TEMA 1 | CAPA DE APLICACIÓN

PARTE 1 - INTRODUCCIÓN

① Conceptual: aspectos de implementación de protocolos de aplicación
- Modelos de servicio de la capa de transporte
- Paradigma cliente-servidor
- Paradigma Peer-to-Peer (P2P)

② Aplicación: aprender el uso de los protocolos de aplicación más comune
HTTP, FTP, SMTP/POP3/IMAP, DNS ...

③ Progmación de aplicaciones de red con la API de sockets.

CARACTERÍSTICAS DE APLICACIONES

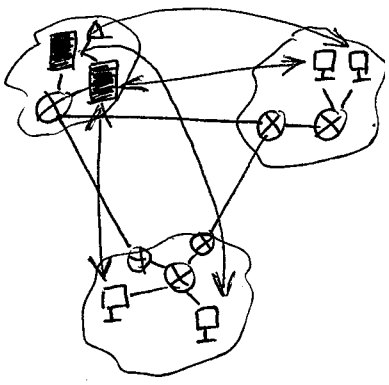
Los programas deben tener las siguientes características:
- Deben ejecutarse en varios sistemas finales
- Comunicarse a través de red
- Ej: servidor web se comunica con el browser.

No se necesita escribir software para los elementos de red. Los elementos intermedios de red no ejecutan aplicaciones de usuario. Las aplicaciones en los extremos de la conexión permiten un desarrollo rápido y simple.

ARQUITECTURAS DE APLICACIÓN

- cliente-servidor
- entre pares (peer-to-peer P2P)
- híbrido cliente-servidor y P2P

▶ CLIENTE-SERVIDOR



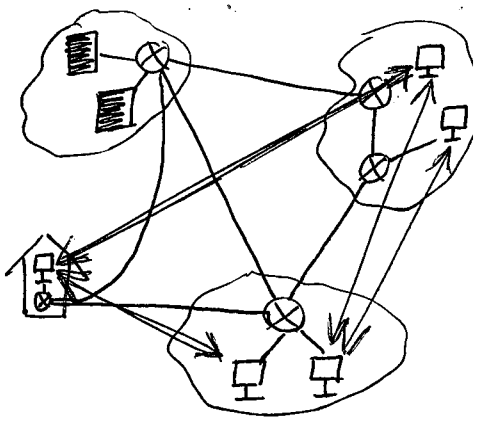
Servidor:
- siempre es un host
- dirección IP permanente
- granjas de servidores para el escalamiento

Cliente:
- se comunica con el servidor
- puede comunicarse intermineutemente
- puede tener IP dinámica
- los clientes no se comunican entre si.

▶ P2P

- Descentralizada:
  - ausencia de un servidor central para el control
  - los participantes pueden comunicarse directamente entre si.
  - todos los nodos actúan como clientes y servidores.
- Distribuida:
  - la información no está alojada en un solo sitio.

- Carga balanceada:
  - se intenta equilibrar la carga entre todos los participantes.
- Redundancia de información:
  - se duplica la información para hacerla más accesible
- Alta disponible
  - la caída de un host no bloquea el servicio.



- Optimización de uso de recursos
  - procesamiento, almacenamiento, ancho de banda, etc.
- Ejemplos: Kazaa, eMule, WinMX, Gnutella, BitTorrent, LimeWire

HÍBRIDAS

Skype:
- aplicación de voz sobre IP.
- servidor centralizado: permite encontrar la dirección remota con la que se quiere conectar.
- conexión directa cliente-cliente.

Mensajería instantánea:
- La comunicación se realiza entre dos usuarios en P2P.
- Servicio centralizado
  - → presencia de cliente: un usuario registra su IP asociada a su usuario cuando se registra
  - → localización: un usuario localiza a la persona con la que quiere comunicar consultando al servidor.

COMUNICACIÓN ENTRE PROCESOS

Definición de proceso: programa en ejecución en un sistema informático

- Dentro del sistema informático los procesos se comunican utilizando la comunicación entre procesos (IPC) definida por el sistema operativo.
- Entre diferentes sistemas informáticos se comunican utilizando el intercambio de mensajes.

El proceso cliente es el proceso que inicia la comunicación

El proceso servidor es el que espera peticiones de los clientes.

NOTA: Las aplicaciones P2P tienen procesos clientes y servidores en todos los sistemas informáticos en los que están funcionando.

Los procesos envían y reciben mensajes a y desde la red a través de sus sockets. Los sockets son análogos a puertas

En el proceso transmisor se sacan mensajes por la puerta y confía en la estructura de transporte al otro lado de la puerta, la cual lleva los mensajes al socket del proceso receptor

El proceso receptor recibe mensajes por la puerta, la cual los pasa al proceso receptor.

- API (Application Programming Interface)
  - Se puede elegir el protocolo de transporte (UDP, TCP, ...)
  - Se pueden seleccionar muchos parámetros del protocolo

Para que un proceso reciba un mensaje este debe tener un identificador único. Un host tiene una dirección IP única de 32 bits.

¿Es suficiente con la dirección IP para identificar al proceso? No → es necesario incluir un número de puerto para identificar a un proceso dentro del host.

Ejemplos de puertos: Servidor HTTP: 80
Servidor e-mail: 25

En UNIX (linux) están descritos en /etc/services

Tipos de protocolos

- dominio público (abiertos)
  - definidos en RFCs
  - permite interoperabilidad
  - HTTP, SMTP...
- propietarios
  - restringido el acceso a las definiciones por las compañías propietarias
  - en general no permite interoperabilidad
  - Skype, Messenger...

Servicios al transporte en la aplicación

- Transferencia de datos confiable:
  - algunas aplicaciones (audio/video) pueden tolerar pérdida
  - la mayoría (FTP, telnet) requieren transferencia 100% confiab
- Tasa de transferencia:
  - aplicaciones con ancho de banda sensitivo (bandwith-sensitive application) como apps multimedia
  - aplicaciones elásticas: e-mail, FTP.
- Retardo:
  - algunas aplicaciones (telefonía IP, juegos interativos, teleconferencias) requieren bajo retardo.
- Seguridad:
  - integridad de datos, encriptación, autentificación, etc

Servicios de protocolos de transporte en Internet

- Servicio TCP:
  - Orientado a la conexión: se requiere un acuerdo entre cliente y servidor.
  - Transporte confiable: sin pérdidas.
  - Control de flujo: el transmisor no sobrecargará al recepto
  - Control de congestión: frena al transmisor cuando la red esté sobrecargada.
  - No provee: garantía de retardo ni ancho de banda mínin
- Servicio UDP:
  - Transferencia de datos no confiable entre los procesos transmisor y receptor.
  - No provee acuerdo entre los procesos, confiabilida control de flujo, control de congestión, ni garantías de retardo o ancho de banda.

PARTE 2 - PROGRAMACIÓN DE SOCKETS

TIPOS DE SOCKET EN FUNCIÓN DE LOS SERVICIOS PROPORCIONALES

- Stream sockets: confiables, orientados a conexión. Servicios de TCP.
- Datagram sockets: no confiables y no orientados a conexión. Servicios de UDP.
- Raw sockets: acceso a niveles más bajos del protocolo TCP/IP.

STREAM SOCKETS (SOCK_STREAM) - TCP

- Define una conexión confiable
- Los datos se envían sin errores y en el mismo orden en el que se envían.
- Control de flujo
- Sin límites a los datos (flujo continuo o stream).
- Ejemplo: FTP.

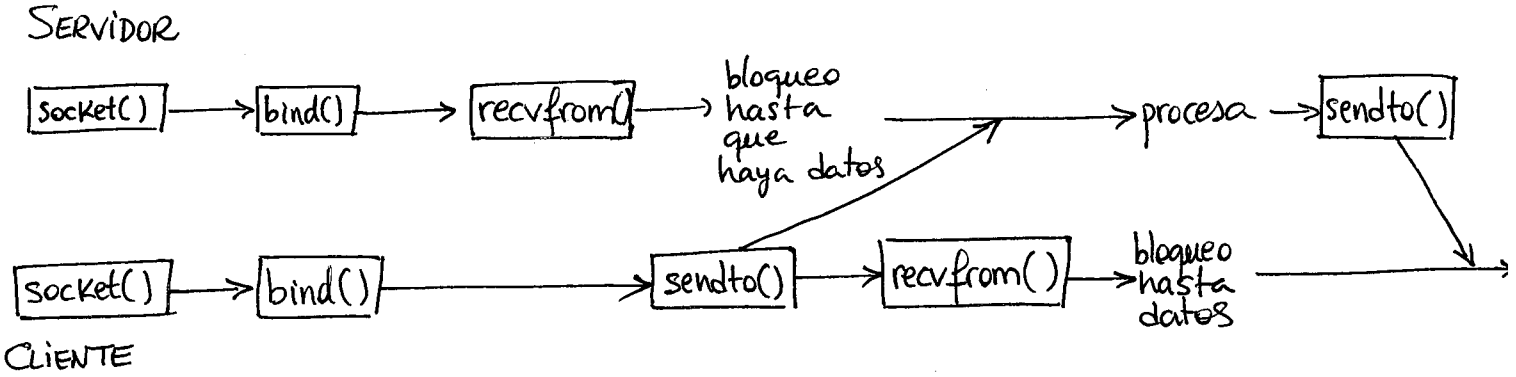
DATAGRAM SOCKETS (SOCK_DGRAM) - UDP

- Define un servicio no orientado a la conexión (sin "handshake").
- Los datagramas se envían como paquetes independientes.
- El emisor añade explícitamente la IP_dest y PD a cada paquete
- El receptor extrae la IP_dest y PD del paquete.
- Sin garantías de llegada ni de orden
- Sin fragmentación
- Ejemplo: NFS.

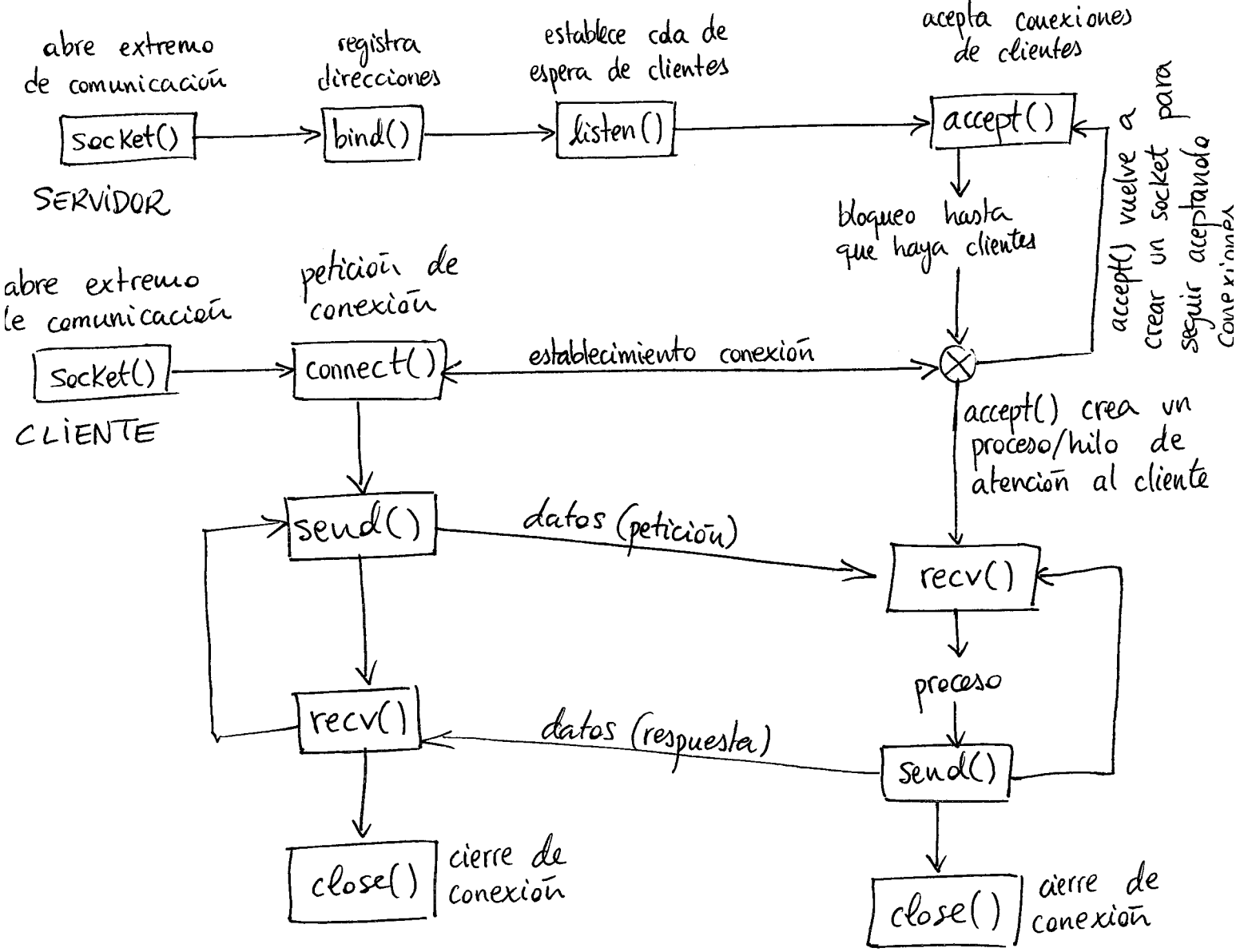
RAW SOCKETS (SOCK_RAW)

- Acceso a los niveles más bajos de la pila de protocolo (IP, ICMP).
- Se utiliza a menudo para probar nuevos protocolos
- Ejemplo: aplicación ping

ESQUEMA : NO ORIENTADO A LA CONEXIÓN



ESQUEMA : ORIENTADO A LA CONEXIÓN



BIBLIOTECA DE SOCKETS

`int sockfd = socket(int family, int type, int protocol)`

Función: iniciar un socket

Argumentos:

- family: indica la familia de direccionamiento
  AF_UNIX, AF_INET, AF_NS, AF_OS2 y AF_IUCV
- type: indica el tipo de interfaz de sockets a usar
  SOCK_STREAM, SOCK_DGRAM, SOCK_RAW y SOCK_SEQPAQUE
- protocol: protocolo, en el caso de INET: UDP, TCP, IP o ICMP

Retorno: sockfd
descriptor de fichero del socket.

`int bind(int sockfd, struct sockaddr *localaddr, int addlen)`

Función: registrar un socket en un puerto

Argumentos:

- sockfd: file descriptor del socket
- localaddr: depende de la familia del socket
  estructura de parámetros.
- addlen: sizeof(localaddr)

Retorno: 0 en caso de éxito. En error devuelve -1 + errno.

`int listen(int sockfd, int queue-size)`

Función: indicar la cantidad de peticiones de conexión que pueden ser encoladas.

Argumentos:

- sockfd: file descriptor del socket.
- queue-size: nº de peticiones de conexión que pueden ser encoladas por el sistema antes de que el protocolo local ejecute un accept.

Retorno: 0 en caso de éxito. En error devuelve -1 + errno.

`int accept(int sockfd, struct sockaddr *foreign-addr, int addrleu)`

Función: extrae la primera conexión de la cola de peticiones

Argumentos:
- sockfd: file descriptor del socket
- foreign-addr: estructura devuelta por la función con la información de la conexión entrante.
- addrleu: tamaño de la estructura.

⊛ Si no hay peticiones, se bloquea el proceso hasta que llegue alguna.

Retorno: en caso de éxito devuelve el descriptor del socket aceptado. En caso de error, devuelve -1 + errno.

`int connect(int sockfd, struct sockaddr *foreign-addr, int addrlen)`

Función: conectarse con un servidor

Argumentos:
- sockfd: file descriptor del socket.
- foreign-addr: estructura con la información de la dirección remota a la que se conecta.
- addrleu: sizeof(foreign-addr)

Retorno: 0 en éxito, en error devuelve -1 + errno.

```
size_t readv(int s, void *buf, size_t lon, int flags)
size_t recv(int s, void *buf, size_t lon, int flags)
size_t recvfrom(int s, void *buf, size_t lon, int flags, struct sockaddr *desde,
                socklen_t *longdesde)
size_t recvmsg(int s, struct msghdr *msg, int flags)
size_t read(int fd, void *buf, size_t nbytes)
size_t send(int s, const void *msg, size_t leu, int flags)
size_t sendto(int s, const void *msg, size_t leu, int flags, const struct
              sockaddr *to, socklen_t toleu)
size_t sendmsg(int s, const struct msghdr *msg, int flags)
size_t write(int fd, void *buf, size_t nbytes)
```

`int close(int sockfd)` Función: cerrar la conexión y liberar las ED.

PARTE 3 - HTTP (HyperText Transfer Protocol)

JERGA BÁSICA

Una página web está constituida por objetos (fichero en HTML, imagen JPEG
Una página web está constituida por un fichero HTML base que incluye referencias a objetos. Cada objeto es direccionable por una URL.

URL: Unified Resource Locator
HTML: HyperText Markup Language

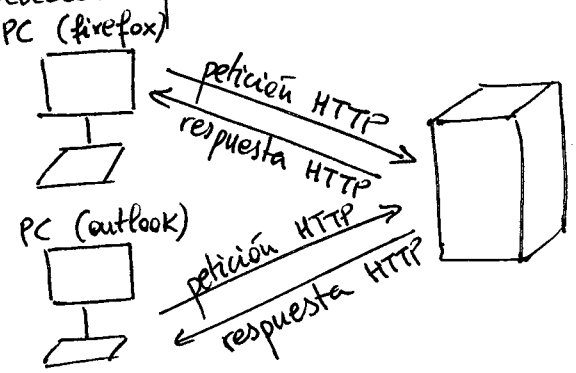
GENERALIDAD DE HTTP

HTTP es un protocolo de capa de aplicación para la web

Sigue un modelo cliente-servidor:

cliente: browser que solicita, recibe y muestra objetos web.

servidor: el servidor recibe peticiones, responde y envía objetos.



FUNCIONAMIENTO HTTP

- Utiliza TCP
  1. El cliente inicia la conexión TCP con el servidor (puerto 80)
  2. El servidor acepta la conexión TCP del cliente
  3. Se intercambian mensajes HTTP entre el browser (cliente HTTP) y servidor.
  4. Se cierra la conexión.
- HTTP no conserva el estado: el servidor no mantiene información sobre las peticiones del cliente.
- Definición de RTT: tiempo ocupado en enviar un pequeño paquete desde el cliente servidor y su regreso.
- Tiempo de respuesta: un RTT para iniciar la conexión, un RTT por requerimient HTTP y primeros bytes de la respuesta + tiempo de transmisión del archivo
- Tipos de conexiones
  - HTTP NO PERSISTENTE: como mucho se envía objeto por conexión TCP. HTTP/1.0 usa HTTP r persistente.
  - HTTP PERSISTENTE: en una conexión TCP pued enviarse múltiples objetos entre el cliente y el servidor. HTTP/1.1 usa conexiones persistent de forma predeterminada.

HTTP NO PERSISTENTE:

Supongamos que se introduce URL: www.escuela.edu/departamento/index.html

1. El servidor espera por conexiones en el puerto 80 de el host www.escuela.edu
2. El cliente HTTP inicia la conexión TCP con el servidor www.escuela.edu en el puerto 80.
3. El servidor acepta la conexión.
4. El servidor le comunica al cliente la aceptación.
5. El cliente HTTP envía el mensaje de solicitud al socket TCP con la URL, indicando que se solicita el objeto departamento/index.html
6. El servidor recibe la petición y envía un mensaje de respuesta conteniendo el objeto solicitado y lo envía a su socket.
7. El servidor cierra la conexión TCP.
8. El cliente recibe el mensaje de respuesta conteniendo el objeto solicitado, lo analiza, lo presenta y encuentra 10 referencias a otros objetos.

: Con las referencias encontradas el cliente repite todo el proceso hasta que todas las respuestas a objetos están resueltas.

Problemas de HTTP no persistente:

- Requiere 2 RTTs por objeto
- OS debe trabajar y dedicar recursos para cada conexión TCP.
- El navegador abre conexiones paralelas generalmente para traer objetos referenciados

TTP PERSISTENTE:

- El servidor deja las conexiones abiertas después de enviar la respuesta
- Mensajes HTTP subsequentes entre los mismos cliente/servidor son enviados por la

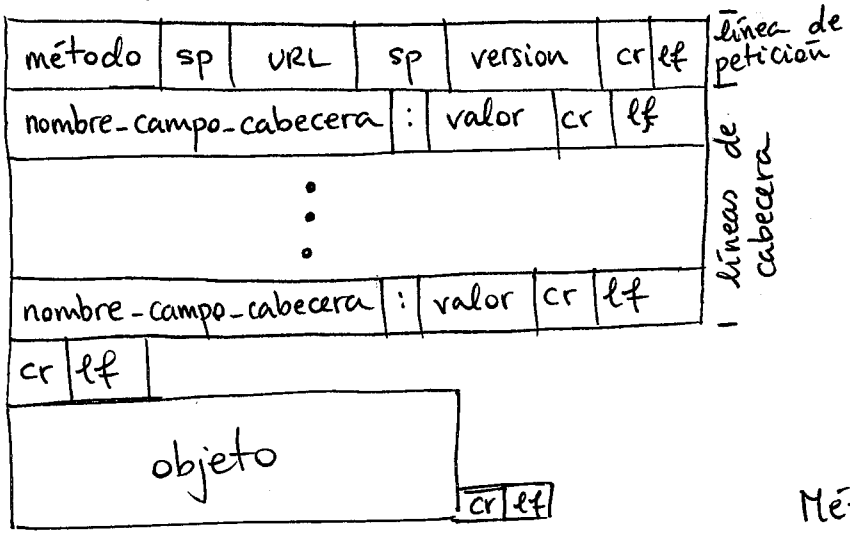
conexión abierta.

▸ Persistencia sin "pipelining":
  - El cliente un nuevo requerimiento sólo cuando el previo ha sido recibido.
  - Un RTT por objeto referenciado.

▸ Persistencia con "pipelining":
  - Predeterminado en HTTP/1.1
  - El cliente envía requerimientos tan pronto este encuentra un objeto referenciado
  - Tan pequeño como un RTT por todas las referencias.

PETICION HTTP

FORMATO



MÉTODOS

HTTP/1.0
- GET
- POST
- HEAD

HTTP/1.1.
- GET
- POST
- HEAD
- PUT
- DELETE

Método GET: la URL del objeto solicitado está en la cabecera del mensaje

Método POST: el servidor suele disponer de un formulario de petición. La petición se realiza según ese formulario como objeto del mensaje.

Resumen de métodos:

GET: obtiene un recurso del servidor. Sin cuerpo (NO)
POST: manda datos al servidor. Con cuerpo (Si)
HEAD: obtiene solo las cabeceras de un recurso. Sin cuerpo (NO)
PUT: almacena los datos del cuerpo en el servidor. Con cuerpo (Si)
TRACE: traza el mensaje por los proxys que pasa hasta el servidor. Sin cuerpo (NO)
OPTIONS: determina los métodos soportados por el servidor. Sin cuerpo (NO)
DELETE: elimina un recurso del servidor. Sin cuerpo (NO).

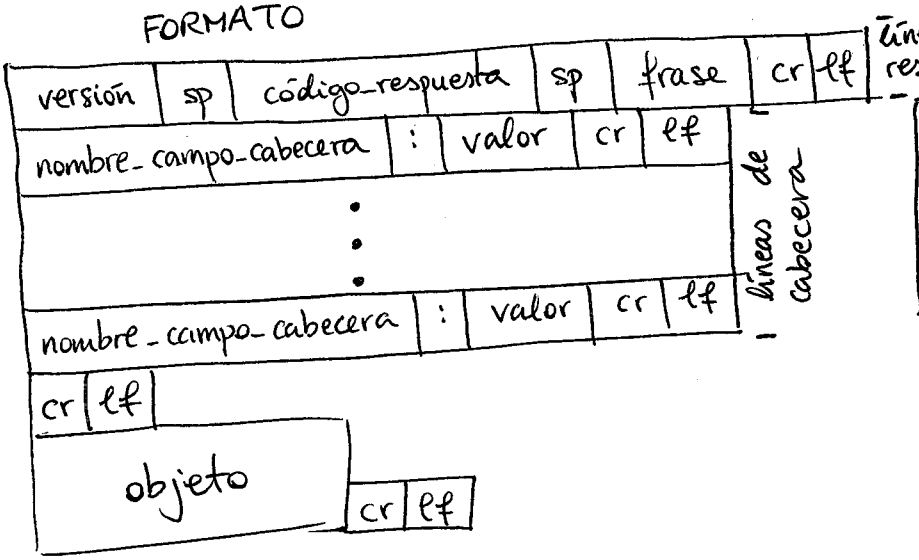
▶ PETICIÓN GET:
- Método más común
- Puede codificar parámetros también en la URL.
- Más inseguro que POST, porque los datos sensibles como contraseñas son visibles en los logs del servidor Web.

▶ PETICIÓN HEAD:
- Igual que GET, pero el servidor solo devuelve las cabeceras.
- Útil para determinar si un recurso existe, ha sido modificado, o su tipo y longitud sin descargarlo.

▶ PETICIÓN POST:
- Típicamente usado para el envío de datos de formularios Web.
- Los datos se codifican en el cuerpo, y suelen ser procesados por un script en el servidor.

RESPUESTA HTTP

FORMATO



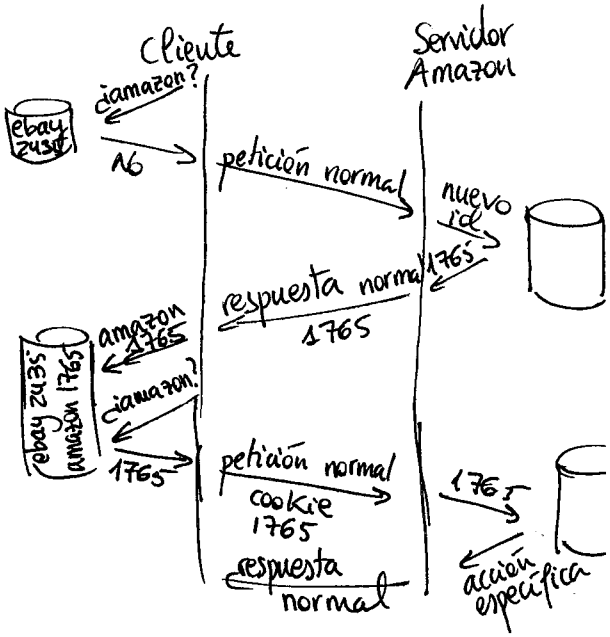
CÓDIGOS DE RESPUESTA

RangoCompleto	RangoDefinido	Categoría
100 – 199	100-101	información
200 – 299	200 – 206	éxito
300 – 399	300 – 305	redirección
400 – 499	400 – 415	error_cliente
500 – 599	500 – 505	error_server

COOKIES

COMPONENTES

1. Línea encabezado cookie en respuesta HTTP.
2. Línea encabezado cookie en petición HTTP.
3. Archivo cookie almacenado en la máquina del usuario y administrada por su navegador.
4. Base de datos en sitio Web.



INFORMACIÓN TRANSPORTADA

- autorización
- shopping carts
- sugerencias
- estado de la sesión del usuario

PRIVACIDAD

- las cookies permiten que el sitio aprenda mucho sobre el usuario
- se puede proveer nombre y correo al sitio
- los motores de búsqueda usan redirecciones y cookies para aprender aún más
- las compañías de avisos obtienen información de los sitios w

SERVIDORES PROXY

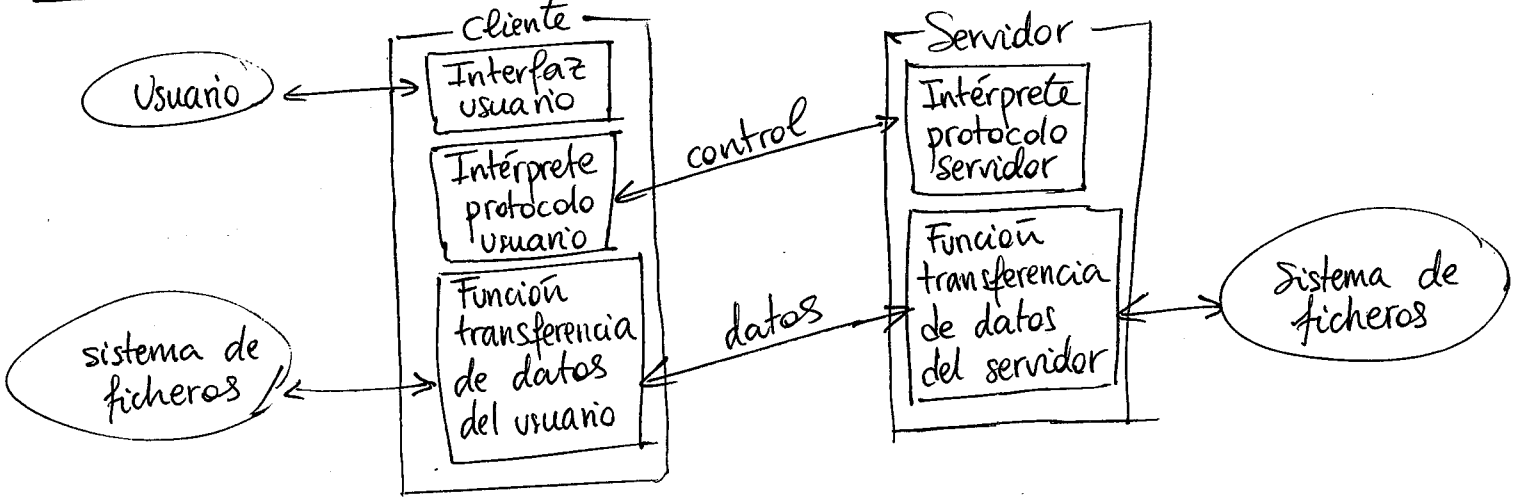
- Caché, que actúa tanto de cliente como de servidor
- Típicamente el caché está instalado en el ISP (compañía, universidad...)
- Reduce tiempo de respuesta de las peticiones del cliente.
- Reduce tráfico de los enlaces Internet de la institución.
- Con cachés la información en Internet es densa y permite a proveedores "pobres" ser eficientes.

PARTE 4 – FTP (FILE TRANSFER PROTOCOL)

CARACTERÍSTICAS

- FTP transfiere ficheros entre dos sistemas (diferente de acceso transparente de ficheros NFS).
- Fue diseñado para trabajar entre diferentes
  - sistemas operativos
  - estructuras de ficheros
  - representación de texto y datos en ficheros.
  - restricciones de acceso a archivos
  - reglas para recorrer los direct
- Servicio público usando como user anonymous.
- Servicio privado usando user y passwd.

ARQUITECTURA



DESCRIPCIÓN

- El cliente está conectado con el servidor mediante una conexión de control
- El cliente envía comandos que son contestados por el servidor. Esta conexión utiliza el protocolo NVT de telnet. Estos comandos permiten cambio crear, renombrar y borrar archivos/directorios.
- Si el cliente pide un directorio/transferencia de archivo, se abre una nueva conexión de datos por la que se envía. Finalizada la transferencia, la conexión de datos se cierra.

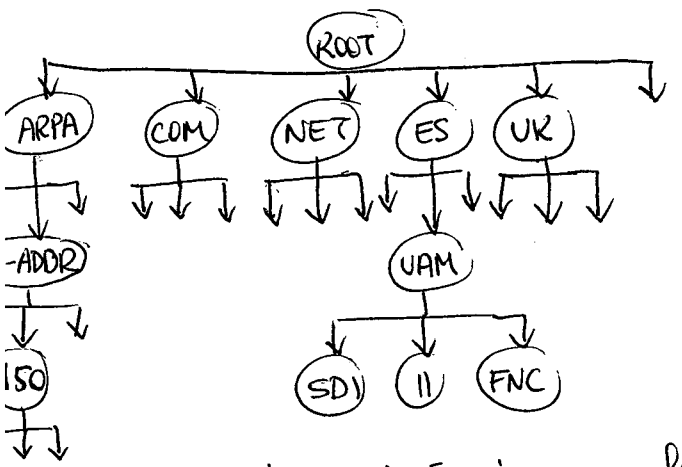
(*) MÁS INFO PROTOCOLO DIAPOSITIVAS

PARTE 5 - DNS (DOMAIN NAME SYSTEM)

CARACTERÍSTICAS DNS

- Utilizado fundamentalmente por aplicaciones TCP-IP para obtener la dirección IP a partir del nombre del host.
- La aplicación cliente encargada de la resolución de nombres se denomina RESOLVER. Se implementa mediante las funciones de librería gethostbyname y gethostbyaddress.
- El resolver obtiene la información:
  - Consultando un fichero host en la máquina.
  - Contactando con un servidor de nombres.
- Este servicio opera tanto sobre UDP como TCP por el puerto 53.
- Una organización debe disponer de un Servidor local de Nombres donde se encuentra la base de datos con los nombres de las máquinas y sus direcciones IP.
- Cualquier consulta desde un cliente de la organización sobre una máquina interna será resuelta en el servidor local.
- Cualquier consulta desde un cliente de la organización sobre una máquina externa será resuelta por el servidor local que contactará con el servidor del host externo
- El servidor antes de contactar con un servidor externo, consulta si tiene dicha información en la caché.

ESTRUCTURA DEL DNS



RAÍZ

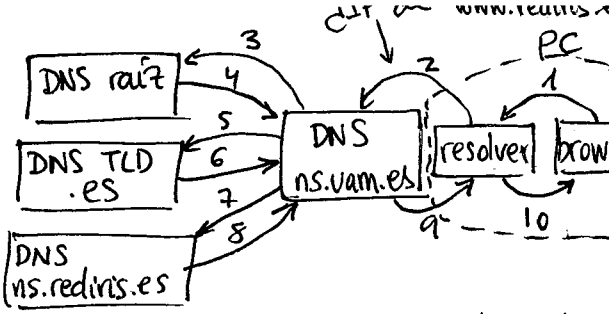
GENÉRICOS/ /GEOGRÁFICOS

TLD's: Dominios de primer nivel
- Dominios genéricos: gTLD
  .com, .org, .net
- Dominios geográficos: ccTLD
  .es, .uk, .pt, .fr

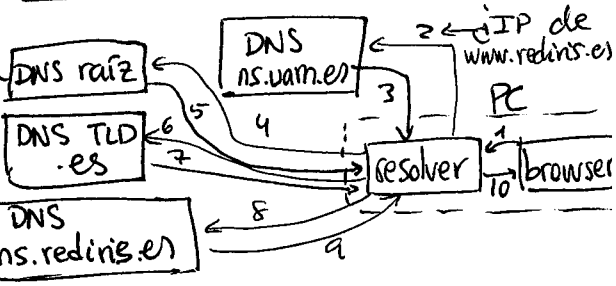
AUTORIDAD Y DELEGACIÓN:
- gTLD: ICANN
- ccTLD: por los países

- Estructura jerárquica en forma de árbol
- Cada nodo tiene una etiqueta
- El nombre del dominio se forma añadiendo las etiquetas separadas por un punto, desde un nodo terminal hasta el root. Ej: ii.uam.es

TIPOS DE CONSULTAS DNS



▶ RECURSIVA: el servidor de nombres dará la respuesta completa, haciendo todas las sub-consultas que sean necesarias (los servidores no tienen la obligación de soportar este tipo)



▶ ITERATIVA: el servidor de nombres dará una respuesta parcial, y es responsabilidad del cliente "seguir preguntando".

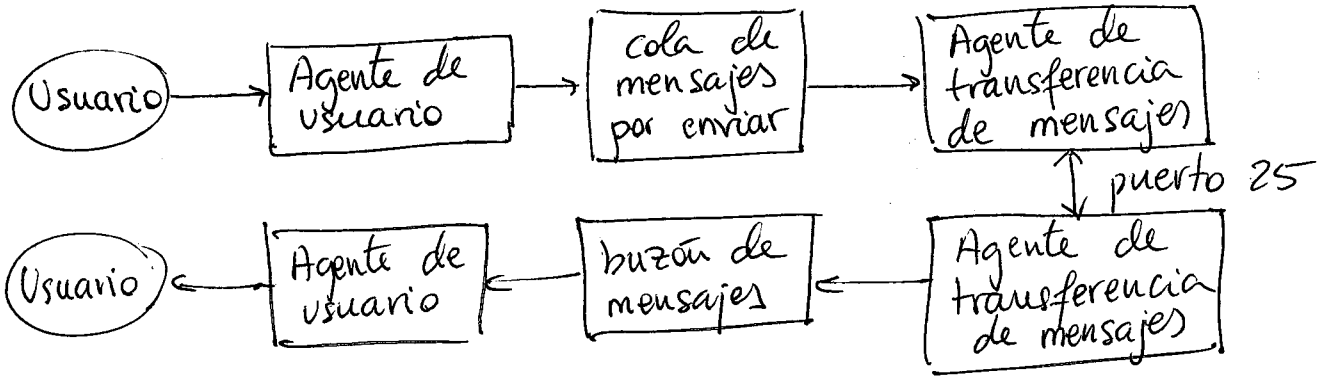
▶ INVERSA: responden ¿a qué dominio corresponde esta IP? Se resuelven iterativamente o recursivamente

⊛ Resolver: servicio o librería instalada en el ordenador del usuario que se encarga de traducir y enviar consultas y nombres de los programas de usuarios.

PARTE 6 – SMTP, POP3, IMAP (mail transfer protocols)

SMTP

- Usa TCP para transferir mensajes email desde el cliente al servidor mediante el puerto 25.
- Transferencia directa: servidor envía correos al servidor receptor
- Tres fases de transferencia:
  1. Handshaking
  2. Transferencia de mensajes
  3. Cierre



PROTOCOLOS DE RECEPCIÓN DE CORREO ELECTR.

▶ POP3: Post Office Protocol

Fases → 1. Autorización
→ 2. Transacción
→ 3. Actualización

Puerto 110

▶ IMAP: Internet Mail Access Protocol

Permite la manipulación de mensajes almacenados en el servidor

Puerto 143

PARTE 7 – DHCP (Dynamic Host Configuration Protocol)

CARACTERÍSTICAS

Información suministrada por DHCP para configurar TCP-IP:

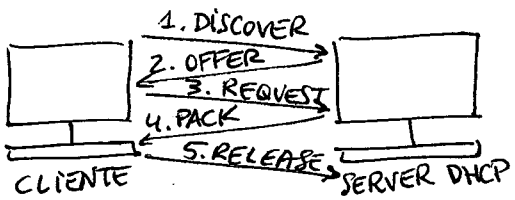
- dirección IP
- Máscara subred
- Router por defecto

- Ventajas
  - Facilita la administración
  - Reduce nº total de IP's.

• UDP puerto 67

• Desventajas
  - Información DNS no actualizada
    Hay que usar Dynamic DNS (DDNS)

• Mecanismos para asignar direcciones:
  - Automática: la dirección IP asignada es permanente para el cliente.
  - Dinámica: la IP está asignada por un periodo de tiempo.
  - Manual: la dirección IP se configura manualmente en el cliente.



1. Cliente manda en difusión mensaje DISCOVER (IP origen: 0.0.0.0, IP Dest: 255.255.255.25

2. Servidor puede responder con un mensaje OFFER que incluye dir. IP propuesta y otras opciones. Se envía con MAC del cliente y IP: 255.255.255.255.

3. Cliente selecciona IP y manda en difusión un mensaje REQUEST.

4. Servidor manda PACK, indicando el tiempo de validez de la IP.

5. El cliente puede soltar la IP antes de ese tiempo con un mensaje RELEASE.

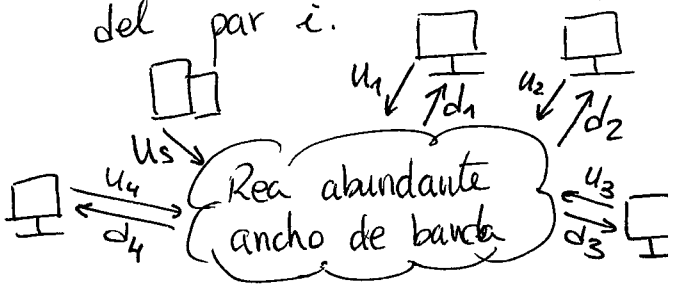
TEMA 8 – APLICACIONES P2P

CLIENTE-SERVIDOR VS P2P

$$d_{cs} = \max\left(\frac{NF}{u_s}, \frac{F}{\min(d_i)}\right)$$

$$d_{P2P} = \max\left(\frac{F}{u_s}, \frac{F}{\min(d_i)}, \frac{NF}{u_s + \sum_{i=1}^{n} u_i}\right)$$

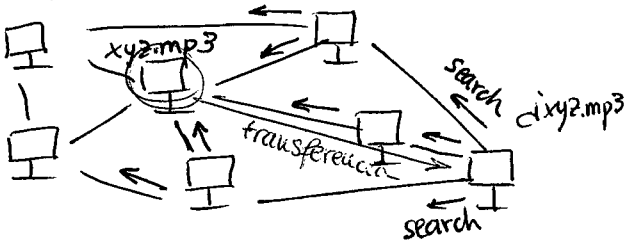
u_s := ancho de banda de subida del servidor
u_i := ancho de banda de subida del par i.
d_i := ancho de banda de bajada del par i.



Funcionamiento BitTorrent: diapositivas (idea: toma y daca)

DHT (Distributed Hash Table)

- Es la base de datos distribuida de P2P
- La base de datos contiene pares (clave, valor) { clave: id usuario; valor nombr / valor: id contenido, valor dir. I
- Los pares consultan la base de datos con la clave, y esta responde con el valor correspondiente.
- Los pares (peers) sdo pueden insertar pares (clave, valor) en la BD.



- Basados en FLOODING: Napster, eMule
  - La búsqueda es C-S
  - La transferencia es P2P

- Protocolo CHORD
  - Los nodos se distribuyen en anillo
  - Tanto nodos como claves tienen un ID en el rango $[0, 2^n - 1]$
    - nodeid = SHA1 (Dir. IP, i)
    - keyid = SHA1 (nombre clave)
  - Cada clave K se almacena en el sucesor del nodo cuyo nodeid es igual o mayor a keyid.