar como una puerta de comunicación
  - El proceso que envía deja mensajes en la puerta
  - Confía en una infraestructura que se encarga de manejar y dejar el mensaje en el socket del proceso receptor

# Sockets

## Del lado del programador

- Se puede elegir el método de transporte
- Se pueden fijar parámetros para el método de transporte



**Figure 2.3** ♦ Application processes, sockets, and underlying transport protocol

# Identificación de los procesos

## Para recibir mensajes, el proceso debe tener un identificador

- El equipo tiene una única dirección IP de 32 bits
- La dirección IP no es suficiente para identificar el proceso, varios procesos pueden ejecutarse en la misma máquina.

# Identificación de los procesos

**Además de la dirección IP que identifica el equipo, hay números de puertos asociados a cada proceso**

- Servidor HTTP: puerto 80
- Servidor SMTP: puerto 25

Existe una entidad encargada de su asignación  
(IANA Internet Assigned Numbers Authority)

- Puertos bien conocidos – Inferiores al 1024
- Puertos registrados – Utilizados por aplicaciones entre 1024 y 49151
- Puertos dinámicos - Entre los números 49152 y 65535

# Protocolo de capa de aplicación

## Define

- Tipo de mensajes intercambiados
  - request – response
- Sintaxis de los mensajes
  - Qué campos, parámetros y cómo son enviados
- Semántica de los mensajes
  - Qué significa la información en los campos
- Reglas para cómo y cuándo un proceso debe enviar y otro responder a los mensajes

# Protocolo de capa de aplicación

## Protocolos de dominio público

- Definidos en RFC (Request For Comments)
- Permiten interoperabilidad entre procesos de diferentes máquinas
- Ejemplos: HTTP, SMTP, FTP, SSH

## Protocolos propietarios

- Ejemplos: Skype

# Servicios de transporte

## Pérdida de datos

- Se pueden tolerar pérdidas (ej: audio)
- No se pueden tolerar pérdidas (ej: Transferencia de archivos)

## Tiempo

Algunas aplicaciones requieren que no haya retardo (delay) en las transferencias (ej: VoIP)

## Tasa de Transferencia Efectiva (Troughput)

- Algunas aplicaciones requieren una gran tasa de transferencia (ej: video)

## Seguridad

- Encriptación de los datos
- Integridad de los datos

# Servicios de transporte

Aplicación	Pérdida de datos	Throughput	Sensible a retardos
Transferencia de archivos	no	adaptable	no
E-Mail	no	adaptable	no
Páginas web	no	adaptable	no
Audio/video	tolerante	Audio: 5kbps-1mbs Video: 10kbps-5mbps	Sí ~ 100ms
Juegos online	Tolerante	Variable	Sí ~ 100ms



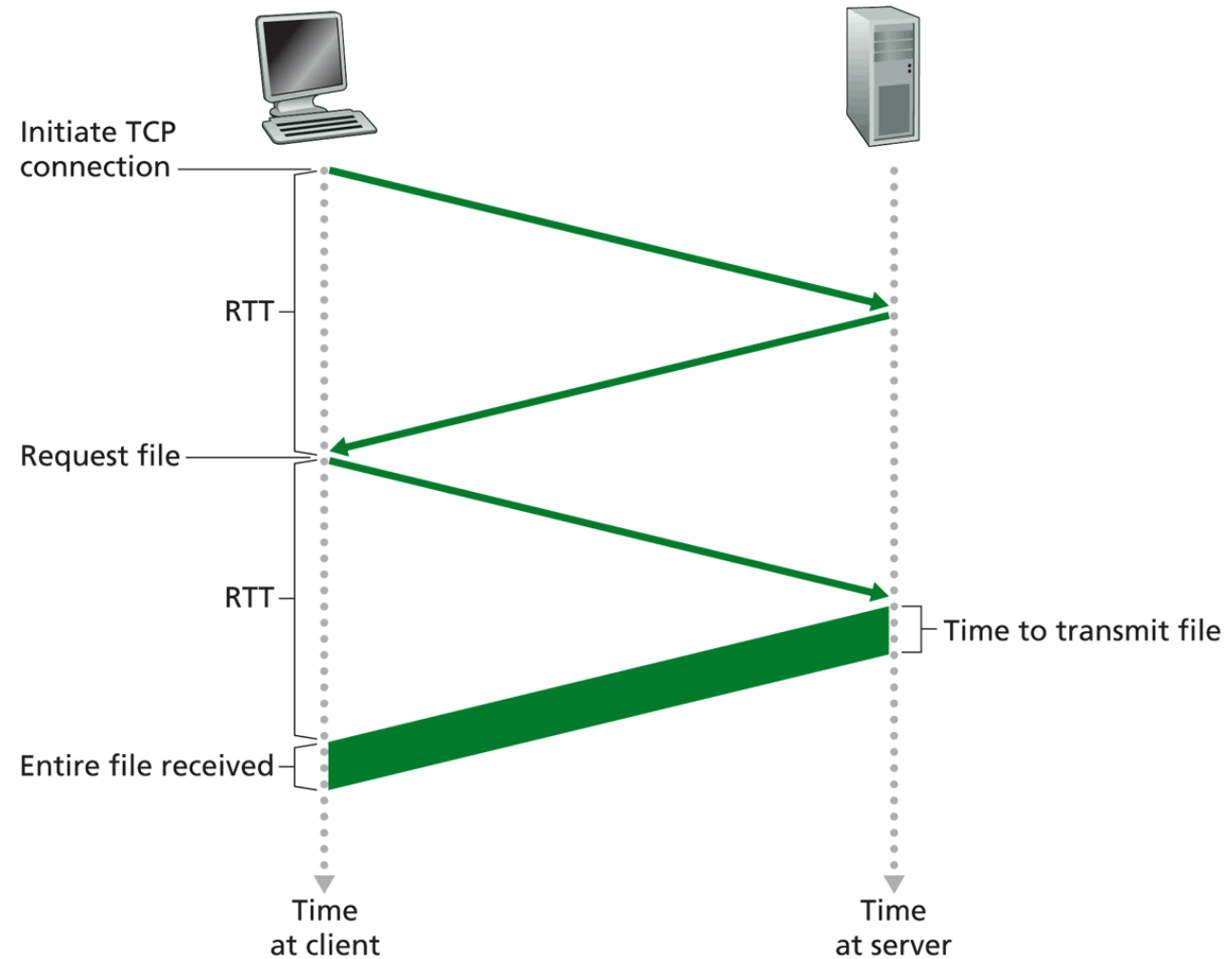
# Servicios de transporte

## Servicios TCP

- Orientado a conexión: hay un establecimiento previo entre los procesos cliente y servidor
- Transporte confiable: Los datos llegan en forma correcta
- Control de flujo: el proceso no envía más de lo que puede aceptar el receptor
- Control de congestión: maneja el envío cuando la red está sobrecargada
- No provee:
  - Control de retardo
  - Asegura o garantiza una mínima tasa de transferencia
  - Seguridad

# Servicios de transporte

## Servicios TCP



**Figure 2.7** ♦ Back-of-the-envelope calculation for requesting an HTML file

# Servicios de transporte

## Servicios UDP

Transferencia de datos no confiable

No provee:

- Establecimiento previo de conexión
- Confiabilidad
- Control de flujo
- Control de congestión
- Control de retardo
- Garantía de tasa de transferencia
- Seguridad

¿Por qué proveer servicios UDP?

# Servicios de transporte

Aplicación	Protocolo de aplicación	Protocolo de transporte
E-mail	SMTP	TCP
Terminal remota	Telnet	TCP
Web	HTTP	TCP
Transferencia de archivos	FTP	TCP
Multimedia	HTTP	TCP-UDP
Telefonía (VoIP)	SIP, RTP, Skype	UDP



**THE HISTORY CHANNEL®**  
**WHERE THE PAST COMES ALIVE™**

# Teoría de conmutación de paquetes

## 1961 – Primer documento sobre la teoría de conmutación de paquetes

Leonard Kleinrock convence a sus colegas de la factibilidad de comunicarse usando paquetes en vez de circuitos.

# Red galáctica



# Orígenes de Internet

## 1962 - Red galáctica

J.C.R. Licklider, del MIT describe un concepto de “Red galáctica”

Un conjunto de ordenadores interconectados globalmente.

Todo el mundo podría acceder rápidamente a datos y programas desde cualquier sitio.

Director del programa de  
investigación  
informática de DARPA.





## Agencia de Proyectos de Investigación Avanzados de Defensa

Agencia de Estados Unidos responsable del desarrollo de nuevas tecnologías para el uso militar.

Surge en el marco de la Guerra Fría y en ella se desarrolló la red ARPANET, origen de Internet.



# ARPANET

## 1967 - Plan para “ARPANET”

Roberts ingresa en DARPA para desarrollar el concepto de redes informáticas

**NPL** (National Physical Laboratory)

**RAND** (Research AND Development)

**MIT** (Massachusetts Institute of Technology)

trabajan en paralelo en un mismo concepto sin saberlo.

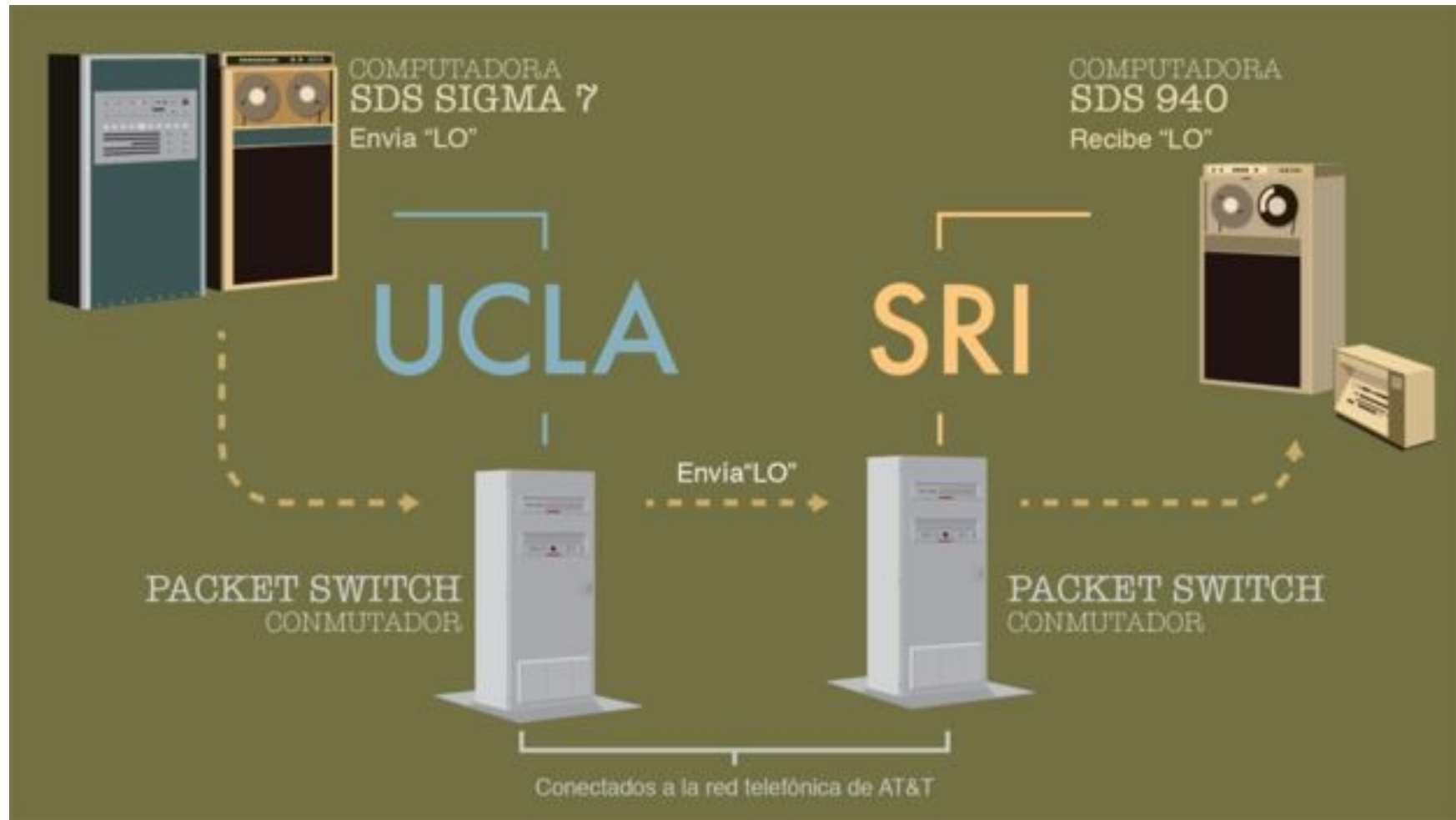
Se adopta el término paquete del trabajo del NPL y la velocidad en el diseño de ARPANET pasó de 2,4 kbps a 50kbps.

## 1969 – Primer comunicación

A modo de experimento se realizó una comunicación mediante línea telefónica conmutada creando la primer red de área amplia (WAN) de la historia.

Se conectó el ordenador TX-2, en Massachusetts, con el Q-32, en California, enviando el mensaje... **“LO”**

# Primer comunicación – LO...



# Internet en 1969



# RFC

## 1969 – Steve Crocker publica el primer RFC “Host Software”

<https://tools.ietf.org/html/rfc1>

Publicación que describe protocolos,  
procedimientos y comentarios

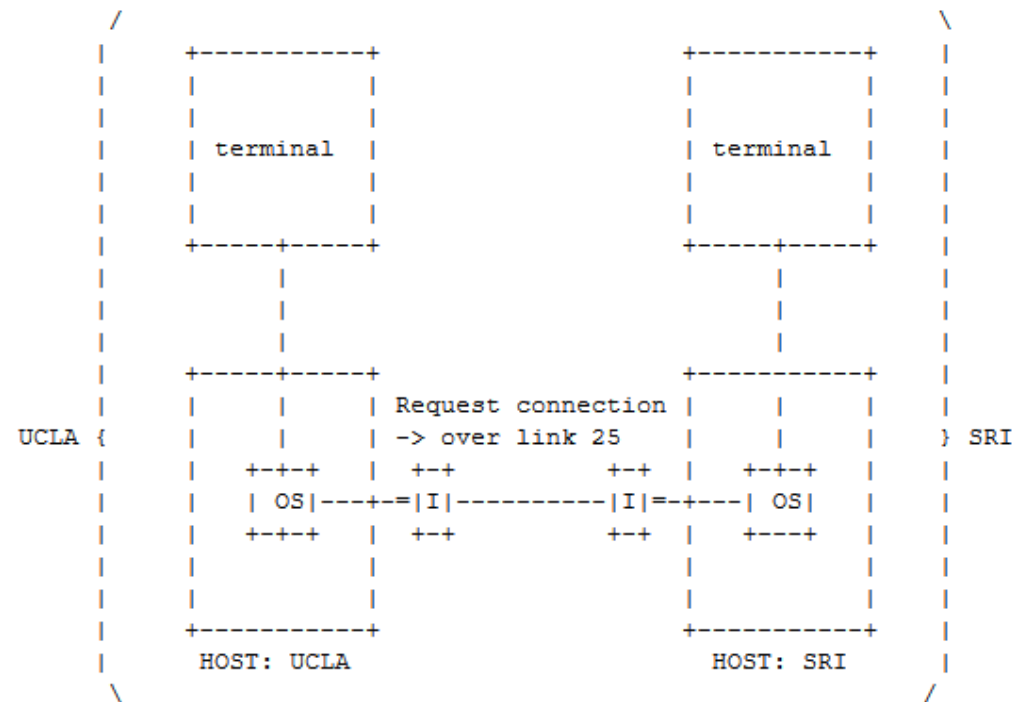
e ideas sobre Internet

y las redes de computadoras.

“Petición de comentarios”

## 30 años de RFC's (1999)

<https://tools.ietf.org/html/rfc2555>



# E-mail

## 1971 – Ray Tomlinson envía el primer correo electrónico.

Implementación del primer sistema de mensajería para intercambiar mensajes a través de la red hacia otros usuarios.



```
Electronic Mail Mailbox          <12,040> NEW ELECTRONIC MAIL MESSAGES!

[ ] from: Ray Tomlinson >> "Hit me back if you get this."
[ ]   to: Ray Tomlinson >> "OMG! I TOTALLY DID! WHAT IS THIS?"
[ ] from: Ray Tomlinson >> "It is like an electronic version of mai..."
[ ] from: intern          >> "Hey U guys need 2 see this funny kitte..."
[ ] from: Ray Tomlinson >> "HAHAHA! But that's really not what this is..."
[#] from: cookebpgdl1     >> "Cheap Phrama Only Today! \./!/\<,K/< \ !!!!!!!!"
[#] from: no-reply        >> "Enhancing your rod will be the best wa..."
[#] from: real mom        >> "Local mom makes $22.80 in ONE WEEK!! "
[ ]   to: Ray Tomlinson >> "What's going on?"
[ ] from: Ray Tomlinson >> "I'm not sure. I can't stop it. :<"
[ ]   to: Ray Tomlinson >> "HA! That looked like face frowning side..."
[#] from: eqefq sweik     >> "Be a sperm-man with Shitloads of Sperms!"
[#] from: cialis4cheep    >> "you p3ni$ will be very gratefull!"
[ ]   to: Ray Tomlinson >> "I hate you."
[ ] from: Ray Tomlinson >> "I hate me too."

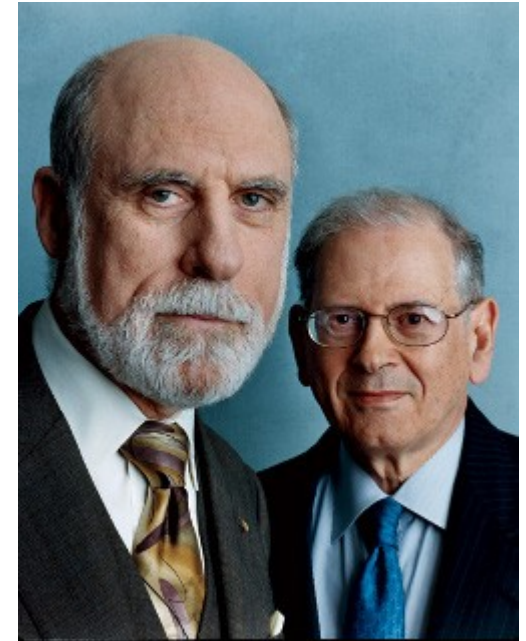
[O] open [R] new [F] mark as delicious meat          1971-09-16T23:59:58.75
```

# Desarrollo de TCP

## 1973 – Vinton Cerf y Robert Kahn diseñan el protocolo TCP

1974 lo publican con ayuda de  
Yogen Dalal y Carl Sunshine en el RFC 675

Algunas personas los consideran  
los inventores de internet.





# Primera red comercial

## 1973 – Ethernet

Bob Metcalfe desarrolla la idea de Ethernet.

## 1974 – Telnet

Una versión comercial de ARPANET es introducida.

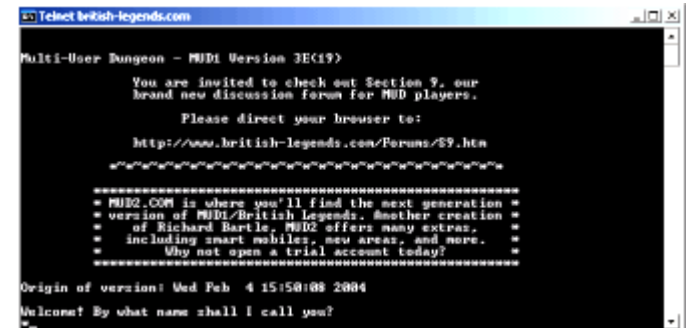
Se considera como el primer ISP.

## 1978 – TCP/IP

TCP se divide en TCP/IP

## 1978 - MUD1

Primer juego multijugador basado en texto (Multi-User Dungeon)



# Primera red comercial

## 1978 – SPAM

La primer documentación de spam.  
Gary Therk envió a 393 personas de la ARPAnet  
promocionando una nueva línea de computadoras.

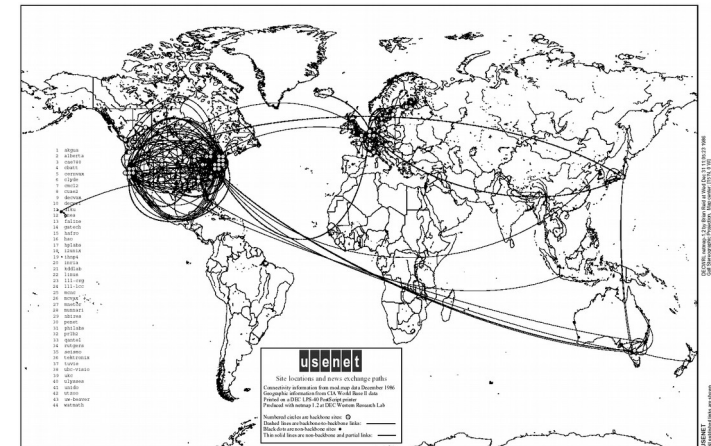


## 1979 – USENET

Dos estudiantes desarrollan la idea de una red de discusión. Sistema precursor de los foros.

## 1984 – Se introduce el sistema DNS

El primer dominio es registrado en 1985 symbolics.com



## 1988 – IRC (Internet Relay Chat)

## 1989 – Primer ISP comercial via dial-up “The Worlud”

# HTML

## **1990 – HTML Hyper Text Markup Language**

Tim Berners-Lee mientras trabajaba en el CERN desarrolló el lenguaje para la elaboración de páginas web.

Define la estructura básica y definición del contenido.

## **1991 – WWW The World Wide Web**

## **1993 – Primer navegador gráfico**

Mosaic desarrollado por NCSA, un año más tarde surgía Netscape

## **1995 – Java y Javascript (LiveScript)**

## **1997 – Google**

## **1998 – Napster**

## **2004 - Facebook**

# Enlaces

## Primer dominio registrado

<http://symbolics.com/>

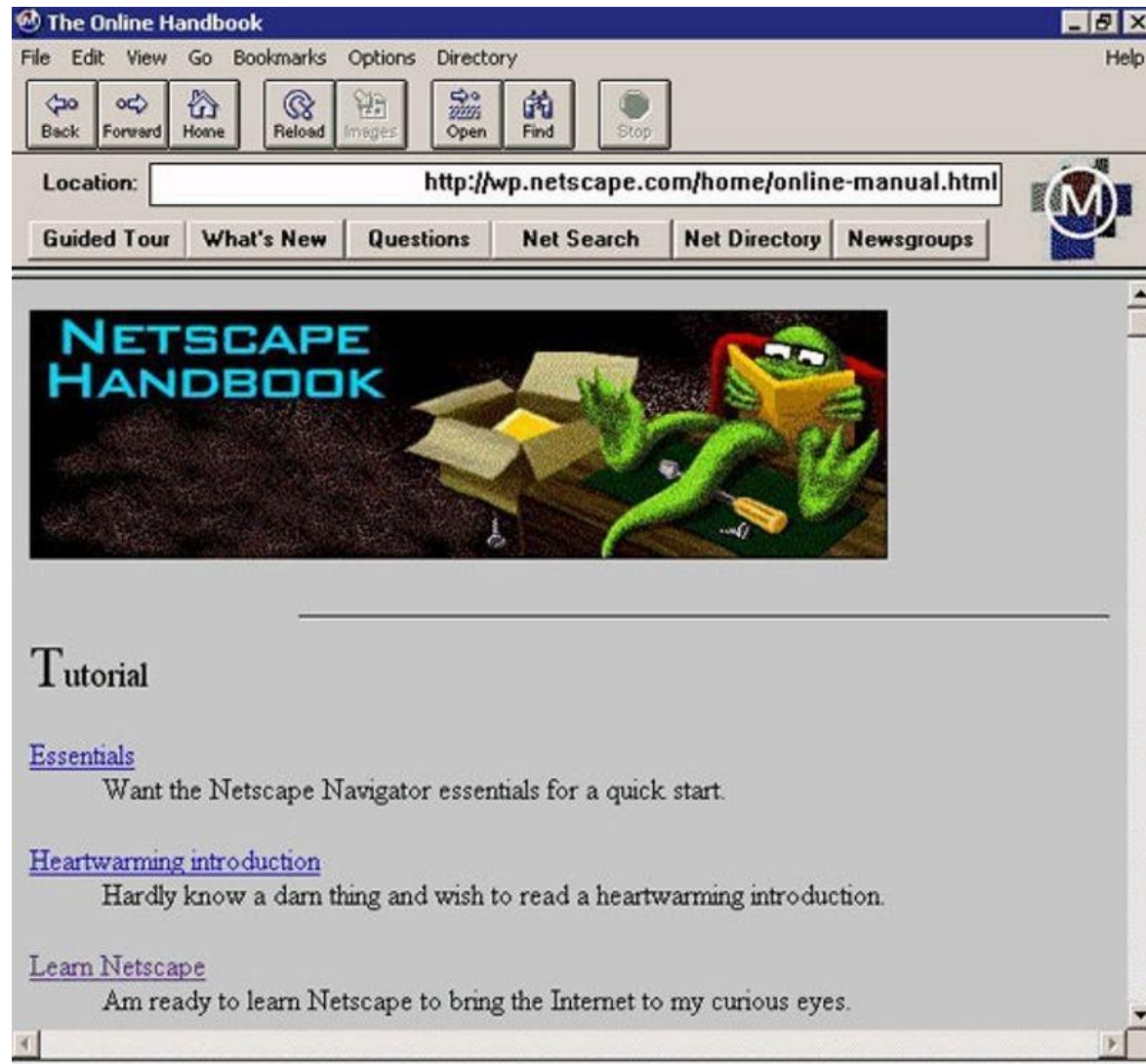
## 30 años de RFC

<https://tools.ietf.org/html/rfc2555#page-3>

## Breve historia de Internet

<https://www.internetsociety.org/es/breve-historia-de-internet/>

# Web y HTTP



# Web y HTTP

## Conceptos

- Página Web: Conetenedor de objetos
  - HTML (HyperText Markup Language)
  - Aplicación
  - Multimedia
  - El documento HTML contiene referencias a objetos
  - Cada objeto es identificable en la red por una dirección URL (Uniform Resource Locator) Ej: [www.fing.edu.uy/images/eva/LOGOS\\_UdelarProEVA.svg](http://www.fing.edu.uy/images/eva/LOGOS_UdelarProEVA.svg)

```
<!DOCTYPE html>
<html class="yui3-js-enabled" dir="ltr" xml:lang="es" lang="es">
<head>
  <title>Curso: Redes de Computadoras</title>
  <link rel="shortcut icon" href="https://eva.fing.edu.uy/theme/image.php/bcu/theme/1520625901/favicon">
  <link href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.css" rel="stylesheet">
  <link href="//fonts.googleapis.com/css?family=Droid+Sans:400,700" rel="stylesheet" type="text/css">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <meta name="keywords" content="moodle, Curso: Redes de Computadoras">
  <link rel="stylesheet" type="text/css" href="https://eva.fing.edu.uy/theme/yui_combo.php?rollup/3.17.2/yui-moodlesimple-min.css">
  <script type="text/javascript" async="" defer="" src="//www.fing.edu.uy/piwik/piwik.js"></script>
  <script id="yui_3_17_2_1_1521070265504_8" charset="utf-8" src="https://eva.fing.edu.uy/theme/yui_combo.php?m/1520625901/core/dock/dock-loader-min.js" async=""></script>
  <script id="yui_3_17_2_1_1521070265504_42" charset="utf-8" src="https://eva.fing.edu.uy/theme/yui_combo.php?m/1520625901/core/in.js&m/1520625901/filter_mathjaxloader/loader/loader-min.js">
```

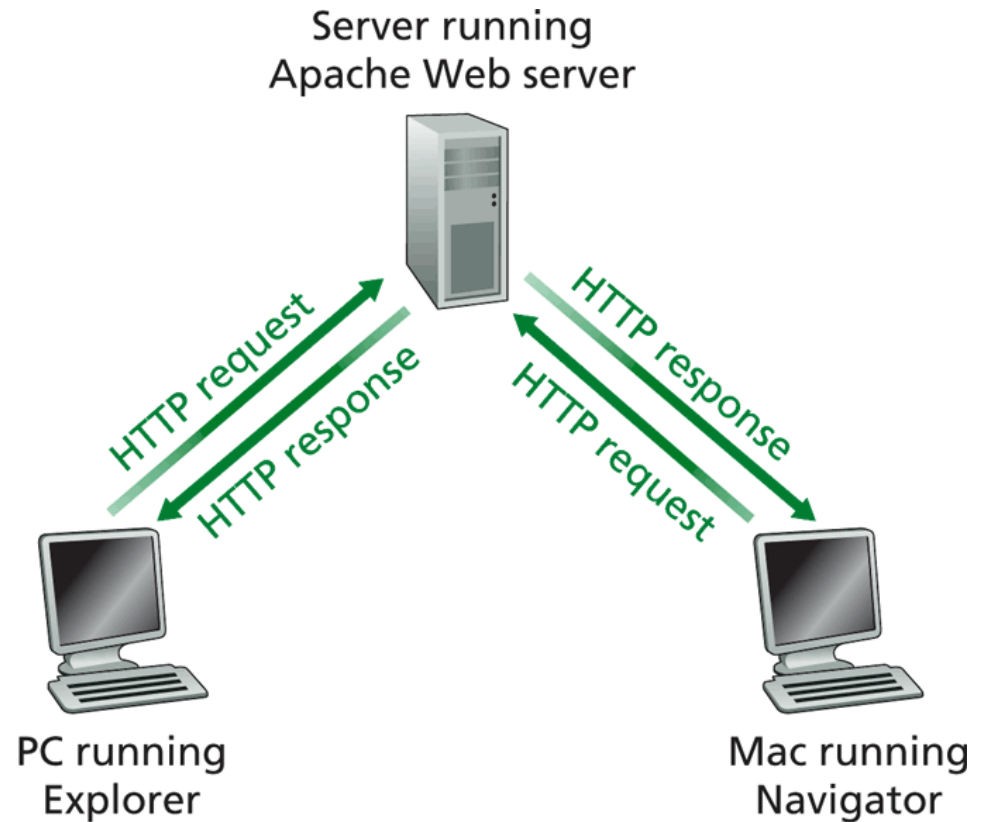
# HTTP

## HTTP (Hyper Text Transfer Protocol)

Protocolo de aplicación de la web

Modelo cliente servidor

- Cliente:  
Navegador que realiza pedidos, recibe objetos y los muestra
- Servidor:  
Servidor web envía objetos en respuesta a los pedidos.



Protocolo interoperable, variedad de navegadores en diferentes equipos/sistemas operativos, con variedad de servidores web en diferentes equipos/sistemas operativos.

# HTTP

## Utiliza TCP

- Cliente inicia una conexión TCP (crea socket) al servidor en el puerto 80
- Servidor acepta una conexión TCP del cliente
- Mensajes HTTP son intercambiados entre cliente y servidor
- Se cierra la conexión TCP

## Protocolo sin estado

- El servidor no mantiene información sobre los pedidos hechos, simplemente responde a cada pedido independientemente.



# HTTP

## No persistente

Cada objeto es enviado en una conexión TCP diferente

## Persistente

Se envían múltiples objetos en cada conexión TCP

# HTTP - Mensajes

## Dos tipos de mensajes

Request (pedido)

Response (respuesta)

## Http Request

- Mensaje ASCII
- Ejemplo

```
GET /tecnoinf HTTP 1.1
Host: www.fing.edu.uy
User-agent: Mozilla/4.0
Connection: close
Accept-language: es
-
```

## Http Response

HTTP/1.1 304 Not Modified

Connection: Keep-Alive

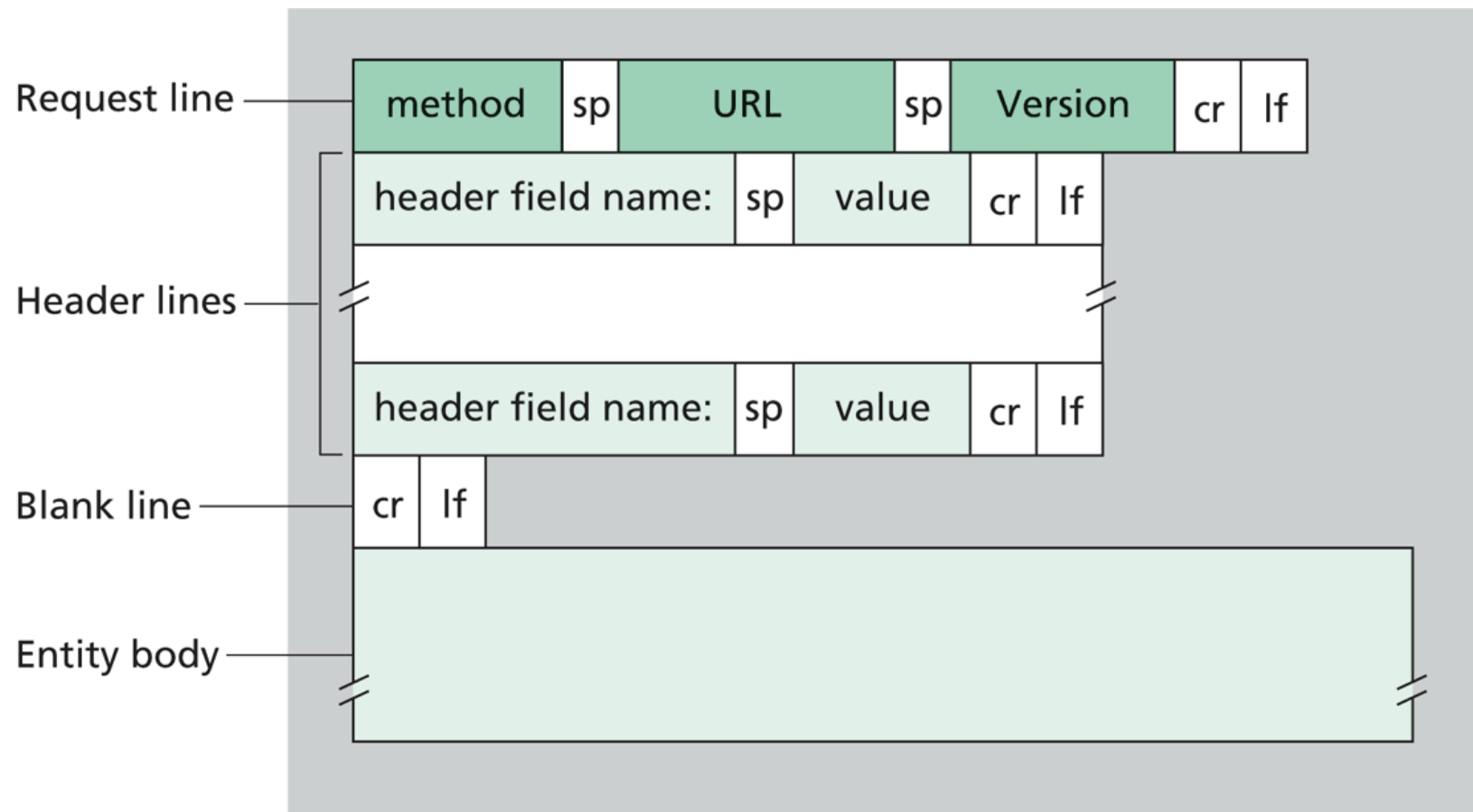
Date: Thu, 15 Mar 2018 00:15:42 GMT

ETag: "32d-4de09aea58e40"

Keep-Alive: timeout=5, max=100

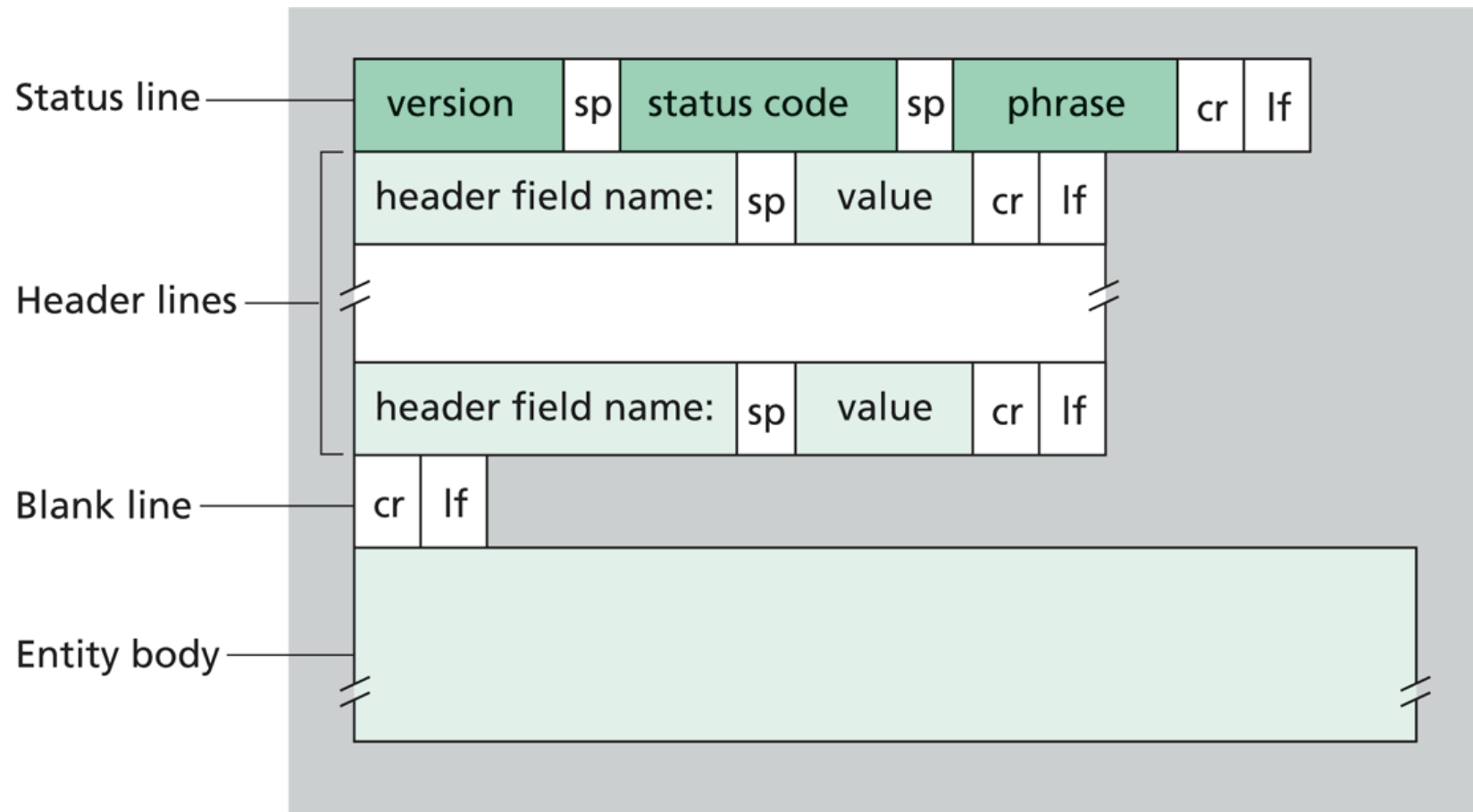
Server: Apache

# HTTP - Request



**Figure 2.8** ♦ General format of a request message

# HTTP - Response



**Figure 2.9** ♦ General format of a response message

# HTTP – Métodos

## HTTP/1.0

GET  
POST  
HEAD

## HTTP/1.1

GET  
POST  
HEAD  
PUT  
DELETE

## HTTP/2

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados HTTP verbs.

# HTTP – Entrada de formularios

## Método Post

Las páginas pueden tener formularios de ingresos de datos

Los datos son enviados al servidor en el cuerpo del mensaje HTTP

## Método GET

La información es enviada mediante una querystring en la URL.

Ejemplo:

`Www.tecnologoinformatico.tk/test?prueba=1&protocolo=html`

# HTTP – Métodos

## Método Post

Las peticiones nunca son cacheadas

Las peticiones no tienen restricciones de tamaño

## Método GET

Las peticiones pueden ser cacheadas

Almacenadas en el historial del navegador

No debe ser utilizado para manejar información sensible

Tiene restricciones de tamaño | 2,083 caracteres en IE

Sólo utilizado para recibir datos

# HTTP Response

Codigo de estado HTTP/1.1 200 OK

Cabecera Connection close  
Date: Thu, 15 Mar 2018 00:15:42 GMT  
Server: Apache  
Last-Modified: Mon, 01 Jun 2017  
18:20:12 GMT  
Content-Length: 6821  
Content-Type: text/html

Datos <!DOCTYPE>  
<html>...



# HTTP – Response códigos de estado

## Respuestas informativas

### 100 Continue

Esta respuesta provisional indica que todo hasta ahora está bien y que el cliente debe continuar con la solicitud o ignorarla si ya está terminada.

### 101 Switching Protocol

Este código se envía en respuesta a un encabezado de solicitud Upgrade por el cliente e indica que el servidor acepta el cambio de protocolo propuesto por el agente de usuario.

### 102 Processing

Este código indica que el servidor ha recibido la solicitud y aún se encuentra procesandola, por lo que no hay respuesta disponible.

# HTTP – Response códigos de estado

## Respuestas satisfactorias

### 200 OK

La solicitud ha tenido éxito. El significado de un éxito varía dependiendo del método

### 201 Created

La solicitud ha tenido éxito y se ha creado un nuevo recurso como resultado de ello. Ésta es típicamente la respuesta enviada después de una petición PUT.

### 202 Accepted

La solicitud se ha recibido, pero aún no se ha actuado.

# HTTP – Response códigos de estado

## Redirecciones

### 301 Moved Permanently

La respuesta indica que la URI de la petición ha cambiado, posiblemente se incluya en la respuesta la nueva ubicación.

## Errores de cliente

### 400 Bad Request

El servidor no puede entender la petición o contiene una sintaxis inválida

### 404 Not Found

El recurso no ha sido encontrado

## Errores de servidor

### 500 Internal Server Error

# Estado del lado del servidor y cookies

## Información enviada por un sitio web y almacenada en el navegador del usuario

### Autenticación

Se brinda una cookie con un código el cual se guarda a su vez en el servidor

### Caducidad

Al finalizar una sesión / cerrar el navegador

Fecha pre establecida

Borrado manual

Llevar control de usuarios

Seguimiento de hábitos de navegación



# Caché web

El caché web almacena documentos web para reducir el ancho de banda consumido, la carga de los servidores y el retardo en la descarga.

## Cachés privados

Funcionan para un único usuario

## Cachés compartidos

Utilizados por proveedores de servicios e instituciones para ahorrar ancho de banda.

## Cachés pasarela

Funcionan a cargo del servidor original, pueden implementar una CDN (Content Delivery Network)

# GET Condicional

Cuando un navegador va a reclamar un objeto, primero mira si está en su caché. Si está, entonces su petición es condicional. Si no está, su petición no es condicional.

Cuando se realiza una petición condicional, el servidor Web envía una nueva versión sólo si la copia local es obsoleta.

## Caché:

Especifica la fecha de la copia en la request  
**If-modified-since: <fecha>**

## Server:

Retorna sólo la cabecera en caso de no haber sido modificado  
**HTTP/1.1 304 Not modified**

## Petición

GET /tecnoinf/sanjose/index.html HTTP/  
1.1  
User-agent: Mozilla/4.0  
If-modified-since: Fri, 31 May 2013  
20:44:33 GMT

## Respuesta

Suponiendo que el objeto no ha sido modificado desde 31 May 2018  
  
HTTP/1.0 304 Not Modified  
Date: Thu, 15 Mar 2018 00:15:42 GMT  
Server: Apache/1.3.0 (Unix)

# Caché web (proxy server)

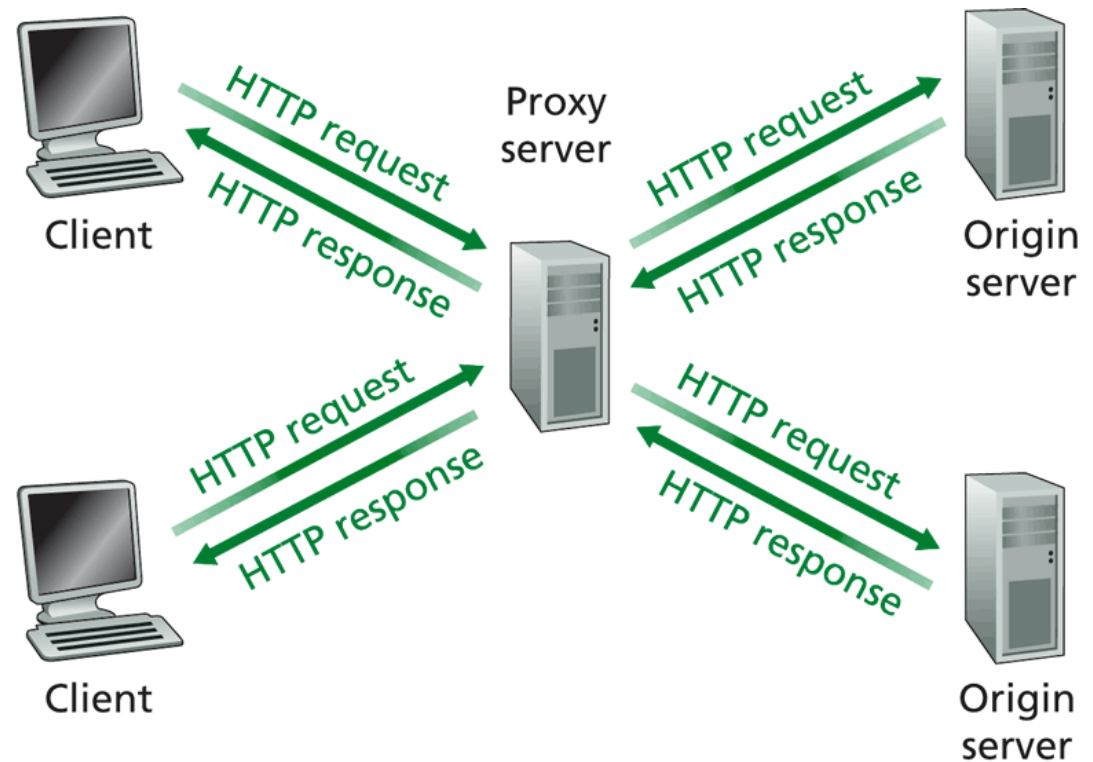
## Satisfacer el pedido del cliente sin involucrar el servidor original

Se puede configurar el navegador para que acceda mediante caché

El navegador envía todos los pedidos al caché

Si el objeto se encuentra se devuelve del caché

Si no se encuentra se obtiene del cliente original, se guarda en el server proxy y se devuelve al usuario



**Figure 2.10** ♦ Clients requesting objects through a Web cache

# Caché web

**Actua como cliente y servidor**

**Utilizado por ISP**

(Universidad, empresa, proveedor residencial)

**Utilidad**

Reducir tiempo de respuesta al cliente

Reducir el trafico en la institución



# Caché web - CDN

## Red de entrega de contenidos

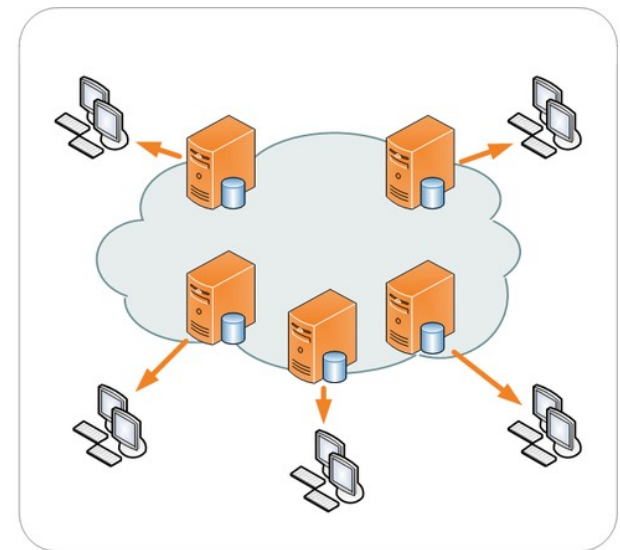
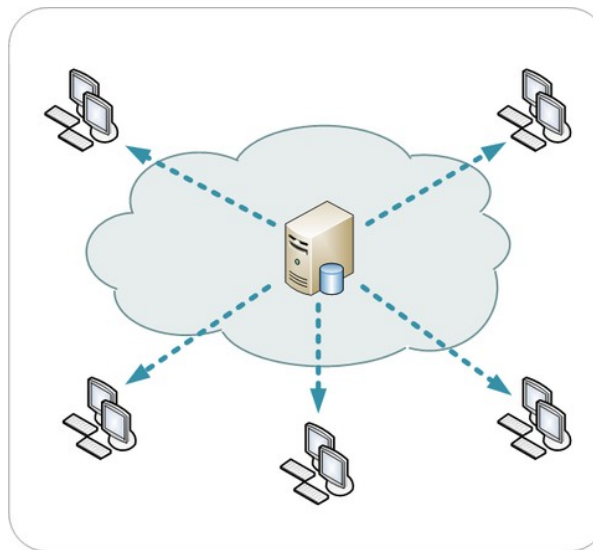
(Content delivery network)

Se contiene una copia de los datos de forma distribuida en varios puntos de la red.

Un cliente accede a los datos desde un nodo cercano

Se reduce el tiempo de respuesta y la pérdida de información

Se reduce la carga de los servidores



# HTTP/1.1 vs HTTP/2

## HTTP/1.1

Domain Sharding

Image Sprites

Concatenation / Minification

## HTTP/2

Compresión de cabeceras

Server Push

Introduce Multiplexación