LA CAPA APLICACION

Andrés Arcia

(Referencia: Kurose-Ross, Computer Networking, Fifth Edition)

La Capa Aplicación

- Prinpicios de las aplicaciones de Red
- Web y HTTP
- FTP
- Correo Electrónico
- DNS
- Aplicaciones P2P
- Programación Sockets TCP y UDP

Capa Aplicación

- Aplicación → **razón de ser** de la Internet
- 70's y 80's → aplicaciones basadas en **texto:** email, ftp, telnet, grupos news, chat.
- 90's → www (navegación, web, comercio).
- final de los 90 → Mensajería Instantánea, P2P, video compartido, VoIP, Streaming, Radio, TV.
- La **capa aplicación** sirve como **entrada** para comprender la pila protocolar.

Lista de las aplicaciones

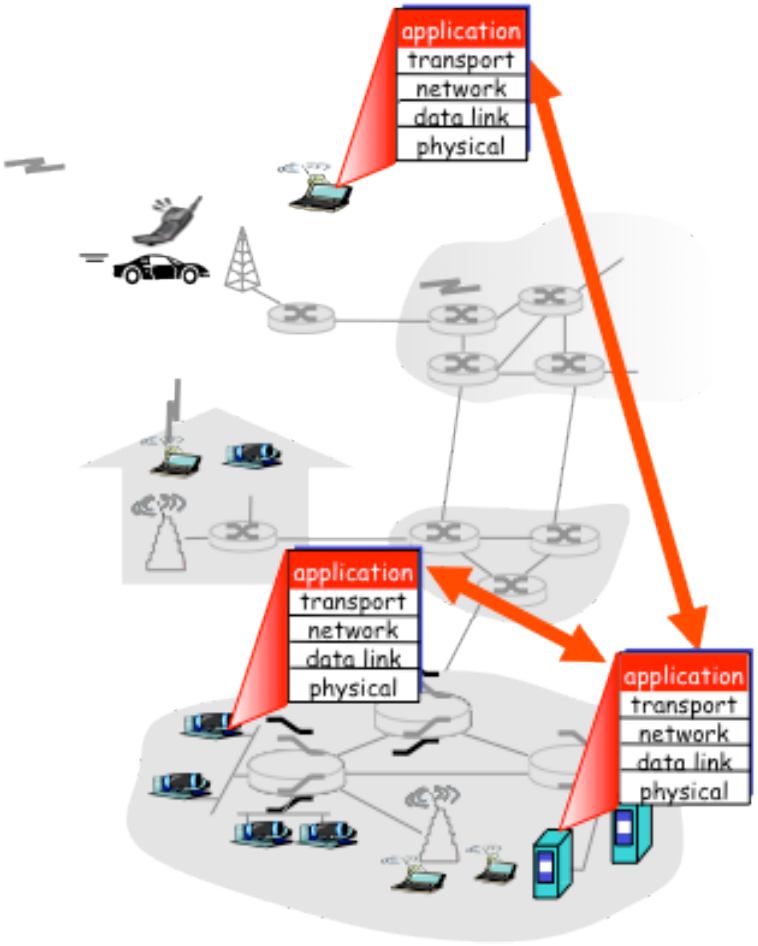
- Email
- Mensajería instantánea
- Login remoto
- Redes P2P
- Juegos multiusuario de red.
- Streaming de videos cortos.
- Redes sociales
- VoIP
- Video conferencia en tiempo real
- Computación Grid.

Principios de Aplicaciones en Red

- Ej: Servidor Web y su cliente (PC, laptops, PDA, Celulares)
- **P2P**: Cada host es un servidor parte de una comunidad.
- Desarrollo de Aplicaciones (capa Aplic.) es diferente de las aplicaciones de otras capas (ej: enlace, red).

Modelo de Red para Desarrollo de Aplicaciones

- Escribir programas que:
  - Se ejecuten en diferentes sistemas.
  - Se comuniquen a través de la red.
- En el Core (centro) de la red:
  - Los dispositivos del Core no ejecutan aplicaciones usuario.
  - Desarrollo de aplic. en los extremos permite una rápida expansión.



Arquitectura de Red

- Diseñe y planifique antes de programar distribuido.
- ¿Cuales son los servicios de red? Cliente/Servidor y P2P.
- Aplicación Web → Cliente/Servidor
  - Cliente requiere Obj., Servidor otorga Objetos.
  - Cliente no se comunica con otros clientes (IP dinamicas)
  - Cliente comunica con servidores a través de una IP (permante)
    - Ej: Web, FTP, email.
  - Servidores populares: Twitter, Facebook (utilizan cluster o granjas de servidores).
  - Los grandes servicios: google, yahoo, youtube (infraestructura intensiva).

Arquitectura Cliente/Servidor

- Servidor
  - Los hosts siempre están encendidos
  - Las direcciones IP no cambian
  - Granjas de servidores para escalar
- Cliente
  - Se comunican con el servidor
  - Intermitentes
  - Las direcciones IP cambian
  - No hay comunicación directa entre ellos

Peer to Peer

- Conecta pares de Hosts activos intermitentes.
- Comunicación **no** pasa por un **servidor central**
- Comunicación arbitraria entre un par de hosts
- Ejemplos:
  - **BitTorrent**: distribución de archivos.
  - **eMule, Limewire**: archivos compartidos.
  - **Skype**: telefonía
  - **pplive**: IPtv

Google Data Centers

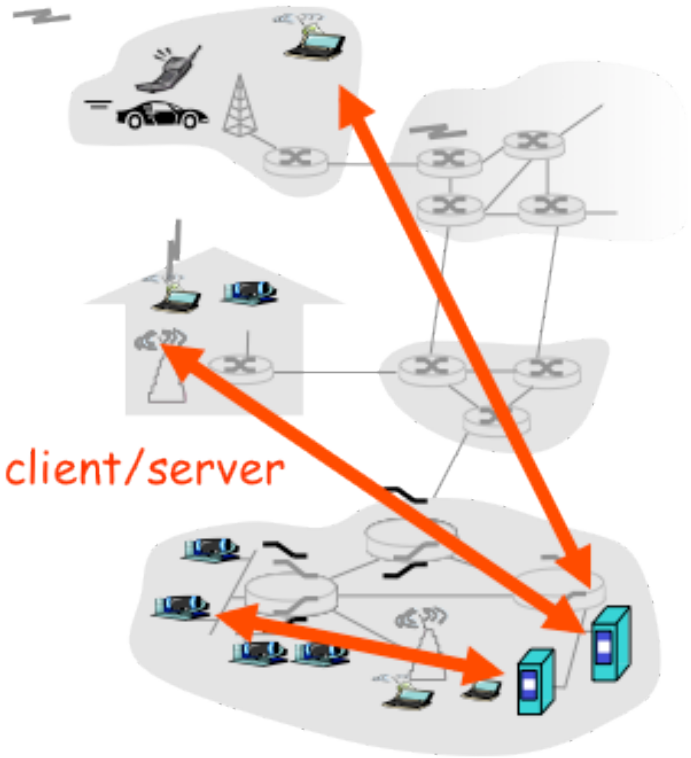
- Costo estimado de cada DC: US$ 600 Millones
  - 2006 → 1.9 billones de US$ en DC
  - 2007 → 2.4 billones de US$ en DC
- Cada DC usa entre 50-100 Megawatts de potencia.

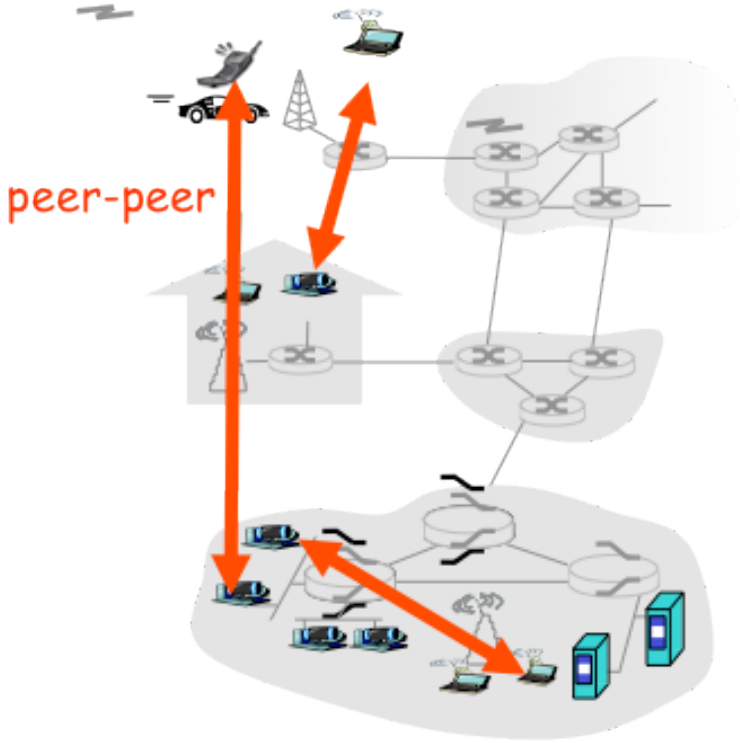
Google Data Center

Caracteristica P2P

- Sistema autoescalable: cada nuevo elemento carga y a la vez sirve al sistema.
- Hosts intermitentes y comunican de forma arbitraria.
- Efectivos en costo: poca infraestructura
- Tendencia actual de usar P2P
- Aplicaciones híbridas: P2P + C/S.
  - Sistemas de mensajeria instantanea
  - Skype

Arquitecturas





Retos de las Aplicaciones P2P

- Asimetría de los ISP → P2P estresa los ISP
  - Aplicaciones deberian estar diseñadas como "ISP friendly".
- Seguridad: la apertura y distribución del sistema lo hace vulnerable.
- Incentivos: Los usuarios **controlan** el ancho de banda, almancenamiento, recursos de computación.

Aplicaciones Hibridas P2P/CS

- Skype
  - Aplicación de VoIP en modo P2P
  - Servidor centralizado para buscar direcciones del compañero remoto
  - Conexión entre clientes es directa (no hay servidores intermedios)
- Mensajería instantánea
  - Dialogo entre 2 usuarios es P2P
  - Servicio centralizado para detectar presencia y localidad
    - La dirección IP queda registrada en un servidor central al estar on-line
    - El servidor central guarda direcciones IP de los contactos.

Comunicación entre Procesos

- Un **Proceso** es un programa ejecutandose en una máquina.
- Interés en la comunicación entre procesos:
  - Diferentes máquinas.
  - Diferentes OSs.
  - Usan la misma pila protocolar (stack).
  - Solo debe regularse el intercambio de mensajes.

Procesos Cliente/Servidor

- En la Web, cliente y servidor **intercambian** mensajes.
- Un archivo se transfiere entre dos procesos distantes.
- En P2P un proceso (nodo) **es** ambos, cliente y servidor
  - **Cliente:** Web Client, P2P downloading
  - **Servidor:** Web, P2P Uploading
- Cliente → inicia conexión, Srv. → recibe conexión.

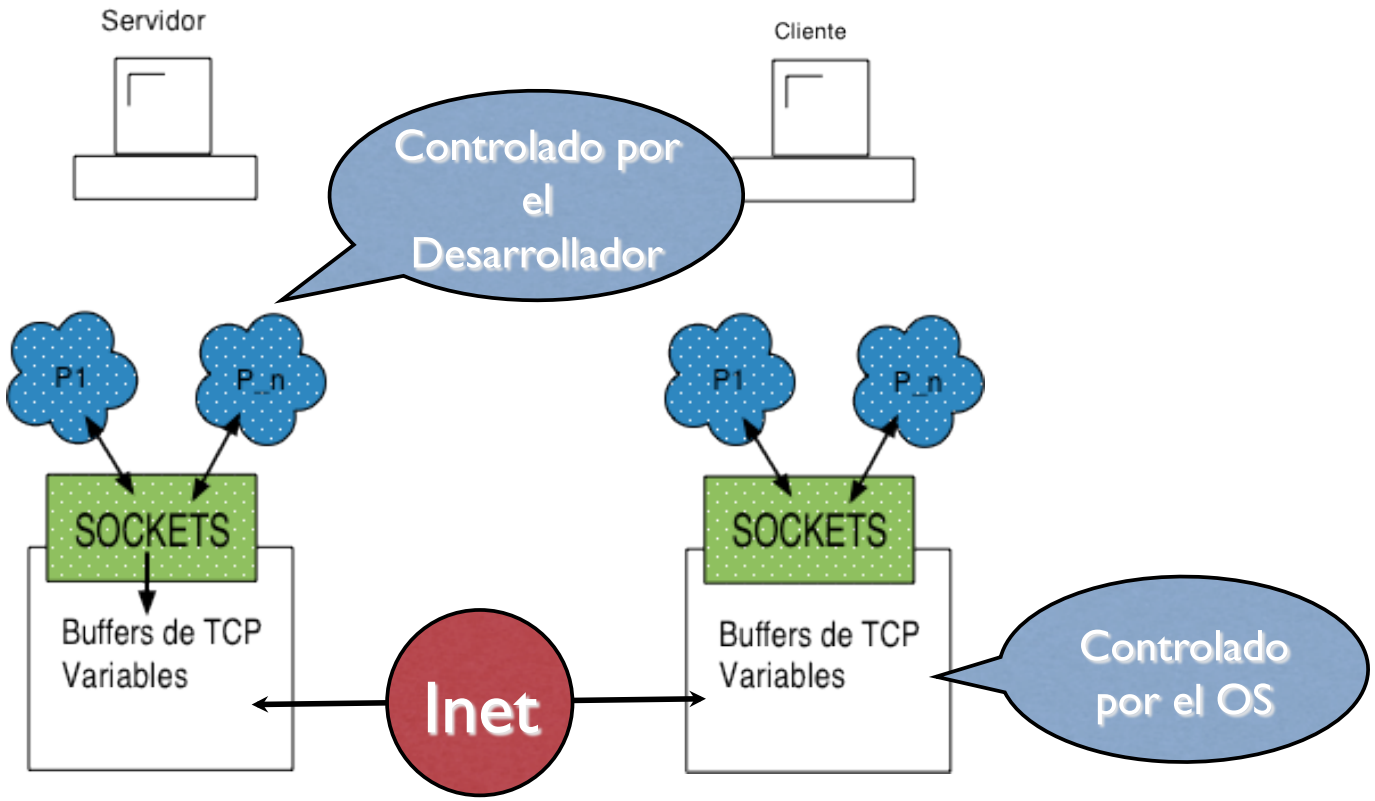
Interfaz entre Procesos y Redes

- Envio/Recepción *msgs* a través de sockets
  - Proceso → analogo a una **casa** (accesible a través de un API)
  - Socket → … es la **puerta,** también el zócalo de servicios conocidos: Luz, Cable.
- Desarrolladores tienen **control** del **API basado en sockets** pero **NO** sobre TCP/kernel (… menos en Linux, GPL software)
- API permite seleccionar protocolo de transporte y parámetros de configuración de los mismos.

Direccionamiento de Procesos

- Para recibir mensajes el proceso tiene un identificador (PID)
- El host tiene un identificador único de 32 bits.
- ¿Puede usted determinarlo en su máquina?
- ¿Bastaría entonces solamente con el ID del host?
  - No, pueden haber varios procesos en un mismo host. Hace falta un puerto.
  - Ejemplos:
    - Mail: 25
    - DNS: 53

Comunicación entre Hosts



Ejercicio Práctico

- Ver teoría de sockets.
- Hacer práctica de programación con sockets.

Definición de protocolo de capa Aplicación

- Tipos de mensajes intercambiados:
  - Solicitud y respuesta
- Sintaxis del mensaje
  - Qué campos hay y como son delimitados
- Semántica del mensaje
  - Significado de la información
- Reglas de cuando enviar y recibir msgs.
- Protocolos de dominio público
  - Definidos en RFCs
  - Permiten interoperabilidad
  - HTTP, SMTP, BitTorrent
- Protocolos propietarios:
  - Skype, ppstream

Servicios de Transporte para Aplicaciones.

- Aplicaciones **depositan** mensajes en los sockets
- Capa transporte **recolecta** y **paquetiza** estos mensajes.
- Aplicaciones **escogen** un protocolo de transporte: TCP, UDP, DCCP, etc.
- Veremos **4 dimensiones** a considerar en los servicios de transportes.

1) Transmisión Confiable

- Hay aplicaciones que son sensibles a las pérdidas (email, web) → consecuencias devastadoras. (*Discuta el por qué.*)
- Protocolo que asegure la llegada **completa** y **correcta** de los datos.
- Si las aplicaciones son tolerantes a perdidas (real-time: voz/video) → *protocolo transporte* ***sin fiabilidad.***

2) Rendimiento (throughput)

- Tasa con la que un proceso entrega bits a otro proceso.
- **Tput** varía en el tiempo, sesiones entran y salen (congestión).
- El protocolo a veces garantiza un **Tput** de **r bps**
  - Ej: telefonía a **32 Kbps**, si no se cae la conexión.
  - Aplicaciones sensibles al ancho de banda.
- Aplicaciones **elasticas** se adaptan a **casi** cualquier ancho de banda; ej: mail, web, ftp, P2P.

3) Tiempo

- ¿Cuales son las garantías de tiempo que ofrece el protocolo?
- Ej: Tiempo entre sockets. No más de 100 ms.
  - Uso en Juegos, video conferencias, supervisión en tiempo real.
- **Si no**: pausa en conversación, sensación menos realista en juegos ➔ deserción del usuario.
- En general hay restricciones en el límite superior del retardo (delay) para ser efectivas.

4) Seguridad

- Encriptación/decriptación de la data por el protocolo transporte.
- Integridad de la data.

Servicios de Transporte

- Protocolos transporte: UDP y TCP (difer. servicios)
- TCP
  - Orientado a conexión: Msg. de control en capa transporte (handshaking), luego **existe una conexión** entre 2 puntos.
  - Fiabilidad de la data: socket emisor envia **toda** la data **sin que falte un byte** en el receptor.
  - Control de Congestión: algoritmos para el bien común en Inet. Hay un compartir "justo" (**fair share**). Restricciones en la tasa de transf. puede dañar TCP.
  - Control de flujo:
  - No timing, throughput o seguridad.

Servicios UDP

- Protocolo transporte que provee servicios mínimos
- No existen conexion, No handshaking
- No hay garantías, No hay orden
- No hay control de congestion (pero se puede con DCCP).
- Otras capas pueden "frenar" el throughput
- Como los firewall "detienen" UDP multimedia → aplicaciones utilizan "TCP"

¿Para que existe entonces?

Requerimientos de las Aplicaciones

Aplicación	Perdida Datos	Rendimiento	Sensible al Tiempo
trans. archivos	sin perdidas	elastica	NO
email	sin perdidas	elastica	NO
documentos	sin perdidas	elastica	NO
audio/video	tolerante a perdidas	audio: 5Kbps-1Mbps video: 10kbps-5Mbps	SI, 100'tos msec
Juegos Interactivos	tolerante a perdidas	mismo anterior	SI, pocos msec
mensajeria instantanea	sin perdidas	elastica	SI y NO

Servicios **no** provistos

- ¿Cuáles de las 4 dimensiones son provistas por UDP o TCP?
  - No hay “timing” ni garantías de ancho de banda (es decir, de rendimiento)
  - Sin embargo las aplicaciones sensibles al tiempo se pueden ejecutar en la Internet.

Aplic. Populares Inet

Aplicación	Protocolo de Capa Aplicación	Protocolo de transporte usado
Email	SMTP [RFC 2821]	TCP
Acceso a terminal remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferencia de Archivos	FTP [RFC 959]	TCP
Streaming Multimedia	HTTP [Youtube], RTP [RFC 1889]	TCP o UDP
Telefonía Internet	SIP, RTP, propietarios (ej: Skype)	Típicamente UDP

Direccionamiento de Procesos

- ¿Como se identifica a un Host?
  - Se necesita un nombre/dirección del **host** y otro para el **proceso**.
- Host
  - Dirección IP → numero de 32 bits
  - NATs → no basta solo con 1 IP.
- Proceso
  - Número del puerto. Bien conocidos (≤ 1024 www.iana.org) ej: 80 HTTP, 25 mail, 465 mail seguro.

Protocolos de la Capa App.

- Mensajes entre procesos
  - ¿Como se estructuran? ¿Qué significan los campos? ¿Cuándo se envían mensajes? ➔ *dominio de las aplicaciones*
- Aplicación define:
  - Tipos de mensaje de requerimiento & respuesta
  - Sintaxis de los mensajes
  - Semantica de los mensajes
  - Cuando y cómo se envian y responden los msg.

Protocolo HTTP

Protocolos de la Capa App.

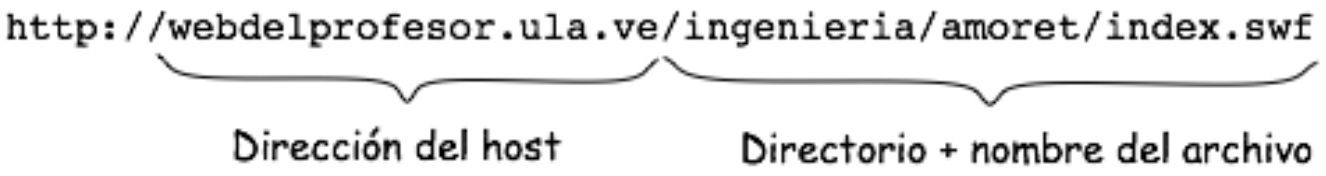
- Protocolos de Aplicación se especifican en RFC.
  - HTTP → RFC 2616. Si Ud. desarrolla un navegador debe seguir estas pautas.
  - SMTP → RFC 5321. Si desarrolla un cliente mail debe seguir estas otras.

HTTP para la Web

- A principios de los 90 Inet fue utilizada por investigadores, academicos y estudiantes. (ej: telnet, ftp, mail).
- WWW → Tim Berners-Lee 1994 (uso general)
  - Popular porque opera a la demanda, veo **lo que** quiera **cuando** quiera.
  - Se puede publicar a bajísimo costo.
  - ¿Qué más ofrece la Web?
    - Google dice: “everything on top of web!”.

HTTP en general

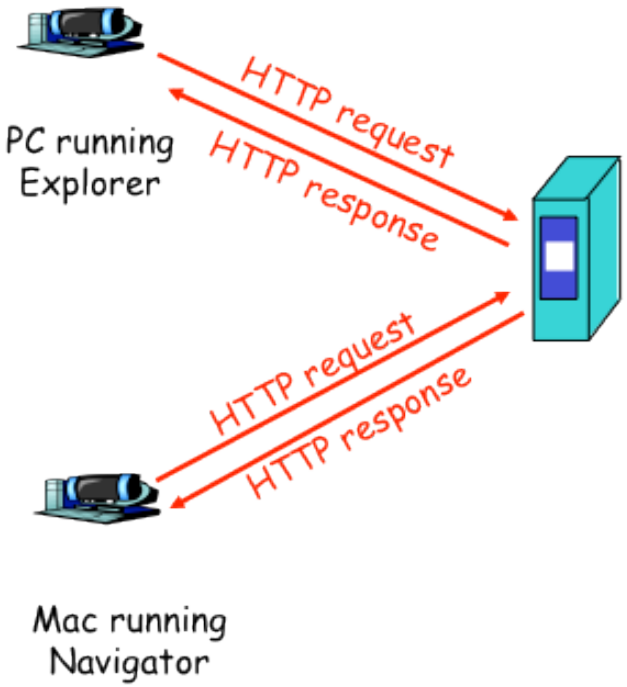
- Hyper-Text Transfer Protocol (HTTP)
  - Definido en el RFC 1945 & RFC 2616
  - Implementado en el cliente y servidor
  - Intercambio de mensajes HTTP
- Terminología
  - Página Web → grupo de objetos (jpeg, archivo html, applet)
  - cada objeto un **único** URL
  - Servidores Web: Apache / Internet Information Server.




```
http_URL = "http:" "//" host [ ":" port ] [ abs_path [ "?" query ]]
```

Idea General de Funcionamiento

- HTTP usa TCP (no UDP)
- Navegador y Servidor usan Sockets (se enlazan en puerto 80)
- Una vez enviado el msg al socket, msg manejado TCP
- HTTP no lidia con perdidas (enfoque por capas)
- No hay estado del cliente (envios repetitivos)
  - HTTP → stateless



Conexión **no** Persistente vs. Persistente

- Cuando un cliente hace varias consultas a un servidor ¿cada una va en conexiones separadas o en una sola?
  - **Por UNA** → conexión persistente (por omisión)
  - **En VARIAS** → conexión NO persistente (activable)

HTTP con no-persistente

- Supongamos 1 archivo HTML y 10 img. jpeg
  1. cliente HTTP inicia conexión TCP en srv (puerto 80) ➔ sockets || servidor acepta la conexión…
  2. cliente HTTP envia solicitud a traves de socket /camino/ archivo.html
  3. HTTP srv recibe el msg y lo procesa, obtiene el archivo.html y lo devuelve por el socket
  4. HTTP srv dice cerrar la conexión → espera que TCP haga el trabajo.
  5. Cliente HTTP recibe la respuesta. Se cierra la conexión. Extrae el archivo del mensaje y se da cuenta que hay 10 mas por recoger.
  6. Repetir 1-4

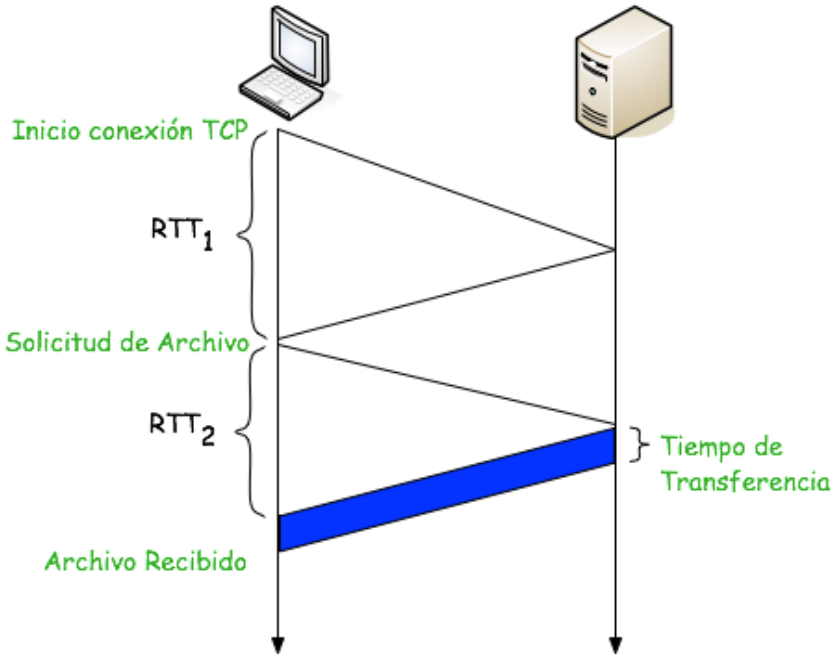
HTTP

- Navegadores modernos permiten controlar el grado de paralelismo (5-10 conexiones paralelas) → recorta el tiempo de transferencia.
- Hoy en día los navegadores usan conexiones en paralelo, en promedio 5 conexiones, y el máximo va hasta las 40 conexiones.
  - Hay quienes argumentan que el problema es de la capa transporte, quien restringe la velocidad de navegación. (ver grupo tcpm o tmrg, IETF).

Conexiones NO persistentes

Veamos el: applet para calculo de retardos de transferencia.

RTT: Tiempo de ida y vuelta de un paquete desde SRV hasta CLT



T. Respuesta:

- Inicio de conexión
- Solicitud de Archivo
- Tiempo de transferencia

Total = 2 x RTT + $T_{transfer}$

Comparación

- No-Persistente:
  - Cada objeto tiene una conexión
  - Buffers TCP son reservados por cada conexión
  - Tiempo de entrega (delay) de 2 RTT minimo.
- Persistente:
  - Envio completo en una sola conexión
  - Solicitud/Recepción de información sin necesidad de espera por la apertura de la conexión.

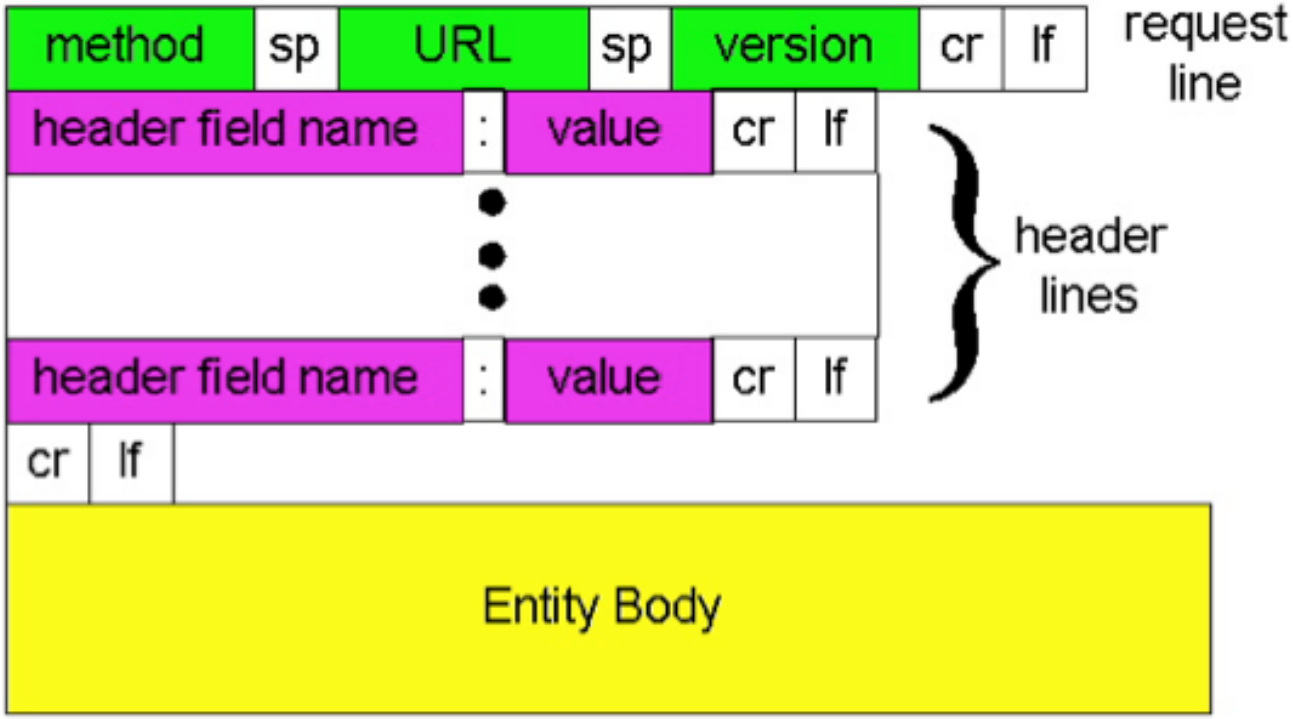
Formato del Msg

- Request:
  - GET /directorio/pagina.html HTTP /1.1
  - Host: www.ula.ve
  - Connection: close
  - User-agent: Mozilla/4.0
  - Accept-Language: fr

Formato del Msg

- Escrito en lenguage comprensible por el humano
- Lineas separadas por enter y la ultima 2 enters
- Linea indicando **Request** luego encabezado
- Métodos: GET, POST, HEAD, PUT & DELETE
- GET es el más utilizado

Formato del Msg

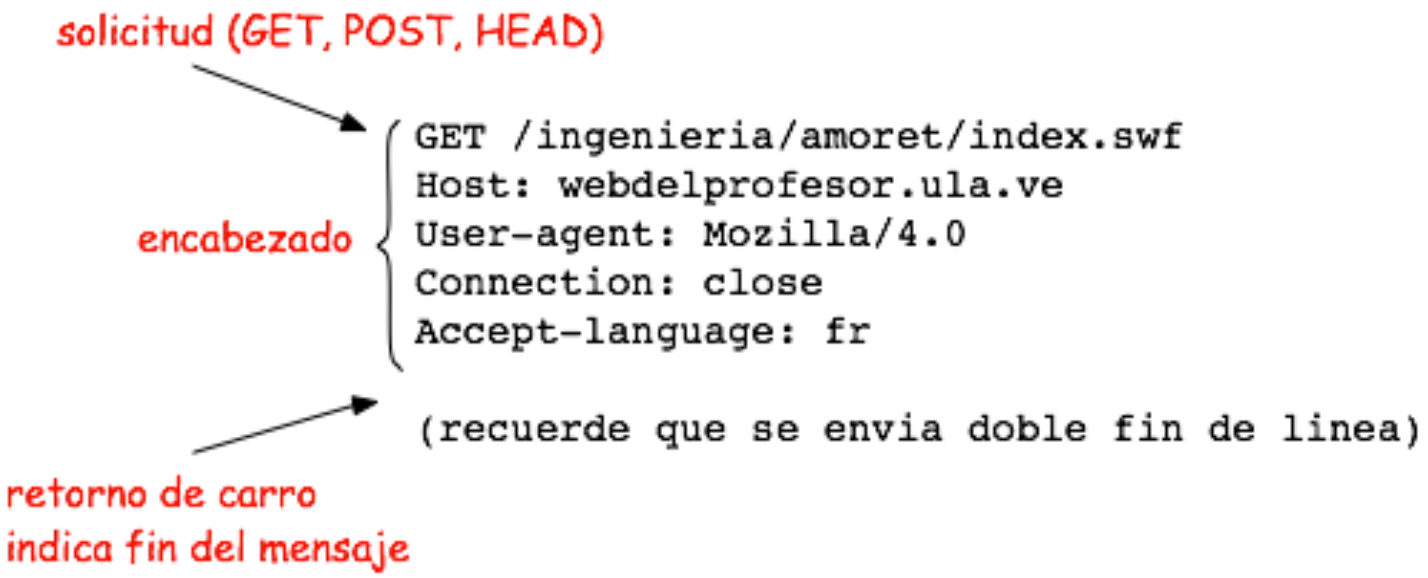


Intercambio de Información

- Método Post:
  - La página Web incluye una planilla de entrada (input form)
  - La información va dentro del cuerpo del mensaje
- Método URL:
  - Utiliza el método GET
  - La entrada se carga en la dirección URL: http://googlefight.com/index.php?lang=en_GB&word1=bill+gates&word2=linus+torvalds

Ejemplo método GET

- El mensaje está en formato ASCII legible por el humano.



Definición de Mensajes

- **Host**: de donde se sacará el objeto (innecesaria pero usada por los caches)
- **Connection**: indica si se quiere cerrar la conexión despues de una transferencia.
- **User-agent**: Permite respuestas personalidas de acuerdo al navegador.
- **Accept-Language**: Preferencia del idioma.

Mensaje Respuesta

- HTTP/1.1 200 OK
- Connection: close
- Date: Ultima fecha de extracción
- Server: Quien generó el mensaje
- Content Length: longitud msg
- Content-Type: Tipo de archivo
- Entity Body: El objeto en si...

Subiendo data desde el cliente

- Método Post
  - La página web incluye una forma de entrada
  - La entrada es cargada al servidor en el cuerpo del mensaje
- Método URL:
  - Utiliza el método GET
  - La entrada se carga en campos del URL de la línea de la solicitud.

```
http://www.google.com/#hl=en&q=merida&fp=7db4f7af4a13aa89
```

Tipos de métodos

- HTTP 1.0
- GET
- POST
- HEAD
  - Para pedir los encabezados del objeto (sin su contenido).
- HTTP 1.1
  - GET, POST, HEAD
  - PUT
    - Cargar un archivo (en el cuerpo-entidad) en el camino (path).
  - DELETE
    - Borra archivos especificados en el URL

Codigos de Estatus

- **200**
  - Solicitud exitosa, el objeto viene en el mensaje
- **301**
  - Movido permanentemente a otro sitio. Nueva posición en Location:
- **400**
  - Request erroneo
- **404**
  - Objeto no encontrado
- **505** Versión del HTTP no soportada

Ejercicio


```
[~]$ telnet webdelprofesor.ula.ve 80
Trying 150.185.130.26...
Connected to ws-01.ula.ve.
Escape character is '^]'.
GET /ingenieria/amoret/ HTTP/1.1
Host: webdelprofesor.ula.ve

HTTP/1.1 200 OK
Date: Sun, 29 Aug 2010 22:23:20 GMT
Server: Apache
Last-Modified: Sun, 02 May 2010 20:44:40 GMT
ETag: "af0096-142-485a28b977e00"
Accept-Ranges: bytes
Content-Length: 322
Content-Type: text/html

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE html PUBLIC "-//
W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd"><html xmlns="http://www.w3.org/1999/
xhtml"><head><title></title><meta http-equiv="refresh" content="0;url=
andreswebpage/Inicio.html" /></head><body></body></html>Connection
closed by foreign host.
[~]$
```

Ejercicio

- Hacer un telnet a webdelprofesor.ula.ve en el puerto 80.
- Escribir el comando:
  - `GET /ingenieria/amoret/index.html`
- Utilice Wireshark para ver los paquetes HTTP. ¿Cuantos paquetes ocupa index.html?

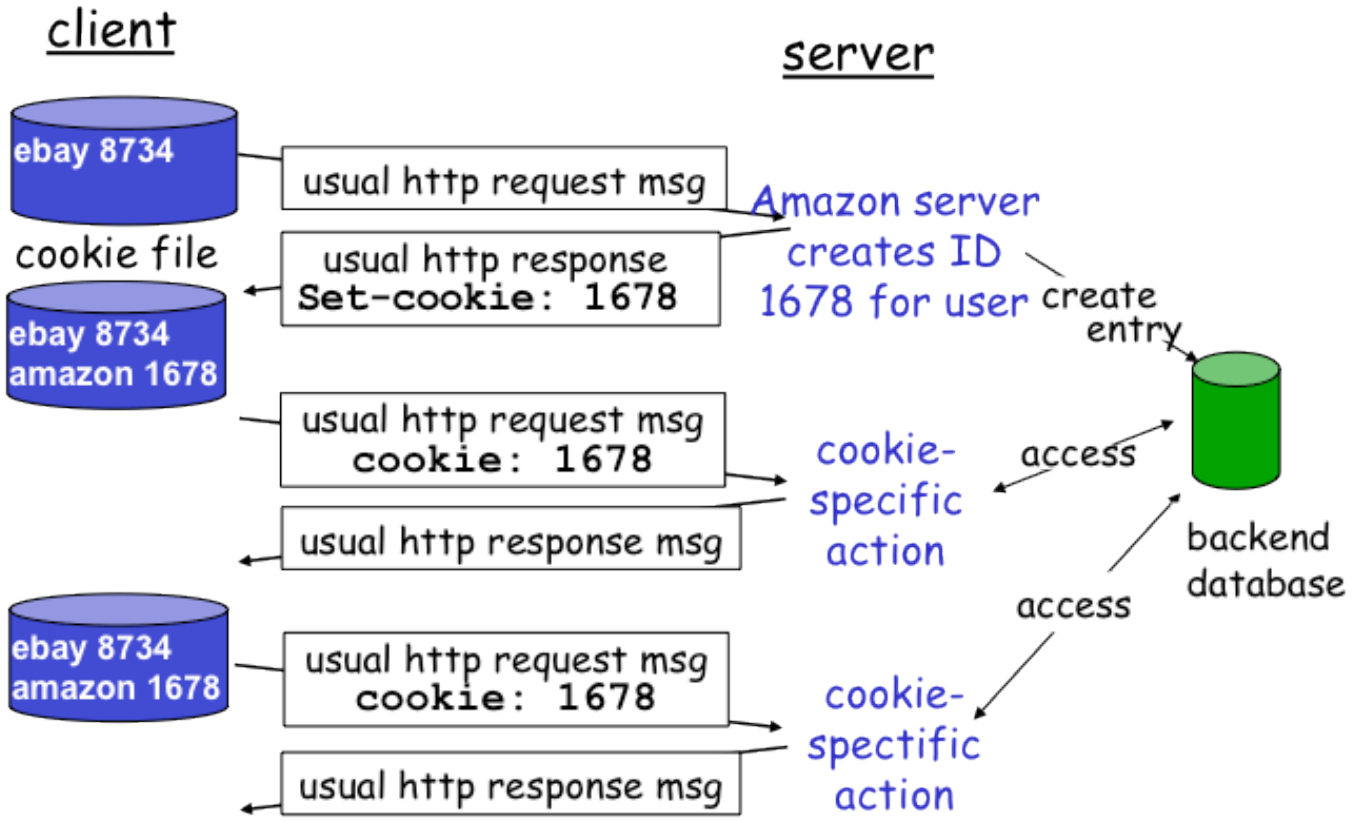
Cookies

- Servidor HTTP no guarda estados → permite a los servidores Web atender miles de conexiones.
- Un **cookie** (RFC 2965) identifica a un usuario → método altamente popular.
- Sirve para llevar cuenta de la navegación en un sitio. Ej: Agrupar objetos a comprar para pagar al final.
- ¿Invade nuestra privacidad? (controversial).

Cookies

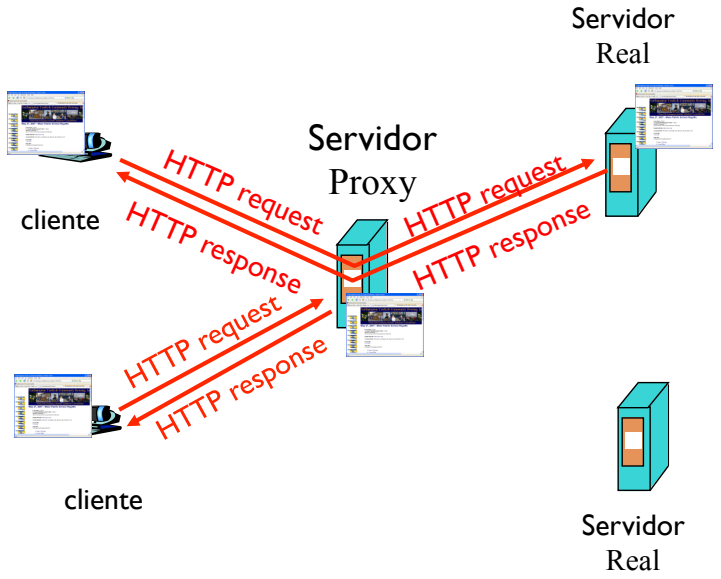
- Cuando se visita un sitio:
  - Se crea un identificador único (UID)
  - Una entrada en la BD con el UID
- Componentes de un cookie:
  - Linea en el encabezado para **la solicitud** y **la respuesta**
  - Archivo cookie en el computador del cliente (manejado por el naveg.)
  - Base de datos en el sitio Web

Cookies: guardar estado

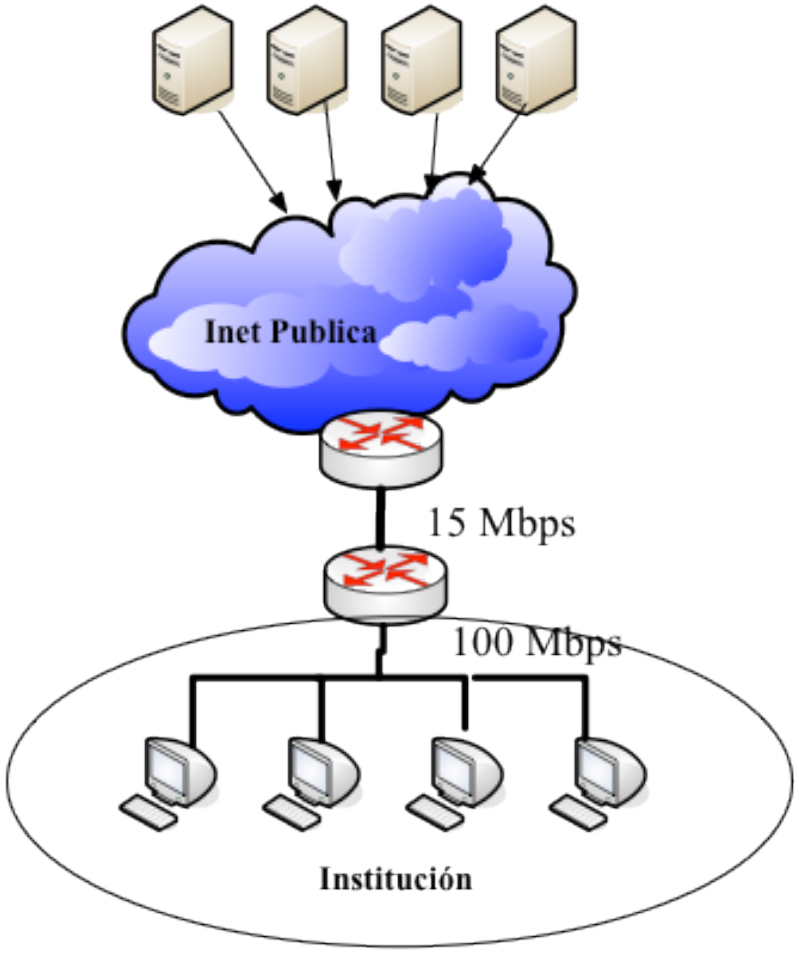


Uso del Cache

- Usuario configura el acceso Web via proxy
- Navegador envía la solicitud al proxy (cache)
  - El proxy descarga la información en el cache
  - Cuando está lista la devuelve al cliente

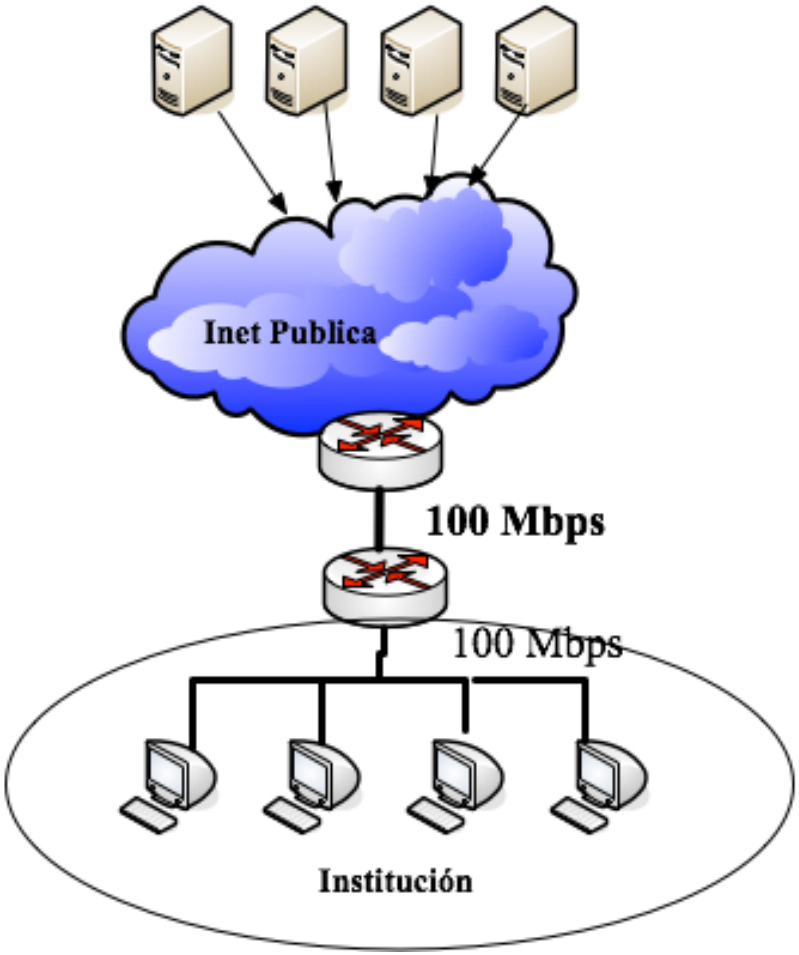


Beneficios del Caching



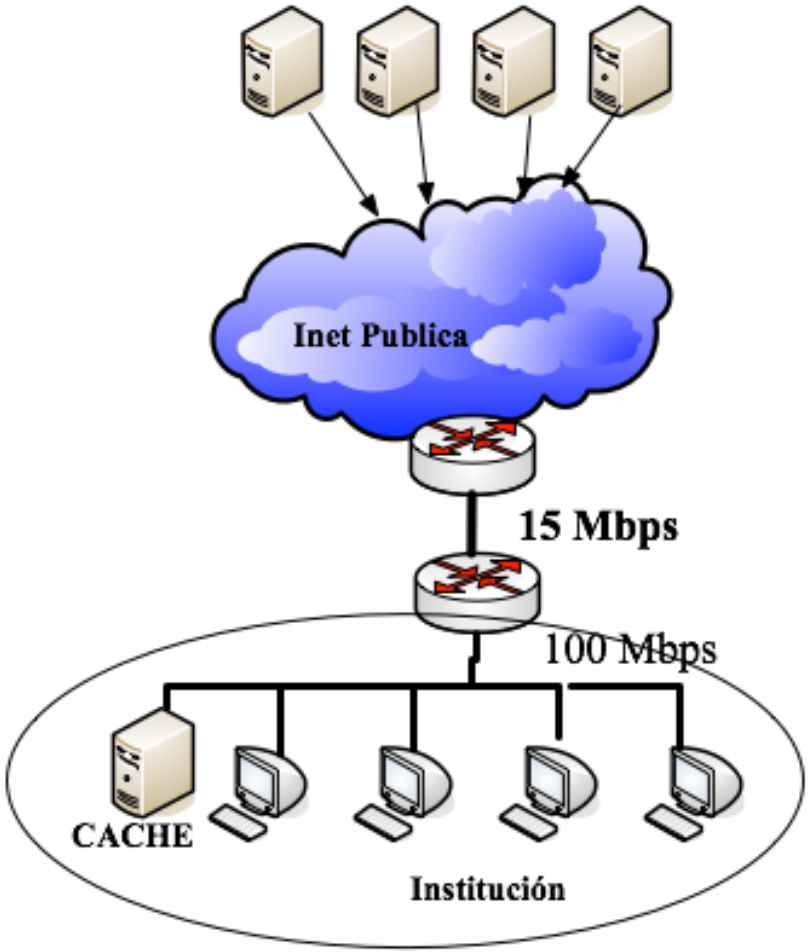
- Tamaño prom. Obj = 1 Mbit.
- Frecuencia 15 req/seg.
- Mensajes HTTP despreciables.
- Tiempo entre una solicitud y respuesta (todos los paquetes IP) ~ 2 s. (**Inet delay**).
- Intensidad de tráfico:
  - Instituto = 15 req/s * 1Mbps /100 Mbps = 0.15
  - Inet Publica = ... / 15 Mbps = 1
  - Delay total = Inet delay + delay acceso + delay LAN = 2 seg + minutos + miliseg.

Solución 1



- Solución posible (1):
  - Incrementar el ancho de banda de acceso a 100 Mbps
- Utilización LAN: 15%
- Utilización acceso: 15%
- Delay total = Inet delay + access delay + LAN delay
- = 2 seg + mseg + mseg
- Actualización costosa

Solución 2: caching



- Caracteristica cache:
  - Tasa de acierto: 0.4 (entre 0.2 y 0.7)
- Consecuencia
  - 40% de request estan localmente
  - 60% en el servidor origen
  - Enlace de acceso se usa al 60% → espera de miliseg (~10ms)
  - Delay total = Inet delay + delay acceso + LAN delay = 0.6*2 seg + mseg + mseg < 1.3 seg.

Conditional GET

- Copia en el cache puede estar "podrida"
- Verificar si los objetos están up-to-date → conditional GET o cGET.
  - Una solicitud utiliza un cGET.
  - Incluye un comando de encabezado:
    - **IF-MODIFIED-SINCE**

Características del Web-Cache

- El cache es cliente y servidor.
- Pertenece al ISP.
- Reduce el tiempo de respuesta para los clientes.
- Reduce el tráfico en el enlace de acceso de instituciones.
  - Menos actualizaciones costosas en el ancho de banda.

Web Caching

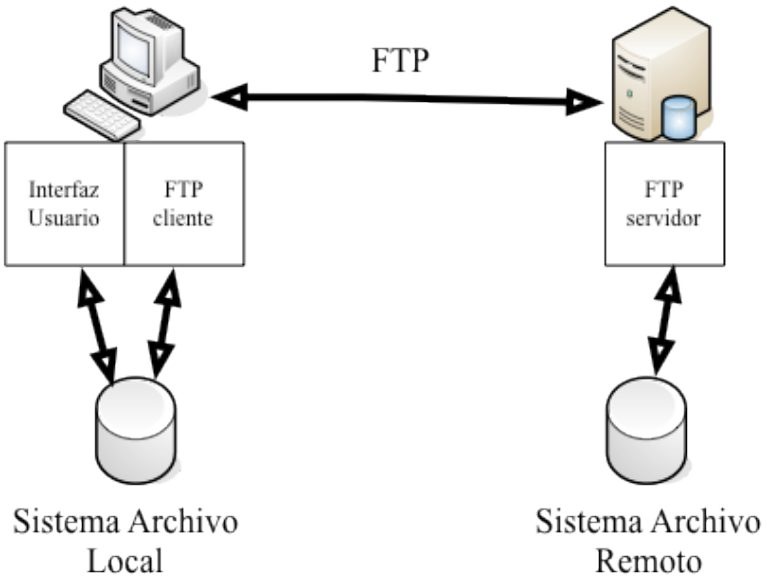
- Se rige también por el principio del proxy server
- Request son dirigidas al Web Cache (acceso vía cache).
  - ¿Qué sucede cuando se solicita un objeto?

```
1. Establecer Conexión TCP y enviar solicitud HTTP
2. ¿Hay copia local en el Web Cache?
2.1 SI: Enviar objeto en el Mensaje HTTP de regreso
2.2 NO: Abrir conexión TCP con srv desde el cache y pedir objeto
3. Guardar copia local (en Cache) y enviar al cliente en msh HTTP.
```

Ejercicio Práctico

Hacer práctica de HTTP.

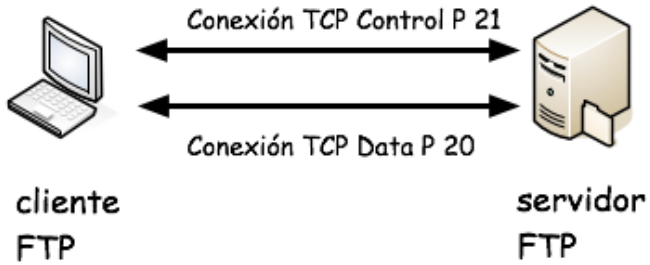
Transferencia de Archivos



- File Transfer Protocol (**FTP**): transferencia desde y hacia un host remoto. RFC 959
- Provee autenticación: login/ password.
- HTTP & FTP tienen en comun TCP.
- Diferencias:
  - FTP usa 2 conexiones TCP: control (puerto 21) y datos (puerto 22).
  - control: id, password, cd, put, get (fuera de banda)
  - HTTP: “en una sola banda” (ctrl+data en la misma conexión).

Características

- FTP abre una conexión TCP para cada objeto a transferir, luego cierra conexión.
- La conexión control queda abierta para todos los comandos.
  - Esto limita el número de usuarios.
- Hay una conexión de datos para las transferencias
- Recuerde que HTTP es *sin estado*.



Comandos/Respuesta FTP

- Los comandos (cli-srv) y respuestas (srv-cli) son en formato arcaico ASCII 7-bits.
- comandos FTP como HTTP son legibles/humano
- FTP está especificado el en RFC 959.
- FTP mantiene el estado: directorio actual, autenticación previa.

Comandos/Respuesta FTP

Comandos

- USER nombreusuario: envío al servidor
- PASS password: envío al servidor
- LIST: lista los archivos en el pwd del servidor
- RETR fichero: obtener un fichero del servidor
- STOR: guardar en el servidor

Respuestas

- 331 Username OK.
- 125 Data connection already open; transfer starting.
- 425 Can't open data connection.
- 505 Error writing file

Ejercicio Práctico

Instale un servidor FTP en su máquina.

Lance el Wireshark

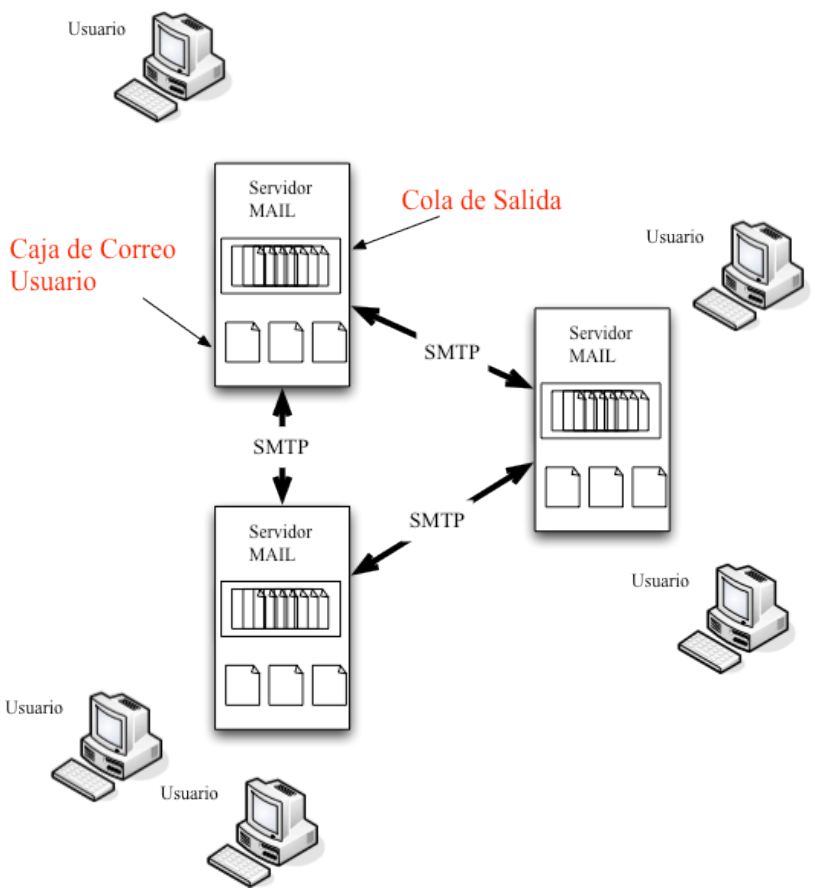
Abra una sesión FTP y observe la traza.

¿Nota algunas de las caracteristicas anteriormente expuestas?

El Correo Electronico

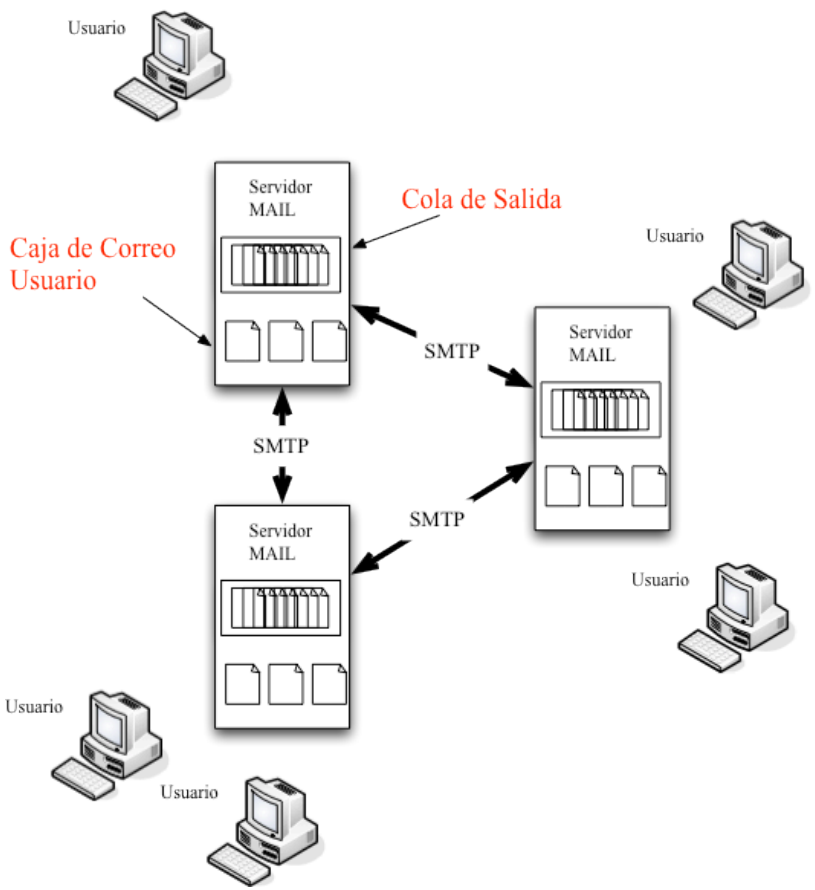
- Aplicación más popular de la Internet
- Comunicación asíncrona con el medio (no hay coordinación para envío/recepción)
- Acotación: El correo electrónico también es la capacidad de escribir información en un espacio de disco ajeno.

Arquitectura Mail



- Lectores de correo para: *enviar, responder, renviar*.
- Componentes: user agent, mail server y protocolo SMTP.
- Servidor Mail **almacena mensajes** en cola de salida para hacer intentos repetidos c/30 min si hay problemas.
- Receptor **se comunica** con el servidor mail para leer mensajes.
- **Populares**: Apple Mail, ThunderBird, Microsoft Outlook.
- Utiliza TCP.
- SMTP funciona como **cliente y servidor**.

Servidor Mail



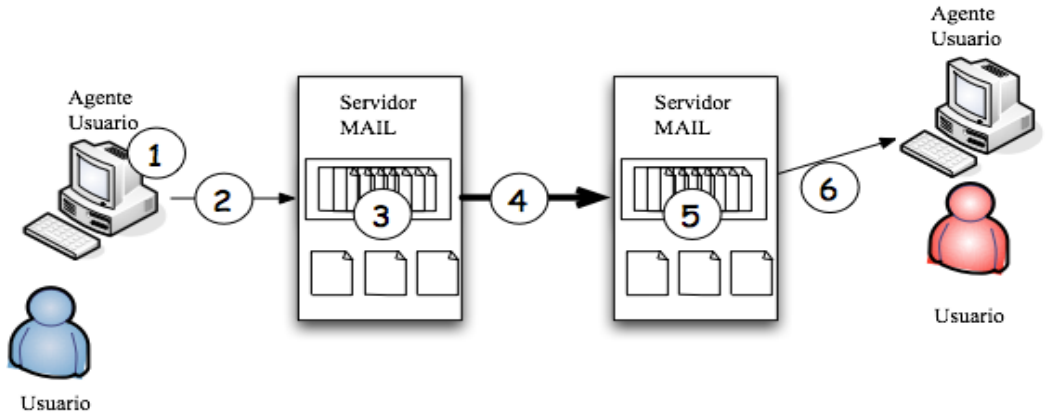
- El mailbox contiene los mensajes de los usuarios
- La cola de mensajes de los mensajes salientes (aun en espera)
- Protocolo SMTP entre servidores mail para envío de mensaje.

Protocolo SMTP

- Simple Mail Transfer Protocol (SMTP) (RFC 5321)
- Mas antigüo que HTTP (< 1992)
  - Recuerden que la Internet nació y ya había una implementación de un sistema email (70's)
- Caracteristicas arcaicas: usa texto ASCII a 7-bits
  - Tenia sentido en los 80's cuando no habían: jpg, mov, wav.
  - Hoy es dolor de cabeza, hay que codificar para enviar nuevos formatos.

¡HTTP no codifica!

Pasos para enviar un mensaje.



1. Usuario (A) hace un msg., provee dirección destino y pasa msg al agente.
2. Agente envia correo al srv. de correo saliente y es puesto en cola.
3. Srv. correo saliente abre conexión TCP al srv. SMTP destino.
4. Luego del inicio (handshake) SMTP, se entrega el correo.
5. Servidor de correo entrega el mensaje en el mailbox (B).
6. Usuario (B) puede leer su correo a través de su cliente mail.

Observaciones

- El correo saliente siempre se encuentra en 2 ptos: origen o destino (nunca en medio).
- Se abre conexión al puerto 25 (srv. SMTP)
- Transferencia confiable (conexión TCP)
- SMTP usa conexiones persistentes.
- Usa ASCII de 7 bits para codificar los mensajes.
- Utiliza “LFCR . LFCR” para determinar el fin del mensaje.

Ejemplo 1


```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Ejemplo 2: servidor mail ULA

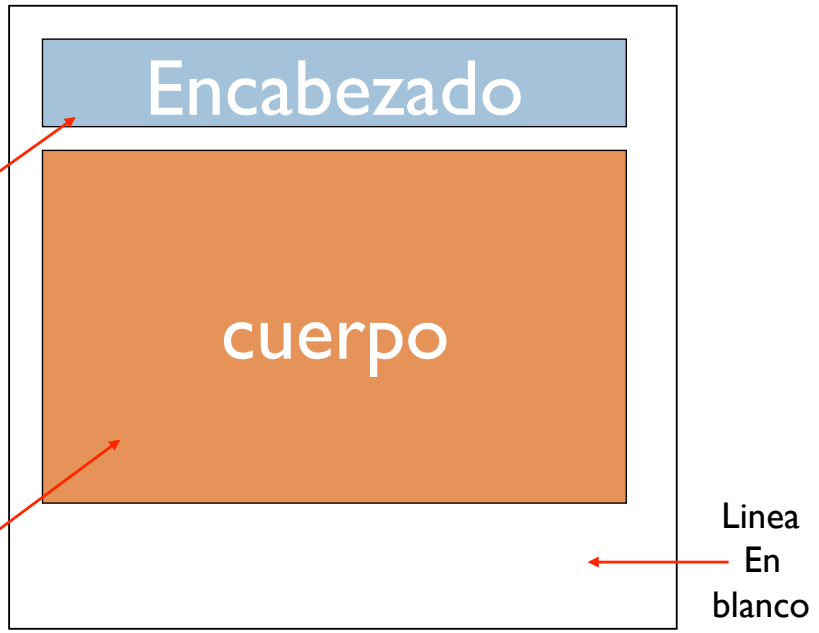
- **S:** 220 belial.ula.ve
- **C:** HELO 201.249.86-62.dyn.dsl.cantv.net
- **S:** 250 belial.ula.ve Hello 201.249.86-62.dyn.dsl.cantv.net [201.249.86.62] (may be forged), pleased to meet you
- **C:** MAIL FROM: <andres@arcia.net.ve>
- **S:** 250 2.1.0 <andres@arcia.net.ve>... Sender ok
- **C:** RCPT TO: <amoret@ula.ve>
- **S:** 550 5.7.1 <amoret@ula.ve>... Mail from 201.249.86.62 refused - see http://www.spamhaus.org/
- **C:** QUIT
- **S:** 221 2.0.0 belial.ula.ve closing connection
- **C:** Connection closed by foreign host.

Hasta

Clase
Hacen
protoc
telnet.

Composición de un mensaje

- SMTP: protocolo para el intercambio de mensajes.
- RFC 822: estandar para el intercambio de mensajes.
- Encabezado:
  - To:
  - From:
  - Subject:
- Cuerpo: el msg. En ASCII



Diferencias entre HTTP & SMTP

- **HTTP**:
  - Protocolo del tipo PULL (conexión iniciada por maquina que **obtiene** info).
  - Información no codificada.
  - Cada objeto está en un mensaje HTTP.
- **SMTP** (headers en el RFC 5322):
  - Protocolo del tipo PUSH (conex. iniciada por maquina que **carga** la información)
  - Información codificada.
  - Todos los objetos en un solo mensaje.

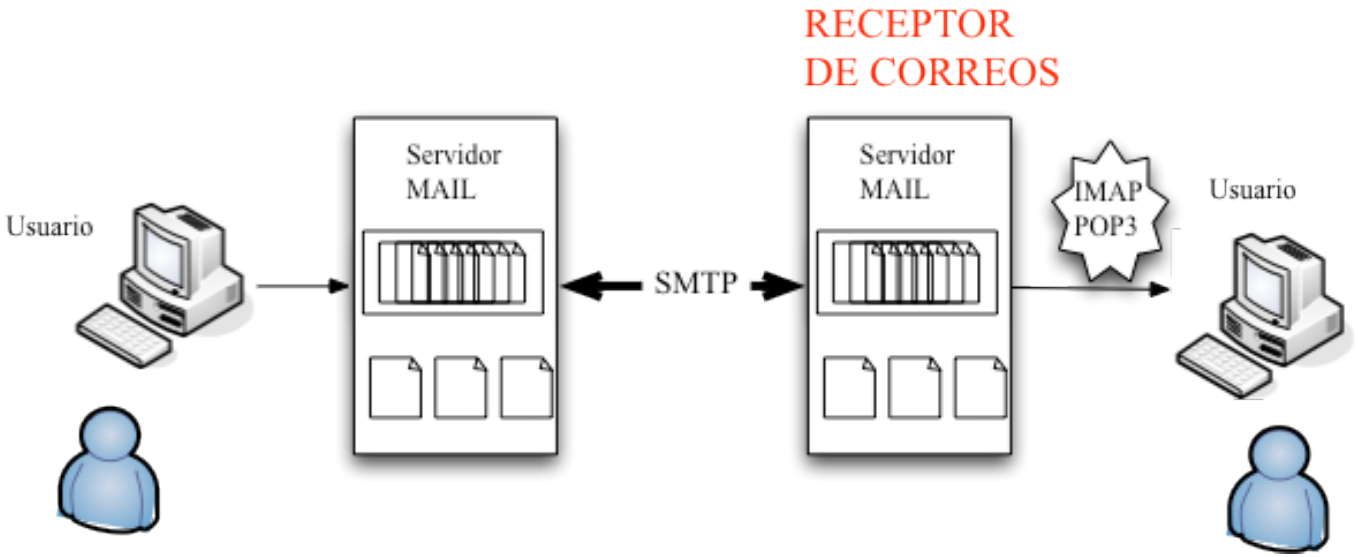
Protocolo de Acceso al Correo

- Antes de los 90 bastaba con logearse a la máquina.
- Después de los 90, arquitectura C/S.
- El problema está en mantener encendido (siempre) el srv de correos para que msg. lleguen.

Protocolo de Acceso al Correo

- ¿Cómo leer el correo que está en el servidor?
- POP3 (post office protocol) → el más simple
  - RFC 1939. Autorización (agente <--> servidor) y descarga.
- IMAP (Internet Mail Access Protocol).
  - RFC 1730. Más características. Almacena mensajes en servidor.
- HTTP: gmail, hotmail, yahoo, etc.

Repaso: ¿Dónde se encuentra un correo en un momento dado?



POP3

- Protocolo de acceso al mail muy simple
- Se inicia conexión TCP al puerto 110
- Tres Etapas: Autorización, transacción y actualización.
  - Autorización: login/password
  - Transacción: Obtener msgs, marcar msgs., estadisticas.
  - Actualización: Borrar definitivamente los msgs. marcados.

POP3 - transacción

- El usuario puede configurar: “download-delete”/”download-keep”
- Transacción Definidas en RFC 1939: list, retr, dele, quit.
- El problema esta con los clientes nómadas, si usa “download-delete” pierde información en otros dispositivos.
- No mantiene información entre sesiones POP3.

Ejemplo de Transacción

Autorización →

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

Transacción →

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

IMAP

- Posible mantener una jerarquía de carpetas en el servidor (con msgs en cada carpeta) RFC 3501.
- Más complicado (y pesado en tráfico) que POP.
- Pueden obtenerse partes de un mensaje.

DNS: Domain Name System

- Mas fácil utilizar nombres que números (problema de naming/binding). ¿Cómo identificamos personas?
- Para un host usamos un nombre: www.ula.ve, webdelprofesor.ula.ve
  - podemos saber ubicación geográfica a través del DNS
- Para procesar rápidamente a través de ruteadores se utilizan las direcciones IP (4 números entre 0-255, separados por punto).

Servicios del DNS

- Base de Datos distribuida implementada en jerarquía de servidores.
- Protocolo de Nivel aplicación que permite hacer preguntas
  - Sobrenombres de Hosts (aliasing)
  - Mail server
  - Traducción de nombre a IP
- Implementación más conocida: BIND (Berkeley Inet Name Domain)
- Utiliza UDP/puerto 53

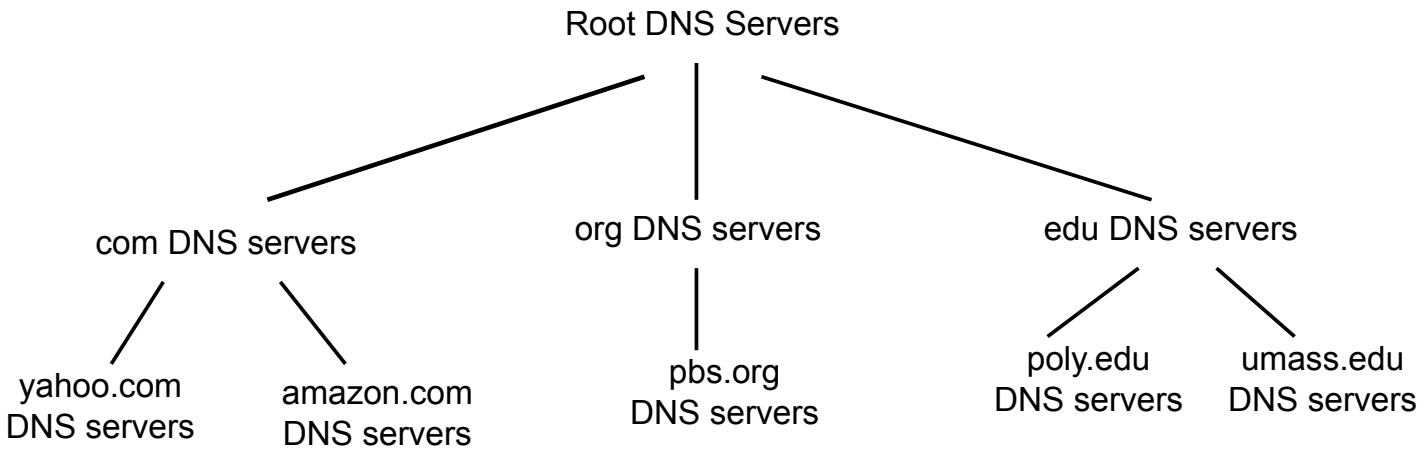
¿Cómo Funciona?

- Se hace la pregunta al Srv. DNS por una dirección IP.
- Cliente recibe una respuesta eventualmente con la dirección IP.
- Luego se pueden iniciar las conexiones TCP.
- Frecuentemente, se usa un cache DNS para acelerar tiempo de respuesta.
- RFC 1034 y 1035.

Tipos de Nombres

- Alias para host: Un nombre complicado puede tener sinónimos.
  - Nombre de base → Nombre canónico (largo y pesado)
  - Otros nombres → Deberían ser más mnemónicos.
- Alias para serv. Mail: Servidor mail también debe ser mnemonico.
  - **Extremo**: Llamar al servidor web y mail con el mismo nombre.
- DNS permite distribuir la carga: varios IP con el mismo nombre.

Arquitectura del DNS



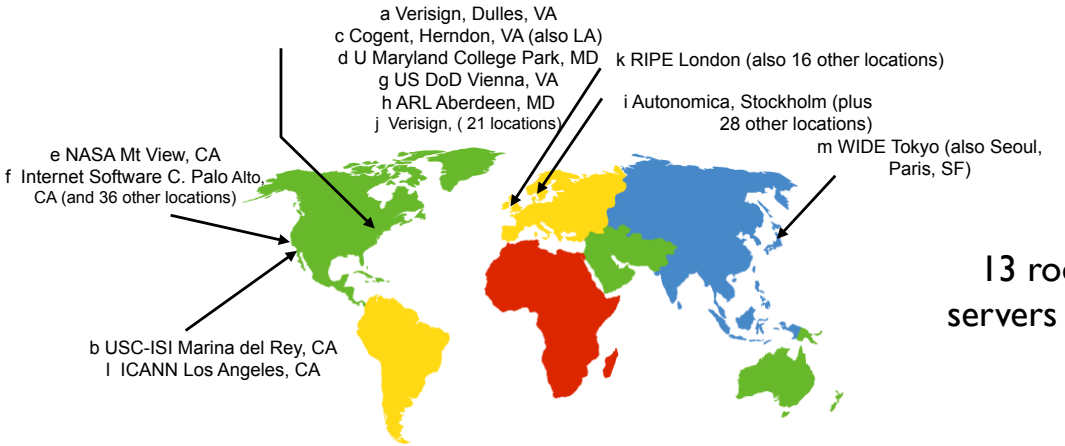
- Cliente pregunta por “com” en el Root Server
- Luego se pregunta a “com” por el “amazon.com”
- Finalmente se preguta a “amazon.com” por el www.amazon.com

¿Cómo funciona DNS?

- En maquinas UNIX la llamada sistema: gethostbyname().
- Pregunta/Respuest al puerto 53

Servidores Raíz (Root) en el mundo

- Contactado por el servidor local de nombres que no puede resolver.
- Servidor raíz:
  - Contacta al servidor autoritativo si no conoce el mapeo.
  - Obtiene mapa
  - Devuelve mapa al servidor DNS local



13 root name servers worldwide

DNS Simple

- Un solo servidor DNS con todos los registros?
  - Punto simple de falla → si falla, falla la Inet.
  - Volumen de trafico muy grande.
  - Base de Datos distante: no esta cerca de todos los clientes → uso injusto de recursos.
  - Mantenimiento: tener todos los reg. de la internet hace un servicio pesado (hasta imposible).
- Conclusion: Una BD Centralizada **NO ESCALA**.

BD distribuida/jerárquica

- Para lidiar con la escala, hay 3 tipos de servidores DNS:
  - Raices
  - Dominio de Nivel Alto (Top-Level Domain)
  - Autoritativos

Tipos de Servidores DNS

- DNS Raíz: Hay una lista de servidores raíces (13 en total) → propósitos de confiabilidad y seguridad (1 srv = 1 cluster).
- Top-Level Domain : Se encargan de grandes porciones de dominio (com, org, edu y paices: ve, fr, us).
  - Dividido en compañias Network Solutions (.com), Educause (.edu)
- Autoritativo: Alberga los registros que hacen el mapping entre hosts e IPs
  - Se hace dentro de la institución o se paga por el servicio.

Servidor DNS local

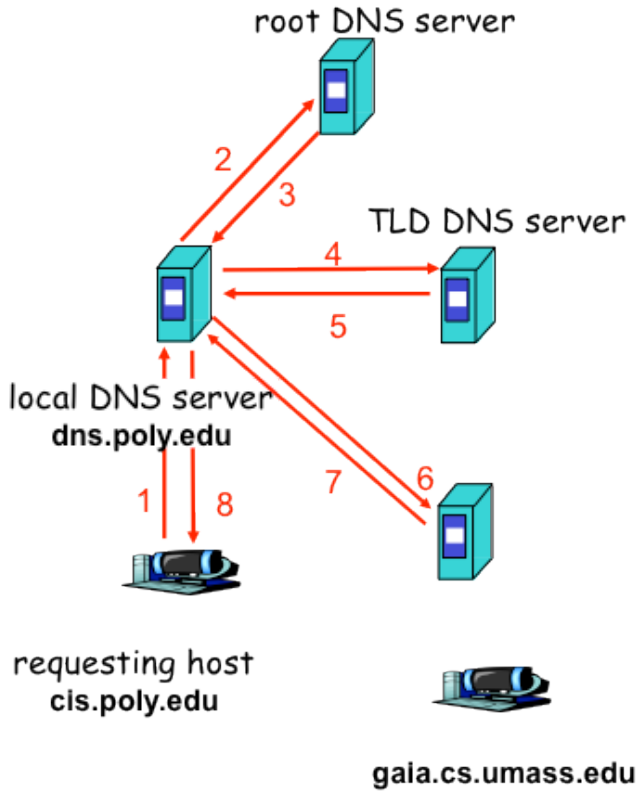
- No pertenece a la jerarquía estrictamente.
- Cada ISP o compañía tiene uno por omisión (empresa, universidad, etc).
- Cualquier solicitud es enviada al servidor DNS local.
  - Actua como proxy y reenvia las solicitudes desconocida
- Se obtiene por DHCP
- Tipicamente cercano al host
- Puede contribuir a generar por encima de los 8 mensajes previos... si cada subdivisión de una empresa tiene su propio DNS.

Consulta DNS Iterativa

Host cis.poly.edu desea el IP de gaia.cs.umass.edu

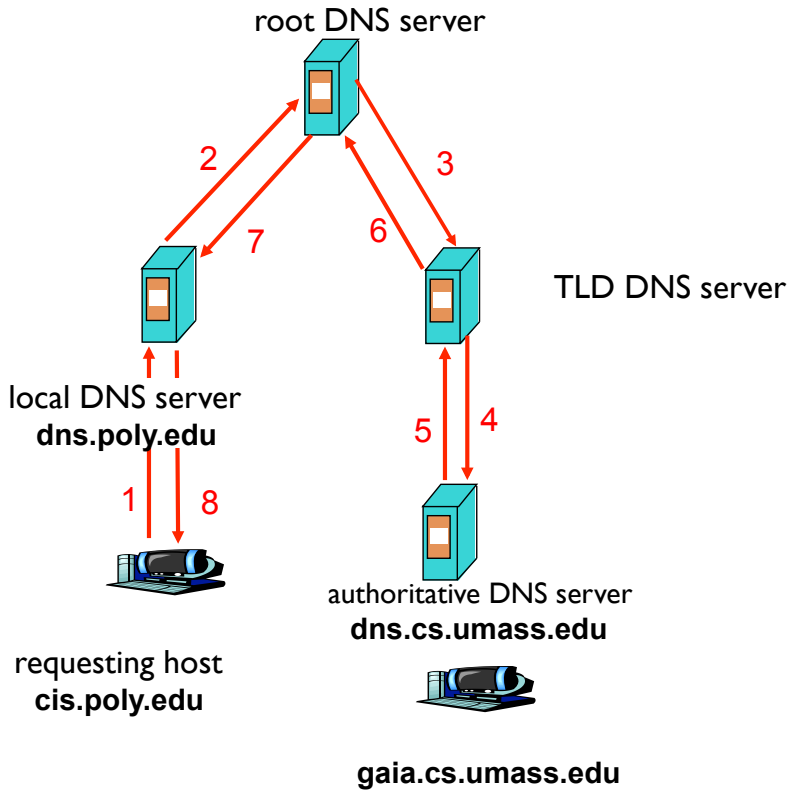
Consulta Iterativa:

- La consulta devuelve el próximo servidor a consultar.
- No conozco el servidor pero pregunte en esta otra dirección...



Consulta DNS Recursiva

- Pregunta recursiva
  - Pone la carga de la resolución de nombres en el servidor contactado.
  - ¿Incrementa la carga? ¿Dónde?



Caching DNS

- Almacena respuestas intermedias en memoria local para usos futuros.
  - Desaparecen con el tiempo
  - Guarda los nombres de los servidores TLD
- Ahorra mensajes explícitos.
- El diseño de los reemplazos/actualizaciones se hace con el RFC 2136.

Registros/Msg del DNS

- Un registro válido es una 4-tupla:
  - <Nombre, Valor, Tipo, TTL>
- Tipo **NS**: Nombre = Dominio, Valor = Servidor DNS autoritativo (para hacer reenvío de una consulta).
- Tipo **A**: Nombre = Host, Valor = IP. → se usa para consultar servidores autoritativos
- Tipo **MX**: Provee el nombre canónico para el servidor Mail
- Tipo **CNAME**: Para obtener el nombre canónico de un host cualquiera. Nombre = alias, valor = nombre real.

Ejemplos

- Intente las siguientes lineas de nslookup:
  - nslookup www.ula.ve
  - nslookup smtp-mail.ula.ve
- ¿Qué concluye después de hacer las consultas?
- Discuta el applet DNS

Dinamica DNS

- Para insertar registros en la BD DNS → buscar un registrar. (www.internic.net)
- Toda autoridad está certificada por el ICANN (Inet Corporation for Assigned Names & Numbers)
- Mínimo número de registros a suministrar:
  - <compañia.com, dns1.compañia.com, **NS**>
  - <dns1.compañia.com, 150.150.150.10, **A**>
- Pero no olvide también los tipo A de WWW y MAIL.

Mensajes Solicitud/Respuesta

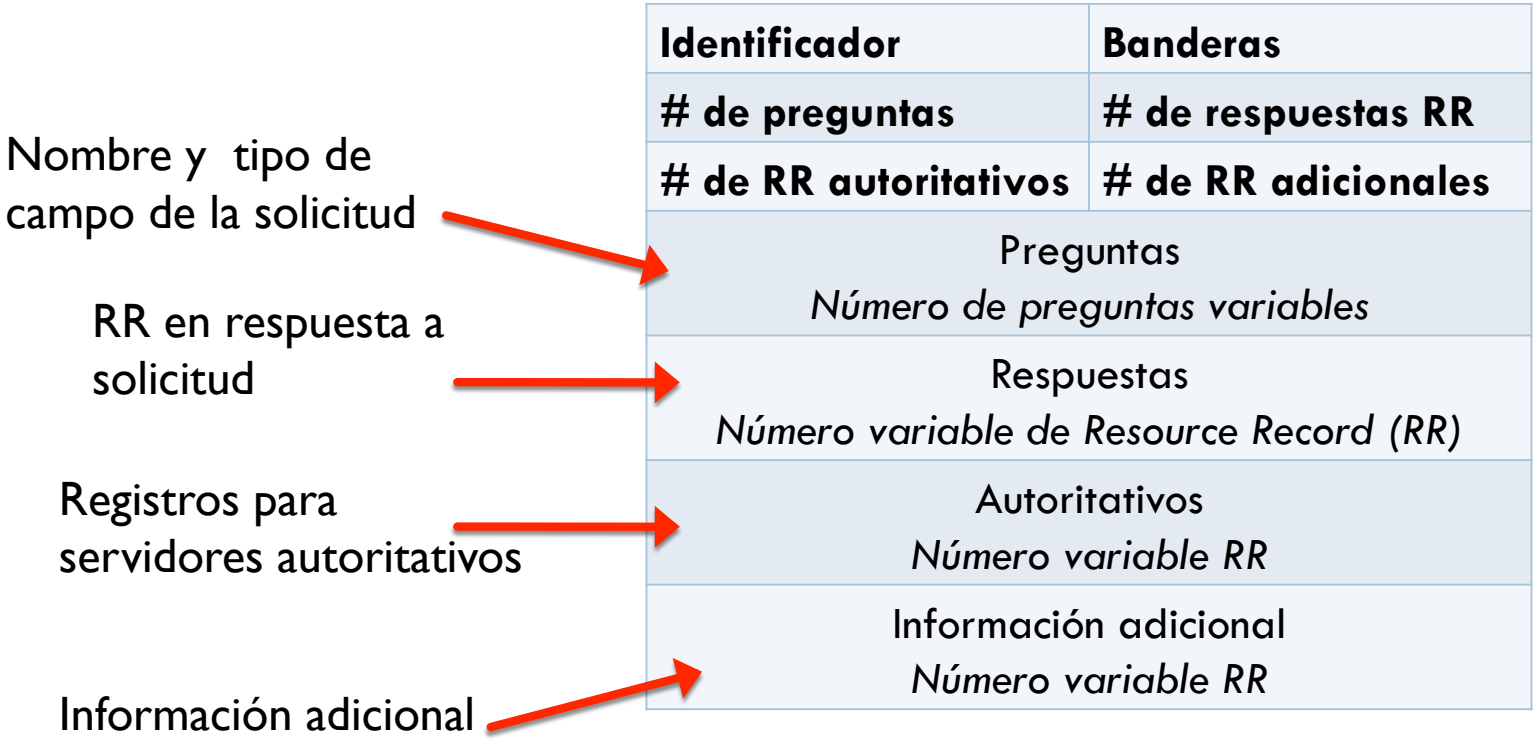
- Los mensajes solicitud/respuesta tienen el mismo formato.

Encabezado:

- Identificador: cada campo es de 16 bits (para solicitudes y respuestas).
- Banderas:
  - Solicitud o respuesta
  - Recursividad solicitadas
  - Recursividad disponible
  - Respuesta autoritativa

Identificador	**Banderas**
# de preguntas	**# de respuestas RR**
# de RR autoritativos	**# de RR adicionales**
Preguntas *Número de preguntas variables*	
Respuestas *Número variable de Resource Record (RR)*	
Autoritativos *Número variable RR*	
Información adicional *Número variable RR*	

Anatomía del Mensaje DNS



Inserción de registros DNS

- Comencemos una compañía nueva: “Rapid Net”
- Se registra entonces el nombre rapidnet.com en algúna oficina de registro DNS (registrar)
  - Se provee el nombre y dirección IP de un servidor autoritativo (primario y secundario)
  - La oficina DNS guarda dos RR en el servidor TLD:

    (rapidnet.com, dns1.rapidnet.com, NS)

    (dns1.rapidnet.com, 150.185.23.12, A)
- Se crea el registro autoritativo tipo A y también el tipo MX para rapidnet.com
- ¿Cómo puede obtenerse la dirección IP del sitio web?

III

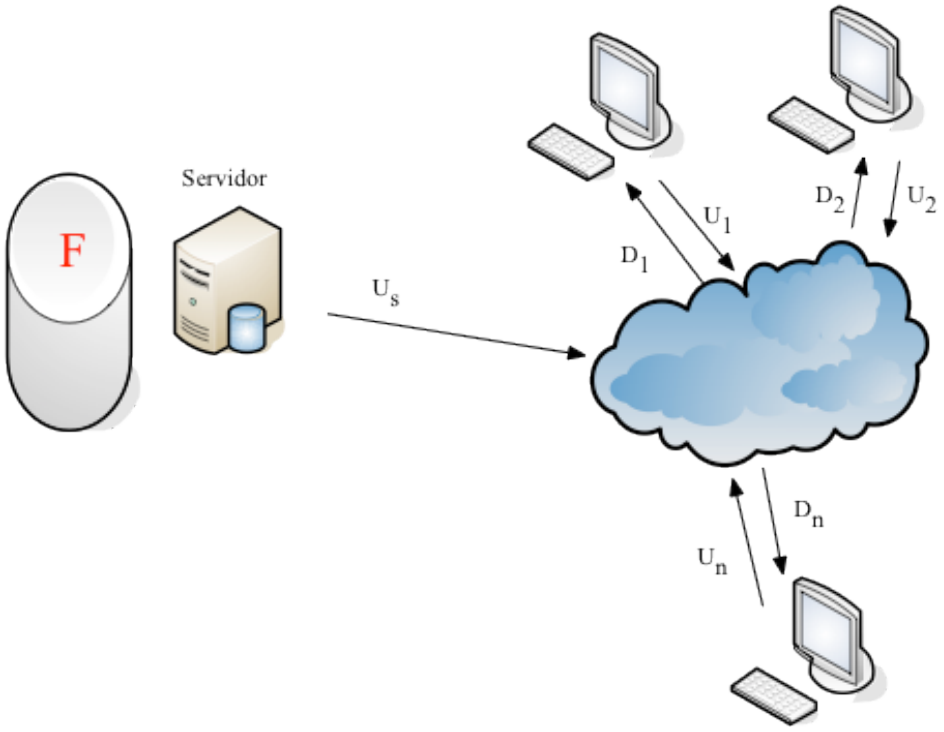
Ejercicio Práctico

Hacer práctica DNS.

Aplicaciones P2P

- HTTP y DNS descansan sobre el supuesto de que los servicios están siempre encendidos.
- P2P hosts comunican directamente con otros hosts
  → Estudiaremos 3 Aplicaciones:
  - Distribución de archivos
  - BD distribuida en una gran comunidad
  - Skype

Distribución de Archivos



U_s: BW-upload Servidor

U_i: BW-upload Peer

d_i: BW-dwload Peer

F: Tamaño del archivo

Distribución (condiciones)

- Tiempo de distribución: tiempo en llevar una copia a **todos los peers.**
- Se asume que en el core hay suficiente BW
- C/S no participan en otras applic. → todo el BW para P2P

Distribución **Cliente/Servidor**

- Se transmite una copia a los N peers = N.F bits

$$t = \frac{NF}{U_s}$$

- d_{min} corresponde a la tasa de download más baja.

$$C_{peor} = F/d_{min}$$

$$d_{min} = \min(d_1, d_2, ..., d_n)$$

- Peor cliente tarda

$$D_{cs} \geq max\{F/d_{min}, NF/U_s\}$$

- Para N lo suficientemente grande

$$D_{cs} = NF/U_s$$

Distribución **P2P**

- Envía el archivo al menos una vez a la red (Tmin):

$$D_{p2p} = F/U_s$$

- Mínimo tiempo en cargar un archivo, es:

$$D_{p2p} = F/d_{min}$$

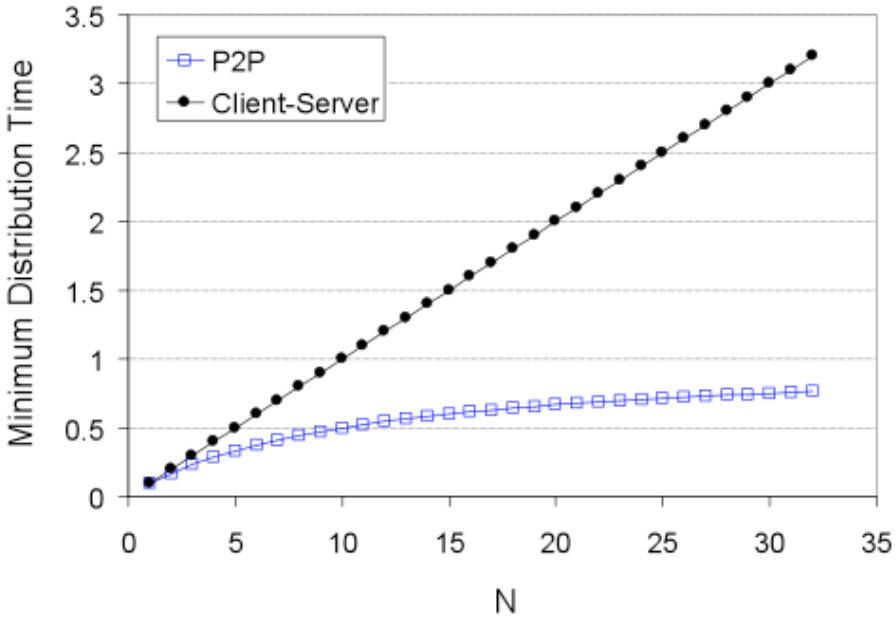
- Tiempo de carga (upload paralelo) y distribución (cada peer colabora de inmediato)

$$u_{total} = u_s + u_1 + u_2 + ... + u_n$$
$$D_{p2p} = \frac{NF}{u_s+u_1+u_2+...+u_n}$$

- Tiempo de distribución (cada peer envía cada bit tan pronto como lo recibe)

$$D_{P2P} \geq max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s+\sum_{i=1}^{N} u_i}\}$$

Comparación

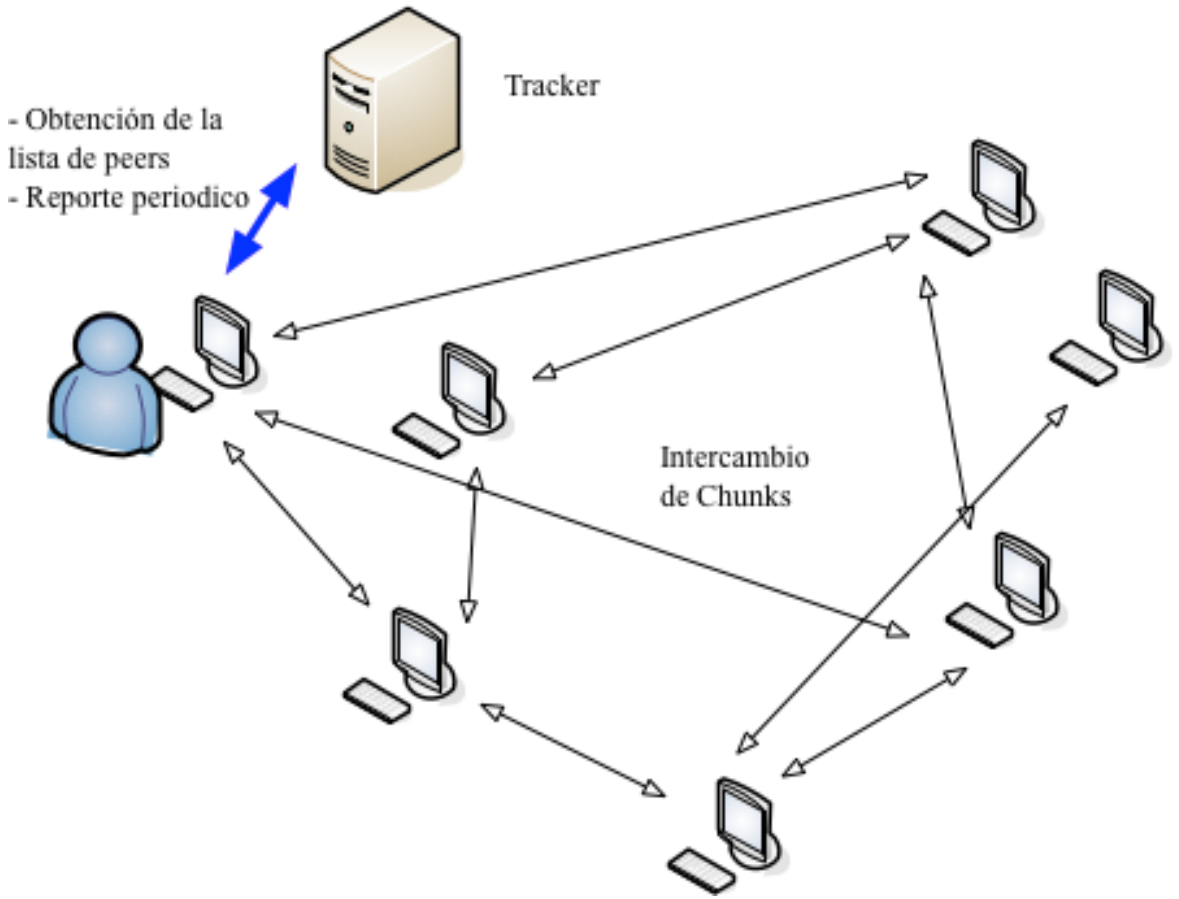


- No hay problema con:
  - La salida (U_s) del servidor
  - La descargas de los clientes.
- $F/u=1hr$, $u_s=10u$, $d_{min}>>u_s$.
- Observe la autoescalabilidad del sistema (menos de 1hr para cualquier carga!)

BitTorrent

- Protocolo P2P para distribución de archivos
- **Torrent**: grupo de peers que participan en la distribución
- Descargas de Chuncks de 256 KB desde los peers.
- Peers entran y salen a voluntad.
  - Cuando se entra no se dispone de chunks.
  - Se pueden ir cuando se tengan todos (egoista) o quedar cuando se tengan (altruista)

Topología de BitTorrent



BitTorrent (2)

- Tienen un **tracker**: todo peer se registra y reporta periódicamente al tracker.
- Cantidad variable de peers (entre 10 y 1000)
- Un peer llega y obtiene un listado de otros peers y se intenta conectar con ellos. → si hay éxito se obtienen peers-vecinos.
- Se pregunta por lista de chunks a los vecinos
  - Se piden los chunks que faltan. Pero a quien?
  - Se otorgan chunks. Pero a quien?

BitTorrent (3)

- ¿Qué chunks deben escogerse?
  - Técnica “el más ausente primero”: los que tienen menos copias en los vecinos.
  - Se hacen propagar mas rápido para estabilizar el sistema.
- ¿A que solicitudes se responde?
  - A los Peers que **DAN** data a la tasa más alta.
  - Recuerde que en el cliente se puede “estabilizar” la tasa de transferencia → mas rapido a la larga
- Se escogen los 4 peers más rápidos
- Cambiar de conjunto de peers cada 10 seg.
- Siempre tendremos beneficios de un peer hasta conseguir uno mejor. → tasas compatibles se buscan entre sí.

Tablas Hash Distribuidas

- Para encontrar el contenido distribuido en una red
- Resuelven el problema de guardar pares <clave, nombre> (donde se reparte el contenido, es decir, los chunks)
- Se requieren miles de entradas en un sistema centralizado (ej. Napster) no da a basto.
- Se usa un sistema altamente distribuido.

Tabla Hash Distribuidas

- Se distribuyen los pares (clave, valor) entre los participantes.
- Cada **Peer** se identifica con un número.
- Se calcula la posición hash y se envía al peer cuyo valor hash sea el más cercano.
- Visión Central → Se necesitaría llevar cuenta de todos los peers (resulta imprácticо).
  - TH Distribuidas no se puede escalar porque cada peer lleva cuenta de otros peers.

Tabla Hash Circulares

- Resuelve el problema de la escala.
- Se conoce solamente el sucesor inmediato.
- Se envían O(N/2) msg en promedio.
- Cuando se encuentra el responsable se regresa el msg (encontrado!) al origen (en corto-circuito).
- Los atajos van dandole rapidez a la connexión.
- Pero, ¿Cuántos atajos en total?

Peer Churn (ent/sal peers)

(mezcladora de Peers)

- Cada peer entra y sale a su antojo.
- Originalmente cada peer conoce su 1er y 2do sucesor.
- Se verifica periodicamente que los sucesores estén vivos (se llaman msg. ping).
- Si un peer abandona (...al estilo de listas dobles).
  - (A) 2do sucesor pasa a ser 1ro
  - (B) 1er sucesor de suc(A) es 2do sucesor de (A).
  - *El que llega*: manda msg a un conocido y luego aplica busqueda secuencial para su <1ro, 2do>.

Skype

- Entre 9-11 Mg de usuarios conectados simultaneamente.
- Servicios PC-a-teléfono, teléfono-a-PC, telefonía PC-a-PC.
- Provee técnicas de loc. de usuarios y puenteo de NATs.
- Sistema propietario y encriptado.
- Dos modos: Peer ordinario y Peer Super-nodo.
- Mapeo de usuario a IP de la máquina → se distribuyen entre super-nodos.

Skype

- Suponga que tiene dos entes detras de un NAT c/u
- En problema es que ningún nodo detras de un NAT puede iniciar la conexión
- **Super-peers + relays** resuelven el problema
- Ambos super-peers escogen un **Relay** para comunicarse.
- Ocurre comunic. entre super-peers para saltar los NATS (via el Relay). Aquí radica el **éxito** de skype.

Ejercicio Práctico

Práctica dirigida de desensamblaje de tramas.