Contents

1

Introducción

- Breve historia de las redes de computadores e internet
- Introducción a Internet
- Standardization Organizations and OSI Reference Model
- Paradigma cliente-servidor

1.1 Historia de las redes e Internet

Historia de las redes

- 1830: telegrafo
- 1866 Primer **cable transatlántico** para telegrafo
- 1875: Graham Bell inventa el **telefono**
- 1951: primer ordenador comercial
- 1960: concepto de **Packet Switching**
- 1960s: proyecto **ARPANET**, origen de Internet
- 1972: Primera red internacional y comercial de Packet Switching, X.25
- 1990s: Internet es abierto al público general

Historia de Internet

- 1966: Defense Advanced Research Projects Agency (DARPA). **ARPANET**
- ARPANET conecta universidades, centros de investigación y centros militares. La parte militar se separa en 1983.

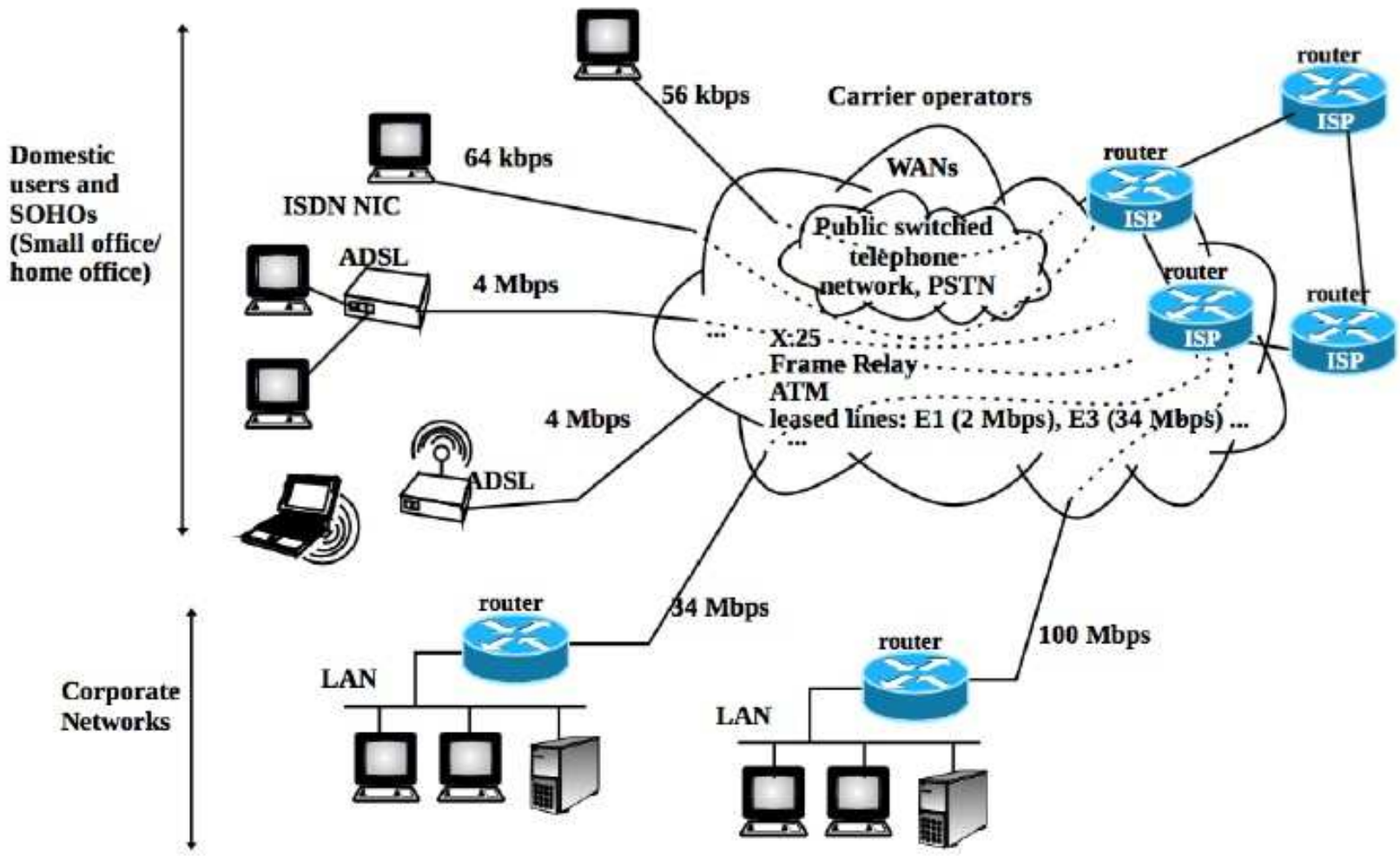
- 1970s: End-to-end reliability was moved to hosts, developing TCP/IP. TCP/IP was ported to UNIX Berkeley distribution, BSD.

- 1990s: Internet es abierto al comercio y el público general por parte de los ISP (Internet Service Providers).

1.2 Introducción a Internet

Organización de Internet y terminología

- Host
- Access Network
- LAN
- WAN
- Telephone company, telco, or carrier.
- Router
- Line Bitrate
- Bits per second, bps.



1.2.1 Bitrate

t_b es el tiempo de transmisión de 1 bit.

Vt = $^1/t_b$

Bitrates típicos:

- k, kilo 10^3
- M, Mega: 10^6
- Giga 10^9
- Tera 10^12
- Peta, 10^15

Ejemplos:

- Public Switched Telephone Network (PSTN) modem: 56 kbps

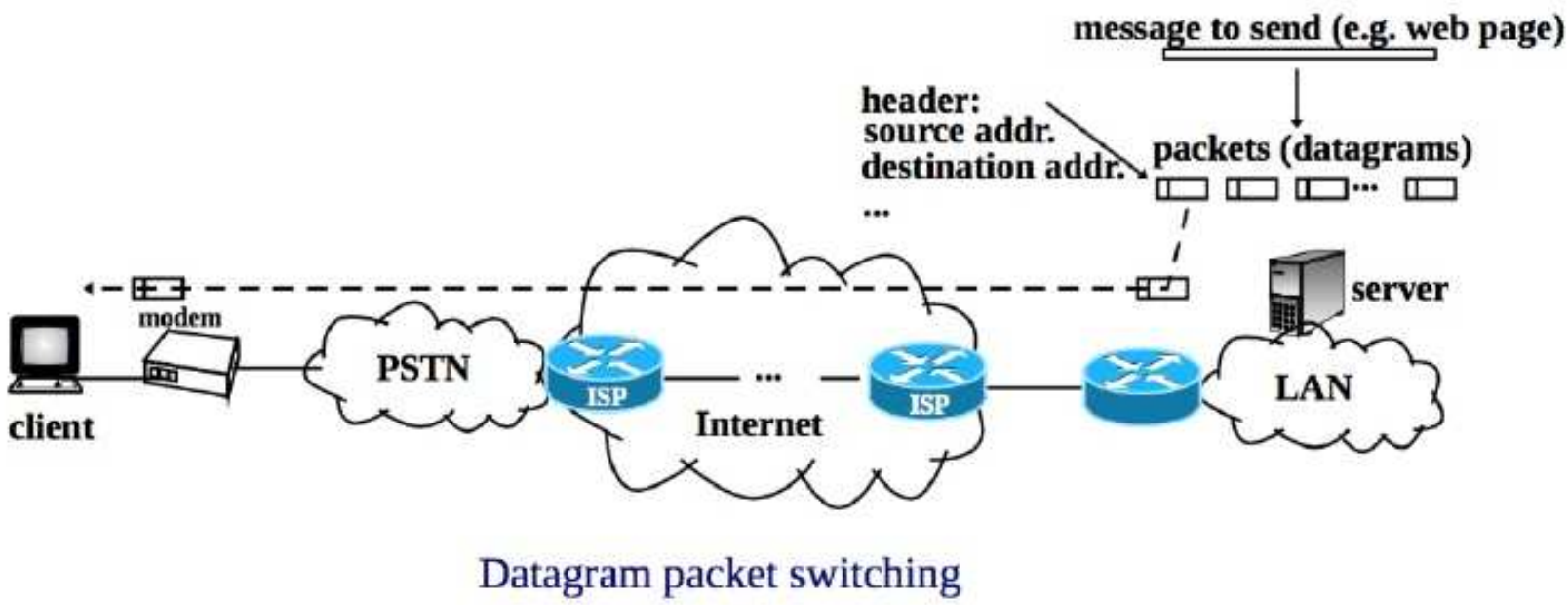
1.2.2 Types of switching

- **Circuit switching**: (Ej: PSTN) require dedicated point-to-point connections during calls.
- **Packet switching**: move data in separate, small blocks – packets – based on the destination address in each packet. When received, packets are reassembled in the proper sequence to make up the message.
  - Virtual circuit: X.25, ATM
  - Datagram: Internet

1.2.3 Parámetros de transmision

WAN: Wide Area Network - Xarxa de gran abast LAN: Local Area Network - Xarxes locals

- Distancia: d metros
- Velocidad de transmisión de la luz: c = 3 $\times 10^8$ m/s
- Retraso de transmisión t_p = d/c segundos
- Retraso end-to-end: D segundos
- Tamaño del paquete: L bits
- Bitrate de transmisión: V_t b/s (bps)



Datagram packet switching

- Tiempo de transmisión de paquete: $t_packet = L/V_t$ seg
- Bandwidht-Delay (retraso del ancho de banda) product $= V_t \times D$ bits

Ejemplo

d = 10km

$t_p = \frac{1}{c} = \frac{10*10^3}{3*10^8} = \frac{1}{3} \times 10^-4$ secs

$V_t = 10\text{Mbps} = 10 \times 10^6$ bps

$t_bt = 0.1\ \mu s$

1.2.4 Switching Paradignms

Circuit switching

- Recursos reservados end-to-end
- Bitrate fijo end-to-end (e2e)
- Flow of bits
- Connection setup
- Fixed end-to-end delay

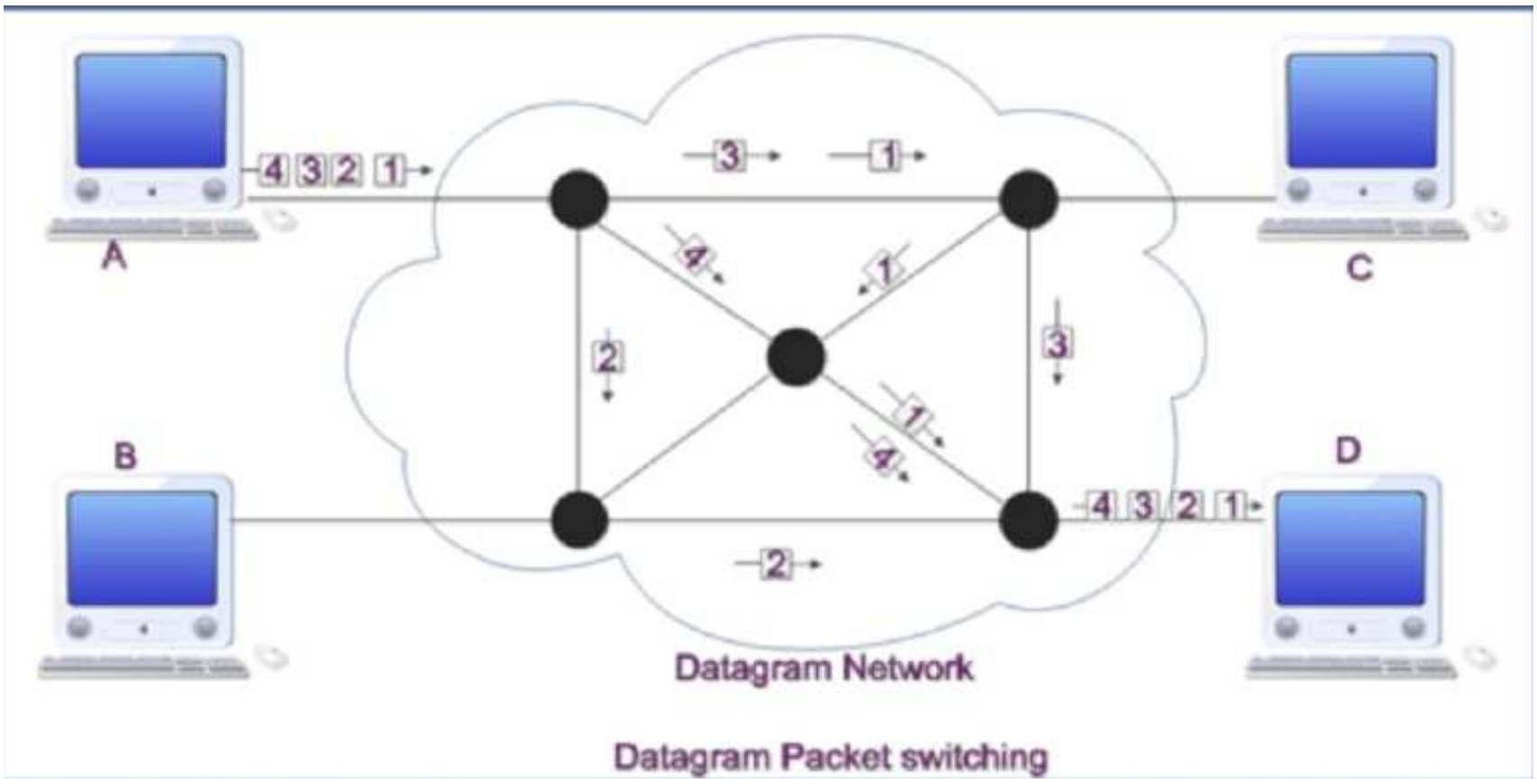
Packet switching

- Resources used on-the-fly
- At link bitrates

- Flow of packets
- Sharing of resources
- Variable e2e delay

1.2.5 Concepto de datagrama

A self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network.



1.3 TCP/IP Protocol Architecture

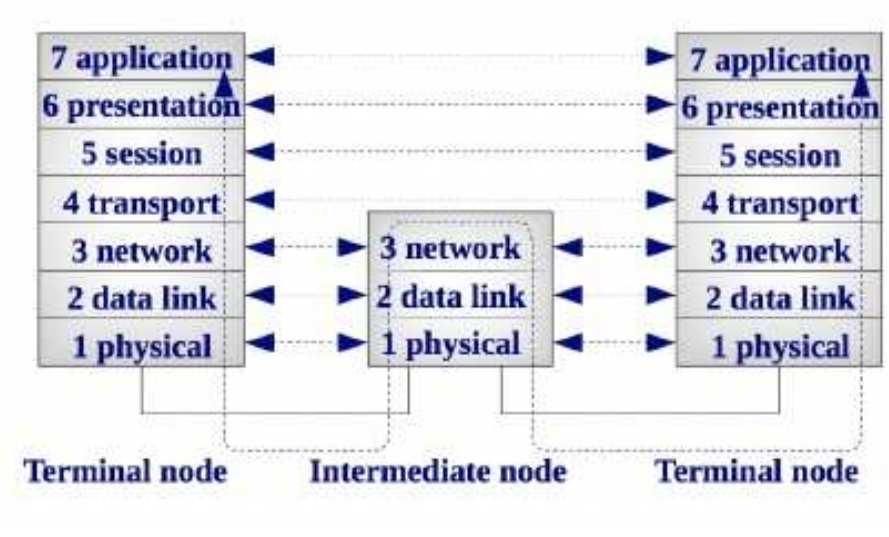
- Focus in inter-networking
- Net layer: datagram transport: IP over everything

1.4 OSI Layers

- Physical
  - Mecanico
  - Electrico
  - Funcional

- Data Link
- Network/Red
  - Transporte de información
  - Hace que a los niveles superiores les es indiferente el nivel de debajo
- Transporte
  - Intercambio de datos entre sitemas
  - Fiabilidad
  - Calidad
- Sesion
  - Como debe ser el dialogo entre aplicaciones
  - Disciplina de dialogo
  - Grouping
  - Recovery
- Presentación: Hace de traductor, adapta la información
  - Formato de datos y codificación
  - Encriptación
- Aplicación
  - Means for applications to acces OSI environment

De Transporte para arriba es de punto a punto (e2e), para abajo hay nodos intermedios (PE: Nivel 3 se entiende con router, nivel 2 con un conmutador...)



7. Application: Processes using network services (web, email...)
6. Presentation: Encoding of text, numbers...
5. Session: "Login" type service.
4. Transport: End to end data transfer.
3. Network: Routing.
2. Data link: Structured transport of bits.
1. Physical: Electric and mechanical.

*Internet jargon: Layer 8: the user.

1.4.1 The OSI ENvironment

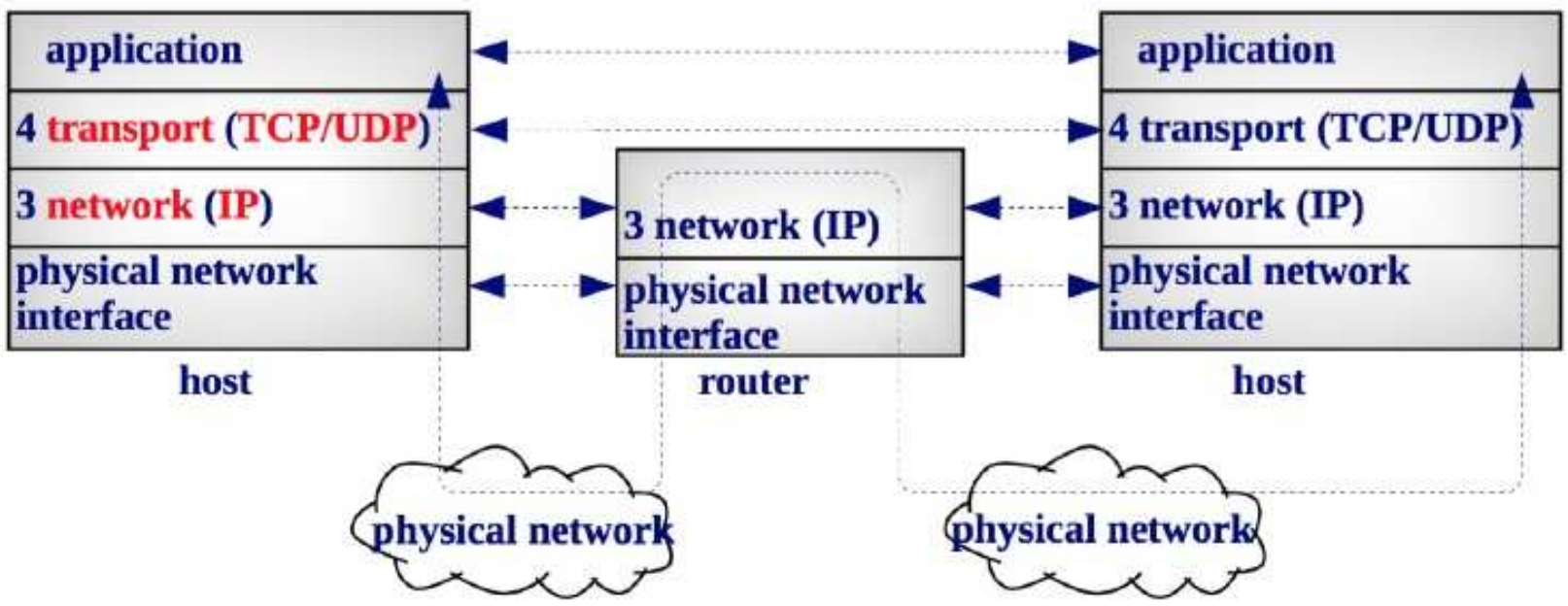
1.4.2 OSI vs TCP/IP

1.5 PROTOCOL DATA UNIT PDU. SERVICE ACCES POINT SAP

1.6 Use of a relay

1.7 TCP/IP

TCP: Transport Control Protocol



El código de red de TCP/IP es parte del Kernel del SO.

Socket interface: es la interfaz de conexión de UNIX para procesos. Implementado primero en BSD.

La syscall socket crea un socket descriptor utilizado para almacenar toda la información asociada con una conexión de red, similar a como un inodo es descriptor de un archivo.

1.8 TCP

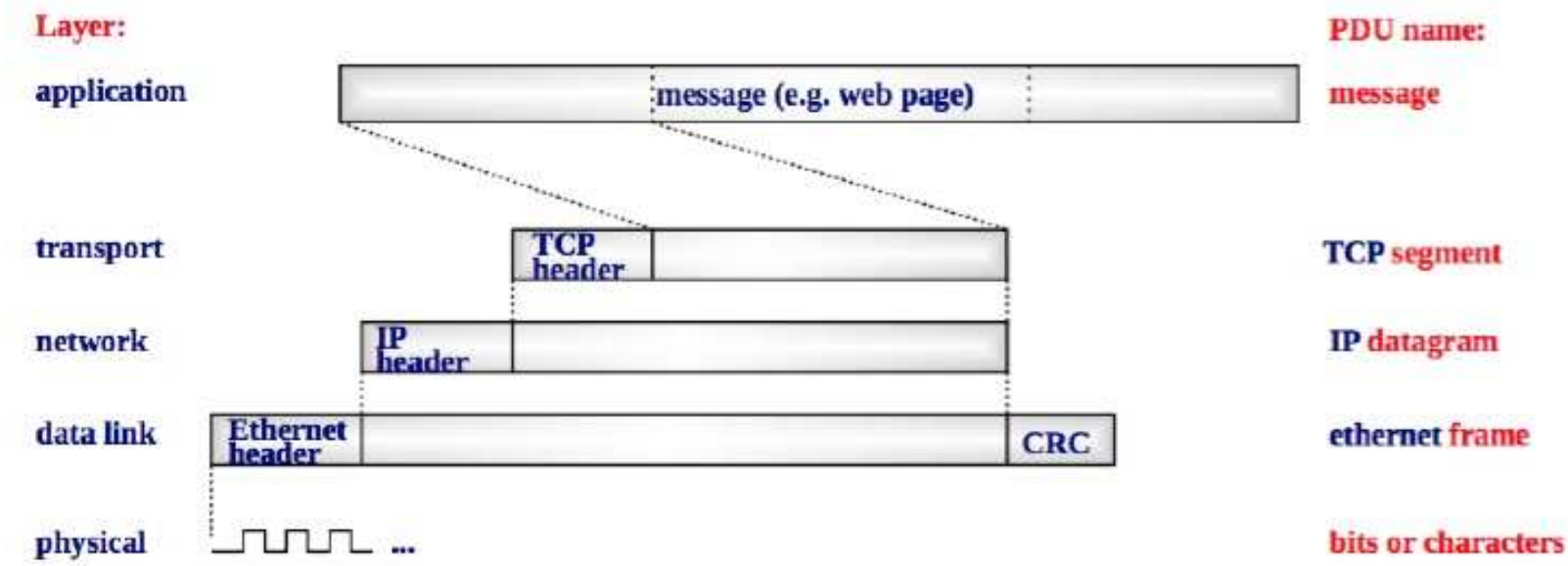
- Usual transport layer is Transmission Control Protocol
  - Reliable connection
- Connection
  - Asociación lógica temporal entre entidades de diferentes sistemas.
- TCP PDU

  - Called TCP segment
  - Incluye el puerto de origen y destino
    * Identifica a los usuarios respectivos
    * La conexión va vinculada a un par de puertos

- TCP tracks segments between entities on each connection

1.8.1 Encapsulación

- Each layer adds/remove the PDU header.



1.9 Client Server Paradigm: Processes, messages, sockets, segments and IP datagrams

1.9.1 The Internet Transport Layer

- Se utilizan 2 protocolos en la capa de transporte TCP/IP
  - UDP: User Datagram Protocol
    * Ofrece un servicio de datagrama (not reliable). It is connectionless
  - TCP: Transmission Control Protocol
    * Offers a reliable service (correct segments are acknowledged, ack, lost segments are retransmitted). It is connection oriented

1.9.2 Client Server Paradigm

Como se inicia la conexión entre procesos? El cliente siempre inicia la conexción hacia una dirección IP conocida, en el header de IP, y a un puerto conocido (¡1024), en el header TCP/UDP.

Well known ports: estan estandarizados por IANA en RFC-1700 (números asignados). En UNIX se encuentran en /etc/services

- ftp: 20
- telnet: 23
- ssh: 22
- chargen: 19
- http: 80
- smtp: 25, smtps: 465, 567
- ntp: 123
- dns: 53
- pop3: 110, imap:143, pop3s: 995, imaps ???

El servidor es un **daemon** esperando peticiones de clientes.

2

Tema 2: Redes IP (Internet Protocol)

El nivel IP es el nivel 3 de las redes.

IP Layer Service

- El objetivo de IP es el direccionamiento de datagramas (routing datagrams)
- El objetivo principal de su diseño era interconectar hosts adjuntos a redes LANs/WANs de diferentes tecnologías.
- Sus características son:
  - Conectionless
  - Stateless
- Best effort

2.1 Cabecera IP - RFC 791

Componentes:

- Version
- IHL: IP Header Lenght. Tamaño de la cabecera en palabras de 32 bits, si no tiene opciones vale 5, es decir 20 bytes (5*4bytes(32bits)).
- Tipo de servicio: preferencia de encaminamiento. El formato es: **xxxdtrc0**, donde xxx permite indicar una precedència.
  - Solo tiene sentido dentro de una misma red y no en Internet
  - El último bit se pone a 0 (no se utiliza)
  - Los bits dtrc quieren decir:

    - d: delay, optimizar el retraso
    - t: throughput: optimizar velocidad eficaz
    - r: realiability: optimizar la fiabilidad
    - c: cost: optimizar el coste

- Total lenght: tamaño total del datagrama en bytes
- Identification, Flags, Fragment Offset: se utiliza en la fragmentación
- Time to live: tiempo de vida. Lo routers decrementan este campo y descartan el datagrama si llega a 0.
- Protocol: multiplexación del protocolo de nivel superior. Los números de protocolo estan estandarizados.
- Header checksum: permite la detección de errores. El checksum lo es solo de la cabecera. El algoritmo de checksum es el complemento a 1 de la suma en complemento a 1 de la información a proteger.
- Source Address, Destination Address
- Options:
    - Record route: los routers añaden la dirección IP de la interfície donde encaminan el datagrama.
    - Loose Source Routing: especifica una lista de direcciones IP de routers que tiene que atravesar el datagrama
    - Strict Source Routing: Direcciones IP de los únicos routers que puede atravesar el datagrama.

2.1.1 Fragmentación

IP puede fragmentar un datagrama cuando la MTU (Maximum Transfer Unit) del nivel de enlace dónde debe encaminarse tiene un tamaño menor que la del datagrama. Pasa en los siguientes casos:

- Cuando un router ha de encaminar un datagrama por una red con MTU menor que la de donde ha llegado
- Cuando se utiliza UDP y la aplicación hace una escritura mayor que la MTU de la red.

El reensamblado de los datagramas se hace en la destinación. Se utilizan los campos:

- Identification: el nivel IP del gost que genera los datagramas pone el valor de un contador que incrementa cada vez que se genera un nuevo datagrama. Este número identifica fragmentos de un mismo datagrama.

- Flags: son 0DM. El primer bit no se utiliza.
  - D: flag de don't fragment. Cuando esta a 1 el datagrama no se puede fragmentar.
  - M: flag de more fragments. A 1 quiere decir que hay más fragmentos.
- Offset: posición del primer byte del fragmento del datagrama original (el primero vale '0'). Se cuenta en unidades de 8 bytes.

El tamaño del payload ha de ser un multiple de 8 bytes. Por ejemplo si un datagrama de 1500 bytes se encuentra con una MTU de 1480 se fragmentará de la siguiente forma:

- 1500 bytes = 1480 bytes de payload y 20 de cabecera
  - 1476 bytes fragmento 1. 1456 (el multiplo de 8 más cercano a 1480) de payload y 20 de cabecera
  - 44 bytes fragmento 2. Faltan 24 bytes de payload y una cabecera de 20 bytes.

2.1.2 MTU Path Discovery

El nivel de transporte TCP agrupa los bytes que escribe la aplicación hasta tener un segmento de tamaño óptimo y después lo envia.

- Normalmente el tamaño óptimo es el que permite enviar los segmentos de tamaño igual a la MTU menor de las redes que tiene que atravesar.

Esto permite enviar segmentos de tamaño grande, pero no tanto como para que se tengan que fragmentar.

La fragmentación no es deseable porque:

- Realentiza los routers
- Puede provocar que haya fragmentos de tamaño pequeño que reduzcan la eficiencia de la red
- Si se pierde 1 solo fragmente se tienen que descartar los demás cuando lleguen al destino

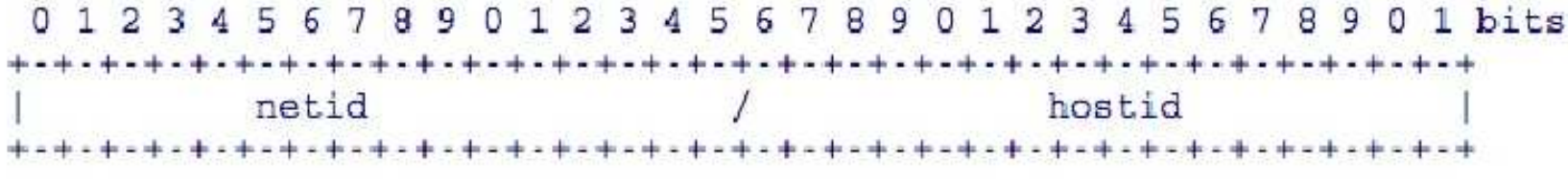
Para conseguir que el tamaño sea óptimo primero se adapta el tamaño de los segmentos a la MTU de la red dónde esta conectado el host.

2.2 Direcciones IP

En un principio se decidió que 32 bits eran suficientes para poder identificar a todos los dispositivos que se conectaban a internet. IPV6 utiliza direcciones de 128 bits

- 32 bits (4 bytes)
- Dotted point notation: 4 bits en decimal. Ej: 147.83.24.28

- Netid identifica la red
- Hostid identifica el host de la red
- Una dirección IP identifica una interfaz: un nodo conectado a la red.
- Todas las direcciones IP en internet deben ser diferentes. IANA asigna bloques de direcciones a RIR (Regional Internet Registries)
- RIR asigna direcciones a los ISPs
- Los ISPs asignan direcciones a sus clientes



2.2.1 Como se organizaron las direcciones? IP Adresses - Classes

- Los bits más grandes identifican la clase
- El número de bits IP de netid/hostid varia en las clases A/B/C
- Clase D: es para direcciones multicas (Ej: 224.0.0.2: "All routers"). Estas direcciones no identifican un hosts, si no un canal, por ejemplo de televisión. Tienen 28 bits para dirección.
- Clase E: son direcciones reservadas (experimentales).

Classe	netid (bytes)	hostid (bytes)	Codification	range
A	1	3	0xxxx···x	0.0.0.0 ~ 127.255.255.255
B	2	2	10xxx···x	128.0.0.0 ~ 191.255.255.255
C	3	1	110xx···x	192.0.0.0 ~ 223.255.255.255
D	-	-	1110x···x	224.0.0.0 ~ 239.255.255.255
E	-	-	1111x···x	240.0.0.0 ~ 255.255.255.255

La UPC tiene una direccion tipo 147.83.X.X. Es clase B. Toda la universidad tiene disponibles 2^{16} direcciones. A Stanford les dieron una clase A. Al principio se daban redes muy grandes. Más adelante se redefinieron las reglas para dar direcciones.

2.2.2 Direcciones especiales

Las direcciones especiales no pueden ser utilizadas en interfaces físicas.

Casa red tiene 2 direcciones especiales:

- Network address

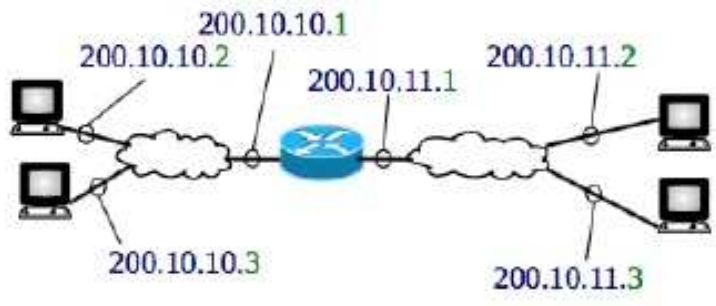
- Broadcast address

Explicación de la tabla

- Si todos los bits del host son 0, se utiliza para identificar la red.
- Si todos son 1, identifica todas las máquinas de la red. Es El broadcast de la red
- Todos los netid y host id 0: no podemos identificar.
- Todos los bits de netid y host id 1: todos los hosts de todas las redes
  - Enviar un datagrama a 255.255.255.255 sería enviar un datagrama a todas las máquinas del mundo, pero el router no lo permite, esta restringido.
- netid 127 y hostid xxx: se refiere a la máquina local. Host loopback. La dirección no esta enrutada.

netid	hostid	Meaning
xxx	all '0'	Identifies a network. It is used in routing tables.
xxx	all '1'	Broadcast in the net. xxx.
all '0'	all '0'	Identifies "this host" in "this net.". Used as source address in configuration protocols, e.g. DHCP.
all '1'	all '1'	broadcast in "this net.". Used as destination address in configuration protocols, e.g. DHCP.
127	xxx	host loopback: interprocess communication with TCP/IP.

- Example:



2.2.3 Direcciones privadas

Cualquier red que comience por 192.168. es privada

- La mayoria de OSs incluyen el stack TCP/IP
- TCP/IP es utilizado en gran diversidad de dispositivos electrónicos.
- Las direcciones privadas han sido reservadas para dispositivos que no utilizan direcciones publicas. Estas direcciones no estan asignadas a ningún RIR y no son únicas. Las hay en cada clase
  - 1 red clase A: 10.0.0.0
  - 16 redes classe B: 172.16.0.0 - 172.21.0.0

- 256 redes clase C: 192.168.0.0 - 192.168.255.0

Cada puerto del router gestiona una red IP diferente.

2.3 Subnetting RFC 950

Inicialmente la netid estaba determinada por la clase de la dirección:

- A con 2^{24} direcciones
- B con 2^{16} direcciones
- C con 2^8 direcciones

Subnetting permite añadir bits desde el hostid hasta el netide (llamados subnetid bits).

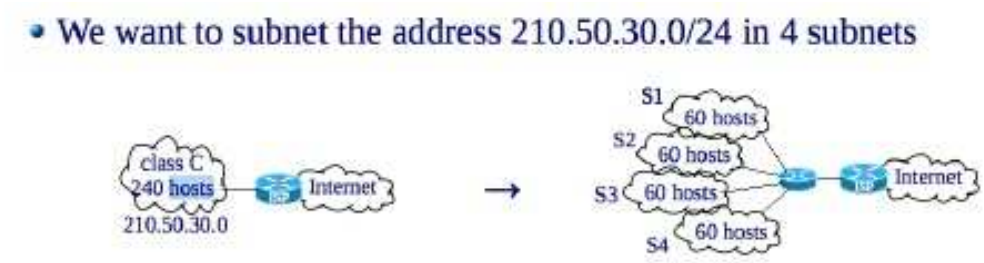
Ejemplo: Para la ISP el prefijo de red es de 24 bits. Para el router interno el prefijo es 26 bits. Los 2 bits extra permiten 4 subredes.

Una máscara es utilizada para identificar el tamaño de netid+subnetid prefix.

Notaciones de la mascara

- Puntuada, como 255.255.255.192
- Dando la longitud de la máscara (número de bits) como 210.50.30.0/26

2.3.1 Ejemplo



B = 210.50.30

subnet	subnetid	IP net. addr.	range	broadcast	available
S1	00	B.0/26	B.0 ~ B.63	B.63	$2^6 - 2 = 62$
S2	01	B.64/26	B.64 ~ B.127	B.127	$2^6 - 2 = 62$
S3	10	B.128/26	B.128 ~ B.191	B.191	$2^6 - 2 = 62$
S4	11	B.192/26	B.192 ~ B.255	B.255	$2^6 - 2 = 62$

2.3.2 Variable Lenght Subnet Mask (VLSM)

Subredes de diferentes tamaños. Por ejemplo subredear una dirección de clase C:

2.3.3 Ejemplo

- We have 1 byte for subnetid + hostid.
- subnetid is green, chosen subnets addresses are underlined.

$$\left.\begin{matrix}\underline{0000}\\ \underline{1000}\end{matrix}\right\} \rightarrow \left.\begin{matrix}\underline{1000}\\ \underline{1100}\end{matrix}\right\} \rightarrow \begin{matrix}\underline{1100}\\ \underline{1101}\\ \underline{1110}\\ \underline{1111}\end{matrix}$$

subnet	subnetid	IP net. addr.	range	broadcast	available
S1	0	B.0/25	B.0 ~ B.127	B.127	$2^7 - 2 = 126$
S2	10	B.128/26	B.128 ~ B.191	B.191	$2^6 - 2 = 62$
S3	1100	B.192/28	B.192 ~ B.207	B.207	$2^4 - 2 = 14$
S4	1101	B.208/28	B.208 ~ B.223	B.223	$2^4 - 2 = 14$
S5	1110	B.224/28	B.224 ~ B.239	B.239	$2^4 - 2 = 14$
S6	1111	B.240/28	B.240 ~ B.255	B.255	$2^4 - 2 = 14$

2.4 CIDR: Classless Inter-DOmain Routing

Inicialmente, el núcleo de las tablas de enrutamiento de internet no usaba máscaras: netid era derivado de la clase de dirección IP.

Cuando el número de conexiones en internet comenzaron a crecer exponencialmente, el tamaño de las tablas de enrutado se disparó.

Con tal de reducir el tamaño de las tablas de enrutado, CIDR propuso una distribución greagráficamente racional de direcciones IP, capaz de agregar routas y usar máscaras en lugar de clases.

Ejemplo de agregación:

$$200.1.10.0/24 \rightarrow 200.1.10.0/23$$

$$200.1.11.0/24 \rightarrow 200.1.10.0/23$$

El término sumarización se utiliza cuando la agregación se hace en el límite de clase. (ej: grupos de subredes esta sumarizado con su dirección base con clase).

2.5 Tablas de encaminamiento y algoritmo de entrega de datagramas

2.6 ARP: Address Resolution Protocol

Motivación: Direcciones de red vs. direcciones físicas.

En una LAN se utilizan direcciones para identificar la estación transmisora y receptora. Por ejemplo ethernet utiliza direcciones de 6 bytes (48 bits).

Ethernet utiliza un "medio compartido" que funciona como BUS: Cuando una estación envía una trama, esta llega a todas las estaciones conectadas a la red ethernet. En cada estación la tarjeta ethernet recibe la trama que se ha enviado al medio.

Todas las tarjetas miran la dirección Ethernet de la destinación que hay en la trama. Si la dirección es la de la tarjeta, entonces la tarjeta interrumpe la CPU y transfiere el contenido recibido por DMA a la memoria del ordenador. En caso contrario descarta.

Cuando un ordenador quiere enviar una trama Ethernet, ha de pasar **la información** (por ejemplo datagrama IP) y **la dirección Ethernet** de la tarjeta del computador donde se quiere enviar.

Después de mirar la tabla de encaminamiento, el nivel IP puede necesitar de una conversión de dirección IP a dirección física.

Dirección física se refiere a la dirección de la red física por la que se enviará el datagrama, por ejemplo Ethernet.

La conversión no es siempre necesaria. Por ejemplo en un enlace punto a punto no se necesita una dirección física, proque hay un único destinatario posible.

ARP es el mecanismo que se encarga de hacer la conversión de dirección IP → dirección física. El procedimiento que sigue es el siguiente:

- IP determina la dirección IP dónde ha de enviarse el datagrama (puede ser dirección del datagrama si es directo o la del router si es indirecto).

- Si la interfície por donde debe enviarse el datagrama no necesita dirección física (ej: enlace punto a punto), entonces IP pasa el datagrama al driver para su transmisiín.

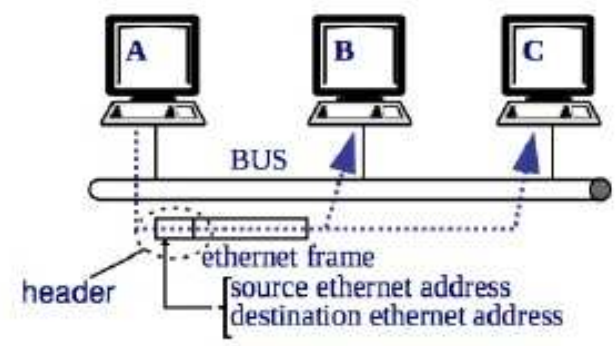
- Si no:

  – IP solicita al módulo ARP la dirección física que corresponde a la dirección IP.

  – ARP tiene una tabla donde guarda las tuplas @IP: @Física. Si la dirección IP solicitada no esta en la tabla, entonces el módulo ARP inicia **proceso de resolución**.

    * Cuando acaba el proceso, ARP retorna al módulo IP la dirección física solicitada

2.6.1 Funcionamiento del mecanismo de resolución



Suponemos que el host A quiere resolver la dirección del host B. Cuando se hace el boot de los PCs las caches ARP estan vacías.

En la imagen se ven las tramas ARP request y ARP reply que se envían durante el mecanismo de resolución. Los detalles son los siguientes:

- La estación A envia un mensaje ARP request con una dirección Ethernet destinación broadcast.
  - Esta trama lleva la @IP que se quiere resolver (la de la estación B). Como que la @destinación es broadcast, todas las tarjetas interrumpirán la CPU cuando reciban la trama. Después el driver pasará el contenido de la trama al módulo ARP.
- El módulo ARP de la estación B verá que se quiere resolver su dirección y enviará una trama **ARP reply**. La @destinación de la ARP Reply es unicast (es la dirección Ehternet de A). Los otros módulos de las estaciones descartarán el ARP request.
- Las estaciones implicadas (A y B) actualizarán la caché ARP con las direcciones IP_B; eth_B y IP_A; eth_A respectivamente.
  - El motivo de que solo las estaciones implicadas actualicen la cache ARP es que **conviene mantener las caches pequeñas** para que sean lo más rápido posible.

Las entradas de la caché ARP tiene asociadas un **time-out**. Cada vez que una entrada se utiliza, el time-out se refresca. Si salta el time-out, la entrada se borra.

2.6.2 Formato de los mensajes ARP

2.6.3 Proxy ARP

2.6.4 Gratuitous ARP

2.7 ICMP: Internet Control Message Protocol

Señaliza mensajes de error o atención.

- Va sobre IP
- Los puede activar IP/TCP/UDP o un proceso de usuario
- Puede ser **query** o **error**
- Los mensajes ICMP de error no pueden generar otro ICMP de error (evitar bucles).

2.8 DHCP: Dynamic Host Configuration Protocol

La asignación de IP puede ser:

- Manual: ifconfig eth0...
- Automática: a través del protocolo ICPC en un enlace ppp o de DHCP en una LAN.

2.9 DNS - Protocolo

2.10 Firewall

Es genéricamente un dispositivo que separa una red que se desea proteger, donde hay los posibles intrusos (resto de internet).

Hay diferentes tipos:

- Sencillo que solo filtran según la dirección IP
- Más complejos que son capaces de seguir y filtrar según el estado de las conexiones y del tipo de mensajes del nivel de aplicación.

DMZ: dónde hay los únicos hosts que se dea que sean accesibles desde el exterior.

Normalmente el firewall restringe el acceso desde el exterior a DMZ, y solo a algunos puertos de los servidores.

- En la red interior se utilizan direcciones privadas.
  - El firewall utiliza NAT para que los host de la red interior puedan acceder al exterior
- El firewall filtra los paquetes que vienen del exterior y no cumplen ciertas condiciones. El filtrado se hace con las listas de control de acceso (ACL)
  - Las ACL se aplican a la entrada o salida de una interfície.
  - Si hay definida una ACL, todos los datagramas se comparan con las reglas de la ACL.
    - Cuando se cumple una regla, el datagrama se acepta o descarta.
  - Las reglas de una ACL se leen en sequencia y se dejan de mirar cuando se cumple una de las reglas.

3

Protocolos P2P. El protocolo TCP

TCP (Transmission Control Protocol): es uno de los algoritmos con más complejidad que se utilizan en Internet.

Antes de ver TCP se estudiaran algunos algoritmos relacionados con TCP, los algoritmos AQR (Automatic Repeat reQuest). Los algoritmos ARQ los podemos encontrar en protocolos que estan distribuidos entre dos unidades, y de aquí su nombre P2P. Este nombre se hace servir en contraposicón a los otros algoritmos distribuidos en más de un punto, como ahora los algorítmos multiacceso que se utilizan en las LANs.

3.1 Protocolos ARQ básicos

El **objetivo** de los protocolos ARQ es conseguir un canal de comunicación fiable, es decir, que la información transmitida llegue

- Sin errores
- Sin duplicaciones
- En el mismo orden que se envia

En cada uno de los niveles que implementan el protocolo hay un emisor y un receptor.

El principio de funcionamiento de los protocolos ARQ se basa en la retransmisión de la información que no llega o que llega con errores al receptor.

El receptor envía señal **ACK** para que el emisor sepa si la información ha llegado correctamente.

Otros elementos que forman parte de los algorítmos ARQ son los siguientes:

- **Números de sequencia**: hacen falta para poder relacionar los mensajes de información.
- **Protocolo orientado a la conexión**: se dice que un protocolo es orientado a la conexión cuando tiene una **fase incial de establecimiento de conexión** y una **fase de terminación**.
  - En estas fases se envian mensajes de señalización que indican

Se estudiaran 3 algoritmos ARQ:

- Stop and wait
- Go back N
- Retransmisión selectiva

El nivel de implementación de ARQ no tiene porque ser el de transporte (como el caso de TCP). En la práctica podemos encontrar protocolos de otros niveles, que implementan un algoritmo ARQ.

3.2 Stop and wait

Su principio de funcionamiento es: "Transmitir una PDU de información y esperar a que se confirme antes de transmitir una nueva".

Hay diferentes formas de implementar este principio. Nosotros supondremos el algoritmo que se detalla a continuación.

Eventos:

- El nivel superior escribe la información que se ha de transmitir. En un tiempo de proceso
- La transmisión comienza inmediatamente y dura un tiempo t_t[segundos] = L_t[bits]/V_t[bps].
- El tiempo de propagación de cad abit que se transmite dura t_p[segundos] = D[m]/V_p[m/s].
- Cuando llega al último bit de I_k al secundario, el nivel superior lee la información recibida en un "tiempo de proceso" que supondremos igual a 0 y el secundario envia la confirmación A_k.
  - El tamaño de la PDU con las confirmaciones es mucho más pequeña que la de las PDUs de información.

Retransmisiones con stop and wait

En caso de error supondremos el diagrama de tiempo de la figura anterior.

Cada vez que el emisor envia una PDU, activa un temporizador T_0. Si el temporizador salta sin haber recibido la confirmación, entonces el emisor retransmite la PDU.

De esta forma si la PDU I_k se pierde, o llega con errores, se reenvia.

Necesidad de los números de sequencia

Los protocolos ARQ necesitan un número de sequencia para poder relacionar las PDUs de información y las confirmaciones.

- En la práctica, es uno de los campos de la cabecera que añade el protocolo

PDU de información: Si emisor no recibe ACK antes de time-out y reenvia, el receptor tendria 2 PDUs que no sabría que son la misma.

PDUs de confirmación: Si el tiempo de proceso del receptor no es 0. Suponemos que es anormalmente grande y el primario envía la PDU I_k antes de recibir la ACK A_k

Eficiencia de stop and wait

Se calcula como el tiempo que transmite una PDU, respecto al tiempo total que se necesita como minimo para transmitirla

$a = t_p/t_t$

Va en función del parametro a. Si 'a' es muy pequeña la eficiencia es próxima a 0.

3.2.1 Protocolos de transmisión continua

Stop and wait puede ser muy ineficiente si no se cumple que el tiempo de propagaci ón es mucho más pequeño que el tiempo de transmisión de las PDUs.

Los protocolos de transmisión continua resuelven el problema anterior dejando que el emisor envíe más de una PDU sin confirmar.

Cada vez que se envia una PDU de información se guarda en el buffer por si ha de reenviarse, y cuando recibe ACK la borra del buffer.

Si no hay errores, la eficiencia del protocolo es del 100%.

Go back N

La idea de este protocolo es que en caso de error, el emisor va atr ás hasta la PDU que falta al receptor y comienza a transmitir de nuevo desde este punto.

Hay diferentes maneras de implementarlo. Supondremos el algoritmo de a continuación:

- Las confirmaciones son acumulativas, es decir, la confirmación A_k confirma todas las PDUs con número de secuencia ¡= k;
- Si el secundario recibe una PDU con errores o fuera de secuencia, deja de enviar confirmaciones hasta que recibe correctamente la PDU que falta y descarta todas las PDU con numero de secuencia != k.
- Cuando salta el temporizador de transmisión de una PDU, el emisor retransmite la PDU I_k y continua con la transmisión de las siguientes.

Retransmisión selectiva

La idea es que el secundario no descarte nunca las PDUs que llegan correctamente, aunque lleguen fuera de secuencia (a diferencia de Go back N). Esto permite mejorar la eficiencia de Go back N en caso de

error, a costa de complicar la implementación del receptor.

Con la retransmisión selectiva se deberán almacenar y ordenar las PDUs que llegan fuera de secuencia.

Hay diversas maneras de implementar el protocolo. Supondremos el siguiente algoritmo:

- Las confirmaciones son acumulativas.
- Si el secundario recibe una PDU con errores o fuera de secuencia deja de enviar confirmaciones hasta que recibe correctamente la PDU que le falta y guarda todas las PDU que recibe con numero de secuencia != k.
- Cuando salta el temporizador de retransmisión de una PDU, el emisor retransmite la PDU I_k, pero no retransmite otras PDUs que ya había enviado antes.
- Cuando el receptor recibe una retransmisión, envia una conformación acumulada que confirma hasta la última PDU de la secuencia recibida correctamente.

3.3 Eficiencia en presencia de errores

En esta sección comparamos los protocolos ARQ básicos que hemos visto anteriormente cuando la probabilidad de perdida o error de una PDU es diferente de 0.

3.3.1 Stop and wait

3.3.2 Go back N

3.3.3 Retransmisión selectiva

3.4 Control de flujo

3.4.1 Protocolos de ventana

Ventana de transmisión: En transmisión continua el emisor puede enviar más de 1 PDU sin confirmar. Para poder hacer un control del flujo, hace falta introducir un **límite al número de PDUs que puede enviar.**

- El emisor puede enviar hasta W PDUs de información sin confirmar
- El receptor solo envia ACK de la PDU de información I_k después de que el nivel superior haya leido esta PDU.

El emisor transmite tan pronto como lo permite el nivel inferior. Antes de cada transmisión comprueba que la diferencia entre el numero de secuencia de la PDU a transmitir y la última confirmada sea inferior a la ventana.

- Si no es así el emisor se queda bloqueado hasta que llegan nuevas confirmaciones.

De lo anterior se puede deducir que **stop and wait es un protocolo de ventana con tamaño = 1**.

Una consecuencia del protocolo de ventana es que en el buffer de transmisión habrá como máximo W PDU, donde W es el límite de ventana.

Esto permite dimensionar el tamaño del buffer de transmisión y de recepción, ya que tendrán que almacenar como mínimo W PDUs.

3.4.2 Ventana óptima

Si el tamaño de la ventana es demasiado pequeño → el protocolo será ineficiente por el tiempo de espera de las confirmaciones.

Si la ventana es demasiado grande será un problema porque los buffers de transmisión y recepción deben dimensionarse para poder almacenar el número de PDUs igual a la ventana.

Ventana óptima: la ventana mínima que permite llegar a la velocidad efectiva máxima.

Si el tamaño de las PDU se fija con un tiempo de transmisión t_t segundos, entonces la ventana óptima es:

$$W = T_U/t_t$$

3.4.3 Dimensionado del campo "número de seqüencia" de las PDUs

3.5 El nivel de transporte de Internet

El nivel de transporte proporciona un canal lógico de comunicación entre las aplicaciones.

En la práctica el nivel de implementación esta formado por 2 procesos que se comunican entre ellos utilizando la red, ya que el nivel de transporte permite que los procesos se comuniquen entre ellos como si estuvieran en el mismo ordenador.

- Para conseguirlo, el nivel de transporte implementa un protocolo P2P entre los 2 hosts que se comniquen que hace un "multiplexado" de la información transmitida por los protocolos.

En internet hay 2 protocolos de nivel de transporte:

- UDP
- TCP

El multiplexado se consigue con un identificador de 16 bits llamado puerto, que identifica los procesos que se comunican.

En TCP/IP las conexiones suelen seguir el paradigma **cliente/servidor**.

- Servidor: espera las peticiones de los clientes. Para esto el servidor escucha las peticiones dirigidas a un puerto well-known.
  - Los puertos well-known tienen un valor en el intervalo [0...1023] y estan asignados por IANA.

- Cliente: es siempre quien inicia la conexión hacia el servidor y tiene un puerto asignado por el sistema operativo.
  - Rango del puerto [1024...10^{16} - 1]
  - El puerto se llama efímero, porque solo identifica el proceso del cliente mientras dura la conexión.

En UNIX el fichero /etc/services tiene el listado de los puertos well-kown.

3.6 El protocolo UDP

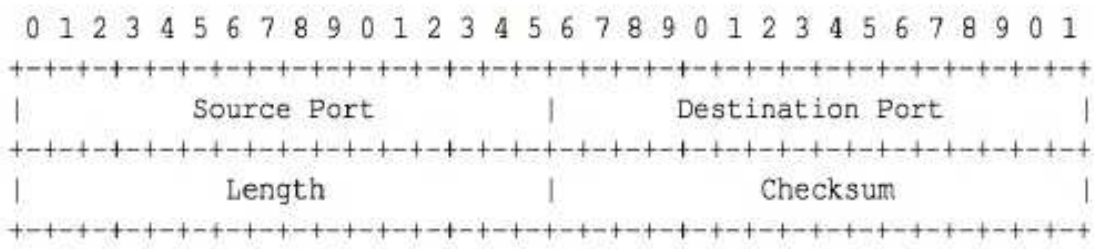


Figura 3.19: Capçalera d'un segment UDP.

Figura 3.20: Pseudocapçalera que es fa servir en el càlcul del checksum d'UDP.

Protocolo de transporte orientado a datagrama. Ofrece el mismo servicio que el transporte de datagramas del nivel IP.

- Básicamente lo único que hace es añadir la cabecera de la figura 3.19 a la información que recibe la aplicación para consumir un "datagrama UDP" y pasarlo al nivel IP para su transmisión.
- No es orientado a la conexión
- No fiable (si se pierde el datagrama UDP no lo retransmite).
- Cada operación de escritura a nivel de aplicación genera un datagrama UDP.

Las aplicaciones en tiempo real utilizan UDP. Por ejemplo telefonia o videoconferencia sobre TCP/IP.

- Estas aplicaciones no toleran retardos muy variables.
  - La aplicación en recepción lee el buffer constantemente.
  - Si un datagrama llega más tarde de cuando tocaría leerlo ha de descartarse porque sería inservible para la aplicación

La cabecera tiene un **tamaño de 8 bytes**. Hay 4 campos

- Puerto fuente y destino, para identificar los procesos que se comunican.
- Lenght con el tamaño total del datagrama UDP (payload UDP + 8)
- Checksum: protege la cabecera y el campo de datos
  - Se calcula aplicando el algoritmo de checksum conjuntamente a una pseudo cabecera, la cabecera UDP y el campo de datos.
    * La pseudocabecera tiene algunos campos de la cabecera IP para hacer una doble comprobación de que el datagrama ha llegado a una destinación correcta.
  - El checksum es **opcional**.
    * Si no se utiliza se envia un checksum igual a 0.
  - Debido a la pseudo cabecera si se utiliza NAT hay que recalcular el checksum de la cabecera UDP.

3.7 El protocolo TCP

Es el protocolo de nivel de transporte que se utiliza para **la transmisión fiable de información**. Es un protocolo:

- extremo a extremo
- ARQ
- orientado a la conexión

Objetivos:

- Recuperación de errores, para tener una transmisión fiable
- Control de flujo. Emisor no envia segmentos más rápido de lo que los puede procesar el receptor.
  - Se podrían perder segmentos por un buffer overflow del receptor
- Control de congestión. El emisor no envia a más velocidad de la que puede soportar la red.
  - Se podrían perder segmentos por el cuello de botella de la red
- Segmentos de tamaño óptimo: Va cogiendo bytes de la aplicación para generar segmentos de tamaño óptimo
  - **MMS: Maximum Segment Size**. Es el tamaño que se considera óptimo para el payload. Típicamente es el mayor tamaño posible, pero que no produce fragmentación a nivel IP.

La figura anterior muestra su funcionamiento:

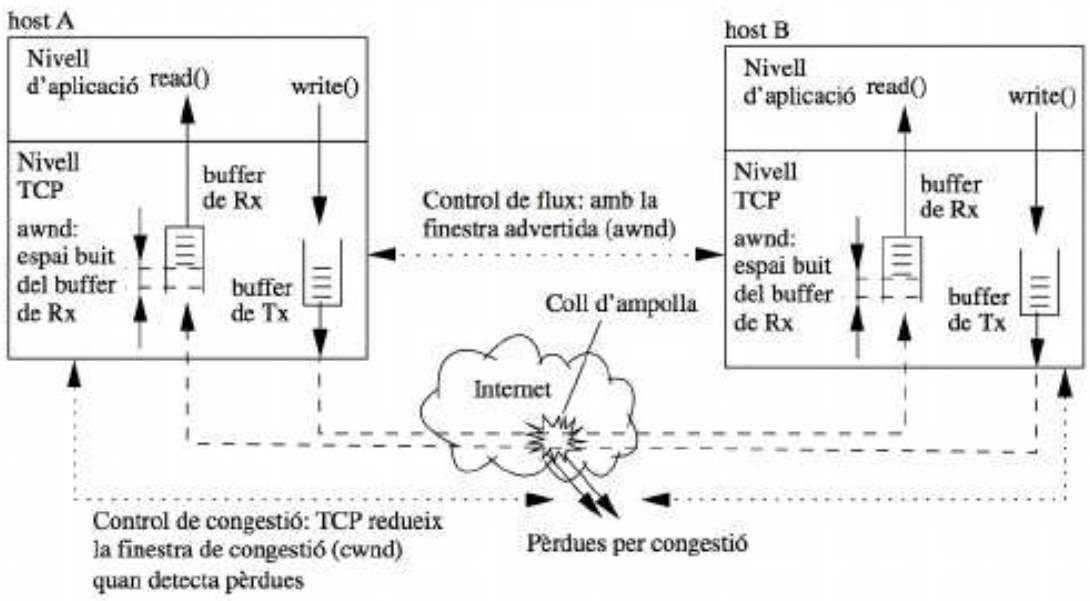


Figura 3.21: Funcionament bàsic de TCP.

- Buffer de transmisión y de recepción.
  - La aplicación hace las syscalls write() y read() para escribir y leer de estos buffers.
  - Cuando el buffer Tx esta lleno, la llamada write() queda bloqueada.
  - Cuando se han enviado datos del buffer Tx y se han confirmado se vacían del buffer y queda espacio libre para que la aplicación escriba.

3.7.1 Control de flujo

Ventana advertida (awnd):Para evitar un buffer overflow de Rx.

- Es una de las informaciones que siempre se envían en la cabecera TCP.
- En este campo se pone el espacio libre del buffer Rx
- Cuando TCP recibe un segmento, guarda el valor de la ventana advertida en la variable awnd.
- El emisor no puede enviar nunca más bytes sin confirmar de los que dice awnd.
  - De esta forma el receptor siempre tendrá espacio suficiente para guardar los bytes enviados en el buffer Rx.

3.7.2 Control de congestión

Ventana de congestión (cwnd): Para evitar perdidas por el bottle-neck de la red.

- El primario ha de reducir el numero de segmentos que envia. Ha de reducir la ventana de transmisión.

- La variable se incrementa si no se detectan perdidas y se decrementa si se detectan.

$$\text{wnd} = \text{minawnd, cwnd}$$

TCP no siempre ha de limitar la velocidad de transmisión de la conexión. De hecho podemos clasificar las conexiones en:

- **Bulk transfer (masivas)**: Durante la transmisión la aplicación siempre tiene datos listos para enviar.
  - El buffer de transmisión siempre esta lleno
  - TCP envía segmentos de tamaño máximo.
  - La ventana de transmisión limita la velocidad efectiva de la conexión.
  - Ej: ftp, mail, web...
- **Interactivas**: aplicaciones en que el usuario interactúa con una máquina remota, por ej. telnet.
  - La información se envía en mensajes de pocos bytes y de forma discontinua.
  - TCP no necesita introducir

3.7.3 Cabecera TCP

Mide 20 bytes, igual que la cabecera IP.

El checksum es obligatorio en TCP, a diferencia de UDP. Para el calculo se coge

- Pseudo cabecera para el calculo del checksum:
  - Source address y destination address
  - Zero
  - Protocol
  - TCP Lenght
- La cabecera TCP
- El payload del segmento

Maximum Segment Size (MSS—awnd): Se utiliza durante el establecimiento de la conexión para sugerir el valor de MSS al otro extremo.

- El valor que sugiere es la MTU de la red donde esta conectado, restando el tamaño de las cabeceras IP y TC (-40).

3.7.4 Números de secuencia TCP

En TCP los números de secuencia y las ventanas se miden en bytes. Es decir, si un segmento lleva S bytes, el número de secuencia del próximo segmento se incrementará en S.

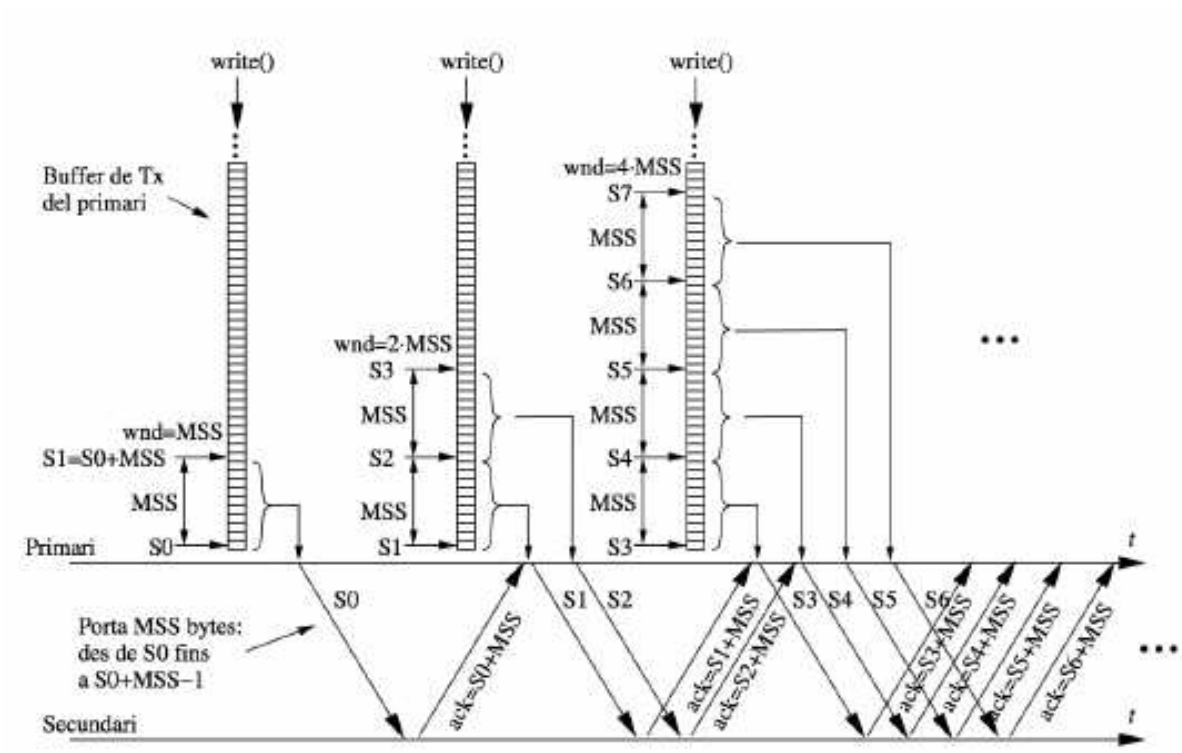


Figura 3.24: Evolució dels números de seqüència de TCP.

La figura anterior muestra la evolución de los números de secuencia a partir del primer segmento de datos de una conexión TCP.

El número de secuencia que lleva la cabecera identifica el primer byte de datos del primer segmento.

Las confirmaciones llevan un valor igual al número de secuencia del segmento que confirman (el primer byte del segmento) más el número de bytes de datos del segmento.

Si el segmento tiene numero de secuencia S_i, la confirmación lleva ack=S_i+MSS

3.7.5 Establecimiento de una conexión TCP

3.7.6 Diagrama de estados de TCP

3.7.7 Control de congestión

Una de las tareas principales de TCP es el control de congestión, es decir, adaptarse a la velocidad de transmisión que fija el bottleneck de la red. Sin control de congestión internet se volvería inestable y colapsaría.

Consiste en una "autoregulación" que hace TCP a través de la **cwnd** (ventana de congestión). La regulación de cwnd es:

- Multiplicative decrease: cuando detecta congestión reduce el valor de cwnd de manera multiplicativa (rápidamente)

- Additive increase: se incrementa lentamente e intenta buscar el punto donde no haya congesti ón, pero con el máximo aprovechamiento de las lineas de transmisión.

TCP utiliza 2 parejas de algoritmos básicos para ajustar cwnd:

- Slow start/ Congestion Avoidance
- Fast Retransmit/ Fast Recovery

Slow Start/ Congestion Avoidance

Hacen el control básico de la ventana TCP. Utilizan las siguientes variables:

- cwnd: es la ventana de congestión.
- snd_una (unacknowledged): Es el primer segmento no confirmado, es decir, el segmento que hace más tiempo que esta en el buffer de transmisión por ser confirmado.
- ssthresh (slow start threshold): límite entre las fases de slow start y cengestion avoidance.

```
Inicialització:
        cwnd = MSS ;
        ssthresh = infinit ;

Cada cop que es rep un ack de noves dades:
        si(cwnd < ssthresh) {
                cwnd += MSS ; /* Slow Start */
        } altrament {
                cwnd += MSS * MSS / cwnd ; /* Congestion Avoidance */
        }

Quan hi ha un timeout:
        Retransmetre snd_una ;
        cwnd = MSS ;
        ssthresh = max(min(awnd, cwnd) / 2, 2 MSS) ;
```

Figura 3.30: Algorismes de *Slow Start / Congestion Avoidance*.

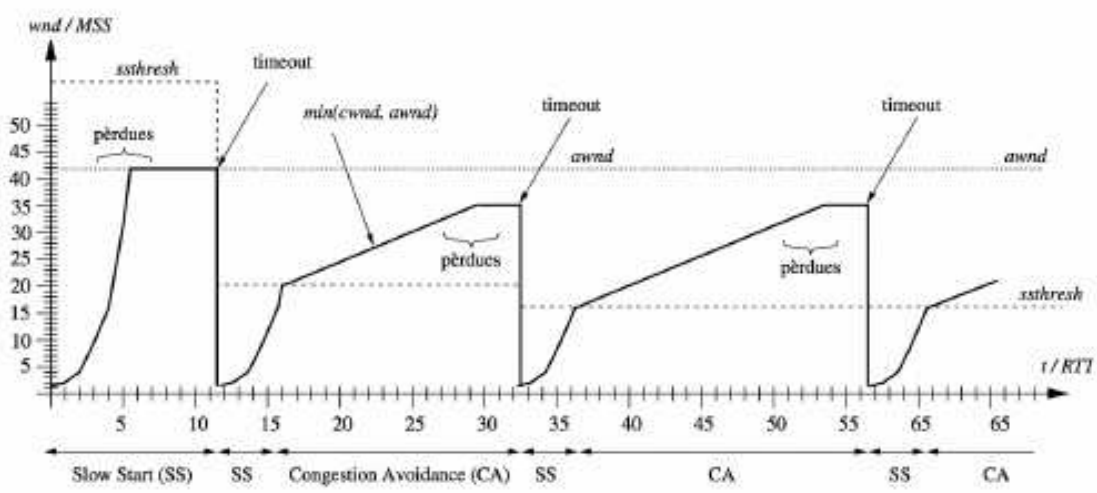


Figura 3.31: Evolució de la finestra de TCP quan hi ha un enllaç congestionat i només actuen els algorismes de *Slow Start / Congestion Avoidance*.

La figura 3.30 sumariza el funcionamiento de los 2 algoritmos.

- Cuando se inicia la conexión TCP *to* cwnd = MSS
  - TCP solo puede enviar un segmento sin confirmar (MSS bytes), de aquí el nombre slow start.
- Mientras no haya perdidas TCP incrementa cwnd en MSS para cada nuevo ACK
  - Cuando recibe 1 ack incrementa a 2*MSS, cuando recibe 2 a 4*MSS, 8, 16...

RTT (Round Trip Time: es el retardo que hay desde que TCP envia un segmento hasta que llega la confirmación.

Si los retardos son grandes y las lineas de transmisión rápidas, los ack suelen llegar en una ráfaga después de un **Round Trip Time**.

4

Redes de area local

4.1 Introducción a las LANs

Las redes se clasifican según su extensión en:

- WAN (Wide Area Network): redes de gran extensión, como la red telefónica. Es importante la escalabilidad.

- LAN (Local Area Network): tienen como condición de diseño la interconexión de un número limitado de estaciones en un area geográfica pequeña. La escalabilidad no es tan importante.

Según las estratégias de interconexión tenemos:

- Switched network (red conmutada): Formada por la interconexión de conmutadores que encaminan la información entre 2 nodos que se comunican. El objetivo es parecido a los routers.

- Multiaccess network (red de multiple acceso): el incoveniente de las redes conmutadas es el coste de los conmutadores. Una aproximación más económica es el acceso multiple a un medio compartido, como un bus, un anillo...

  - El emisor envia un mensaje que lo reciben todos, pero solo se lo queda al que va dirigido y el resto lo descartan.

  - Hace falta un protocolo de **control de acceso al medio (Medium Access Protocol, MAC)**.

4.2 Topologias

Se puede utilizar un espacio libre y una transmisión via radio (sin cables), pero tradicionalmente se han utilizado medios cableados.

Las topologías más utilizadas en la práctica han sido las de **bus y anillo**.

4.2.1 LAN en bus

El bus puede consistir en un cable coaxinal. Las estaciones estan conectadas al bus por unos *taps* que interceptan el bus.

Cuando una estación transmite, la señal se propaga por el bus y la reciben todas las estaciones.

En los extremos del bus hay unas resistencias llamadas *terminadores* que evitan que la señal se refleje y provoque ecos e interferencias.

4.2.2 LAN en anillo

Cada estación se conecta al anillo con un repetidor que intercepta el anillo. Los repetidores pueden estar en 3 estados:

- Recepción: el repetidor descodifica la señal y después de un cierto retraso libera los bits descodificados a la estación que tiene conectada, y vuelve a codificar la señal para enviarla al anillo abajo.

- Transmisión: el repetidor descodifica la señal y después de un retraso libera los bits descodificados a la estación que tiene conectada. Retransmite abajo los bits que recibe la estación.

- Cortacircuito: si no hay ninguna estación conectada o si alguna falla, el repetidor puede pasar a un cortacircuito. Es como si el repetidor no estuviese.

Token-ring fue una red LAN desarrollada por IBM, pero que ha quedado prácticamente substituida por Ethernet

4.3 Arquitectura IEEE de una LAN

El organismo de estandarización de LANs ha sido IEEE. Los estándares que desarrolla tienen el prefijo 802.

- 802.3 *to* Ethernet

- 802.5 *to*4 Token-Ring

- 802.11 *to* Wireless Fidelity (WiFi)

OSI se divide en 2 niveles: Logical Link Control (LLC) y Medium Access Control (MAC).

4.3.1 Logical Link Control

Define la interfície de nivel superior. El estándar define 3 tipos de servicios:

- No orientados a la conexión

- Orientados a la conexión

- Confirmados y no orientados a la conexión

El estándar se refiere a los campos *Destination SAP* y *Source SAP* como direcciones LLC, pero tienen poco que ver con el concepto de dirección visto hasta ahora.

SAP: Service Access Point. Punto de comunicación entre niveles. En el caso de LLC, tiene un significado análogo al campo de protocolo de la cabecera IP: identifica el nivel superior donde ha de entregarse el contenido de la trama.

4.3.2 Medium Access Control

A diferencia de LLC, MAC es diferente para cada tecnología LAN.

Su objetivo es regular el acceso al medio compartido.

La estructura de datos (PDU) de nivel MAC se llama "trama" y es lo que se transmitirá por la red física. Contiene:

- Dirección MAC fuente
- Dirección MAC destinación
- Control: información adicional para el funcionamiento de protocolo
- Payload: lleva la PDU de nivel superior
- CRC: lleva el control de errores de toda la trama

4.4 Tipos de MAC

El protocolo MAC ha de ser:

- **Distribuido:** no ha de hacer falta un árbitro regulador para controlar el acceso.
- Eficiente
- Equitativo: no han de haber estaciones privilegiadas que puedan conseguir velocidades de transmisión mayor que las otras.

2 estratégias para el diseño de protocolos MAC

Paso testimonial

El acceso esta regulado por un "testimonio" (token). La estación que tiene el token es la que puede transmitir. El token se va pasando entre las estaciones, típicamente después de la transmisión de una trama.

Acceso al medio aleatorio

No existe token. Las estaciones intantan transmitir y si se da la casualidad de que 2 lo hacen simult´aneamente (colisión), se esperan un tiempo aleatorio y lo vuelven a intentar.

Para que MAC sea equitativo todas las estaciones deben tener el mismo tiempo de backoff.

Las colisiones no son un mal funcionamiento, si no que hacen un trabajo similar al del token.

En la práctica:

- MACs por token para LANs de anillo (FDDI, Token-Ring)
- MACs aleatorios para LANs en bus (ethernet)

4.5 Protocolos de MAC aleatorios

4.5.1 Aloha

Se desarrollo a principios de los 70 en la Universidad de Hawaii, con el objetivo de comunicar los computadores de los campus que estaban distribuidos por diferentes islas, via radio y de la forma m ás sencilla posible.

Reglas básicas del protocolo

- Cuando una estación tiene una trama lista para transmitir, la envía
- Después de enviar una trama, la estación espera un ack.
  - Si no llega en un tiempo fijado, la estación supone que se ha producido una colisión. → la estación espera un tiempo aleatorio (tiempo de backoff) y vuelve al primer punto.
- Si una estación recibe una tama correcta envía inmediatamente la confirmación.

4.5.2 Carries Sin Multiple Aceso (CSMA)

Mejora la baja eficiencia de Aloha. CSMA en lugar de transmitir inmediatamente al tener la trama lista, cosa que provoca muchas colisiones, escucha el medio antes de transmitir. Si el medio esta libre transmite inmediatamente, si no lo esta, espera a que lo esté.

Igual que en Aloha, en CSMA también hay confirmaciones y su ausencia se identifica con una colisión. En este caso la trama se intentará retransmitir tras un tiempo de backoff.

En caso de que le medio este ocupado hay 2 estrategias:

- **CSMA 1 persistente**: cuando la estación ve el medio libre transmite inmediatamente.
  - Se llama *1 persistente* porque transmite con probabilidad 1.

  - Tiene el inconveniente de que si hay diversas estaciones que estan esperando a transmitir (o la estación tiene + de 1 trama a transmitir), se producirá colisión

- **CSMA no persistente**: Cuando el medio se libera espera un tiempo aleatorio (diferente a back-off) y repite el algoritmo CSMA-no persistente

4.6 Ethernet

Diseñado por Bob Metcalfe para interconectar los ordenadores de Xerox (empresa dónde trabajaba). Se basó en Aloha para su diseño.

Actualmente se ha convertido en la tecnología LAN más utilizada.

4.6.1 Tramas ethernet

Existen 2 formatos de tramas ethernet:

- Ethernet II o DIX: primeras tarjetas ethernet comercializadas por Intel, Xerox (DIX)...
  - No se utiliza el sub-nivel LLC
- IEEE.802.3: estándar aprobado por IEEE

Preamble (8 bytes)	Destination MAC Address (6 bytes)	Source MAC Address (6 bytes)	Frame type (2 bytes)	Payload (46 to 1500 bytes)	CRC (4 bytes)

Figura 4.10: Format de la trama Ethernet II o DIX.

Preamble (8 bytes)	Destination MAC Address (6 bytes)	Source MAC Address (6 bytes)	Length of the frame (2 bytes)	Payload (46 to 1500 bytes)	CRC (4 bytes)

Figura 4.11: Format de la trama IEEE-802.3.

Campos:

- Preamble: para sincronizar las tarjetas en la recepción.
- Destination y Source Address: Identifican estación emisora y receptora. Tiene 6 bytes (48 bits) y son únicas de cada tarjeta. IEEE subministra bloques de direcciones a las empresas para garantizar que sean unicas
- Type: identifica el protocolo de nivel superior
- Payload: campo de información. Entre 40 y 1500 bytes. Si el número de bytes es menor a 46, ethernet añade bytes adicionales hasta los 46.

  - Hace falta un mecanismo para detectar los bytes añadidos → en IP, el header lenght de la cabecera.

- CRC: para la derección de errores

IEEE ethernet lleva un campo lenght en lugar de tipo. Esto permite que no haga falta un mecanismo para detectar los bytes añadidos por ethernet cuando el payload ¡ 46 bytes.

Para hacer compatible los 2 formatos, el campo de tipo en DIX, es siempre mayor a 1500 y así la tarjeta cuando recibe detecta el tipo.

Todos los estándares de IEEE utilizan el sub-nivel LLC. El protocolo de nivel inferior y superior vendr án identificados por los campos **DSAP y SSAP** del nivel LLC.

Para hacer compatibles los 2 formatos, IEEE definió una extensión de LLC llamada SNAP (Sub-Network Access Protocol).

**********FALTA**********

4.6.2 Protocolo MAC Ethernet

El protocolo que se utiliza se conoce como **CSMA with Collision Detection (CSMA/CD)**. La idea es la misma que CSMA pero ahora continua escuchando al medio mientras transmite una trama y deja de transmitir inmediatamente si detecta una colisión.

- Si no se detecta

4.6.3 Tamaño mínimo de una trama Ethernet

4.6.4 Funcionamiento full duplex

4.6.5 Nivel físico Ethernet

4.7 Switches Ethernet

Un elevado nombre de colisiones reduce la eficiencia de una LAN.

Cuando el numero de estaciones crece, es necesario segmentar o dividir el *dominio de colisiones*.

Dominio de colisiones: número de estaciones que comparten el mismo medio de tranmisión y que puede colisionar directamente entre ellas.

- Un conjunto de estaciones connectadas a un segmento 10Base5, 10Base2 o un hub forman un dominio de colisiones.

Un **router** segmenta el dominio de colisiones, porque cuando recibe una trama por un puerto, no la transmite simultáneamente por ningún otro puerto (como haría un hub), en su lugar utiliza **Store and Forward**

1. Captura la trama completamente

2. Procesa el datagrama IP y decide por que puerto ha de transmitirse

3. Se almacena en la cola de transmisión del puerto que toca

4. Se transmite la trama accediendo al medio como lo haría cualquier otra estación.

Cada puerto del router es un dominio de colisiones diferente.

Bridges: dispositivos creados para segmentar el dominio de colisiones, más económicos que los routers.

5. Nombre reducido de puertos

6. Hace un procesado **store and forward**

7. A diferencia del router, a nivel 2, tiene una NIC diferente para cada puerto

El **bridge** contiene una **tabla MAC** donde hay tuplas MAC, #puerto. Cuando llega una trama:

1. Captura la trama completamente

2. Mira la tabla MAC para saber por que puerto ha de transmitirse

3. Procesa el datagrama IP y decide por que puerto ha de transmitirse

4. Se almacena en la cola de transmisión del puerto que toca

5. Se transmite la trama accediendo al medio como lo haría cualquier otra estación.

La tabla MAC se forma de manera automática (**learning bridges**). Cada vez que llega una trama

1. El bridge mira la dirección de origen.

2. Después mira la dirección de destino.

   (a) Si la dirección no se encuentra en la tabla MAC $\rightarrow$ se hace una copia de la trama en todos los puertos (excepto en el de llegada)

   (b) Si esta en la tabla se copia solo en la cola de transmisión del puerto que dice la tabla

3. Como en las caches ARP, las entradas tiene un time-out para mantener la tabla pequeña.

Switches: introducidos después de los bridges, teniendo más puertos y mayor capacidad de conmutación de tramas entre los puertos. Son capaces de conmutar tramas simultáneamente entre puertos diferentes.

- Una estación A conectada a puerto 1 puede transmitir hacia una estación B conectada al puerto 2, mientras una estación C en puerto 3 transmite hacia estación D en puerto 4.

Lo anterior quiere decir que si los puertos son a 10Mbps, las estacions A i B podrán transmitir de manera simultánea a aprox. 10Mbps hacia las destinaciones.

4.7.1 Spanning Tree Protocol

Un switch tiene funciones de encaminamiento, que a primera vista podrían parece similares a las de un router.

En una red formada por switches, cuando las tablas MAC están inicializadas, las tramas "saltan" de un switch a otro desde origen a destinación.

Diferencia importante: los routers se pueden conectar con una topología arbitraria (el algoritmo de encaminamiento se encarga de actualizar las tablas adecuadamente).

- **Los switches en cambio no admiten topología arbitraria**
  - Un switch retransmite una trama por todos los puertos menos por el que ha llegado
    - Puede quedarse una trama circulando infinitamente → saturación de la red.

En una red con switches no pueden haber bucles, pero en la práctica es deseable que los pueda haber (ej: tolerancia a fallos).

STP (Spanning Tree Protocol: solventa los problemas que representan los bucles.

- Los switches que lo implementan intercambian mensajes para dejar una topología en árbol, en la que no haya bucles.
  - Los puertos que están en este estado, descartan todas las tramas de datos que reciben y, por tanto, no participan en la inicialización de la tabla MAC.

4.7.2 Dominios broadcast

Los switches permiten aumentar la escalabilidad de la red al segmentar el dominio de colisiones.

Una LAN con switches es una mezcla entre medio compartido y medio conmutado.

Una vez que los switches tienen las MAC inicializadas, encaminan tramas para que atraviesen solo los enlaces necesarios para llegar al destino.

Los switches no resuelven las tramas broadcast, y es un factor que limita la escalabilidad.

- Cuando switch recibe un broadcast, lo envia por todos los puertos, menos por el que lo ha recibido.

Todas las redes conectadas por hubs y switches forman un **dominio de broad**

4.7.3 Control de flujo

Los puertos tienen un mecanismo de detección para asignar una velocidad a los dispositivos conectados. Por ejemplo un switch 10/100, los puertos pueden transmitir a 10Mbps o 100Mbps.

Si un dispositivo con NIC de 100Mbps transmite a uno con NIC de 10Mbps, el de 10 se llenara rápidamente y empezaran a producirse perdidas de tramas si no hace nada.

Cuando pasa lo anterior se activa el **mecanismo de control de flujo**

4.7.4 Repartición de medio de transmisión

4.7.5 LANs virtuales

Por motivos de eficiencia y seguridad, los servidores y hosts han de estar en relacionados en dominions de broadcast diferentes, pero un switch Ethernet forma un único dominio broadcast.

VLANs: permiten que la distribución lógica de los dominios de broadcast no corresponda con la distribución y conexión física de los conmutadores

4.8 LANs sin cable

4.8.1 Mecanismo CSMA/CA

4.8.2 Tramas 802.11

4.8.3 Direccionamiento

5

Aplicaciones en red

5.1 Charsets

Las diferentes culturas de un idioma se identifican de la siguiente forma:

- es_ES
- es_CO
- en_EN
- en_GB

Hay alfabetos que son compartidos entre diferentes lenguajes, por ejemplo el catalán y el francés.

5.1.1 Alfabetos

La base de los alfabetos es **ascii**

Los caracteres se codifican siguiendo diferentes convenciones:

- Repertorio: un conjunto de carácteres
- Código: correspondencia entre repertorio y números naturales
- Codificación: algoritmo que convierte los códigos numéricos a una secuencia de octetos (¿ 256 carácteres)

US-ASCII: 95 carácteres + control = 128: 7 bits (1 octeto)

ISO 8859-1 (ISO Latin 1): 190 + contro = 256: 1 octeto. **Por defecto para HTTP**

5.2 DNS

Permite a los usuarios utilizar nombres en lugar de direcciones ip.

- Sigue el paradigma cliente/servidor
  - Nivel de transporte TCP/UDP
  - Puerto well-known 53
- Hay una base de datos con los nombres y las direcciones para poder hacer la resolución
- El sistema de nombres esta organizado en una jerarquía
  - Permite distribuir la base de datos alrededor del mundo, en lugar de tenerla centralizada en 1 servidor

Los **nombres** consisten de:

- node-name: rogent, www...
- domain-name: ac.upc.edu, upc.edu...

rogent.ac.upc.edu, www.upc.edu

El DNS consiste de una base de datos mundial distribuida.

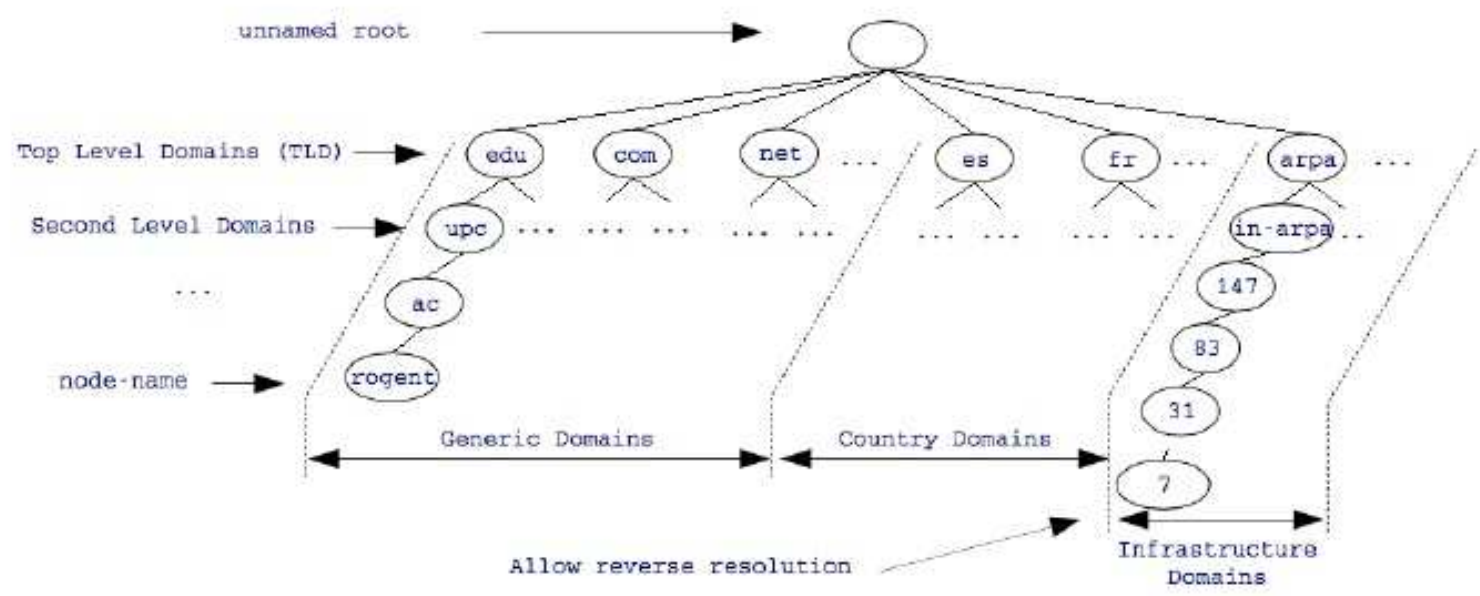
Resource Records (RRs): Las entradas en la base de datos DNS.

- La información asociada a un nombre se compone de 1 o más RR

Los nombres son case-insensitive.

5.2.1 Jerarquía de dominios

La base de datos de DNS esta organizada en un árbol:



La **ICANN** es la responsable de gestionar y coordinar el DNS.

- Delega los **Top Level Domains (TLD)** a los registradores:http://internic.net

- Los dominos delegan la administración de subdominios

El **dominio raíz** no tiene nombre y de el se desprenden los TLD (Top Level Domains).

Cada **TLD** tiene un administrador (registar) que delega parte del dominio en "sub-dominios".

- Del TLD edu cuelgan los universitarios.
  - De aquí cuelga el sub-dominio upc, la administración del cual esta delegada a la UPC.

Los nombres se escriben comenzando por el host y separando los dominios por puntos hasta el TLD.

- El **nombre** se puede hacer **acabar en punto** para indicar que se especifican todos los dominios (fully qualified domain name)
  - **rogent.ac.upc.edu.**

5.2.2 Organización de la base de datos

Cada administrador de un dominio ha de mantener parte de la base de datos de DNS en un **servidor primario** y uno o más **servidores secundarios**.

- Estos servidores se conocen como **autoridad (authority)**
- En estos servidores tiene que haber
  - El nombre y dirección de los host que cuelgan de su dominio
  - La dirección de los servidores primarios y secundarios de las autoridades de subdominios que han delegado.

En la época en la que se escribió el libro habían **13 root-servers** que contenían las direcciones de los TLD. Estan distribuidos por el mundo y tienen por nombre *a.root-server.net, ... , m.root-server.net*.

El acceso a la base de datos DNS se hace a utilizando un **Name Server (NS)**

- Contiene RR permanentes y cacheados. Los cacheados se eliminan después de un tiempo
  - ∗ Cada subdominio tiene una autoridad que consiste en un **primario** y en un NS de backup.
- En este contexto, los subdominoos se refieren como zonas, y los subdominios delegados subzonas.

Una autoridad tiene la información completa de una zona:

- Nombres y direcciones de todos los nodos en la zona
- Nombres y direcciones de todas las autoridades de subzona.

Root Servers: son el punto de entrada a la jerarquía de dominios.

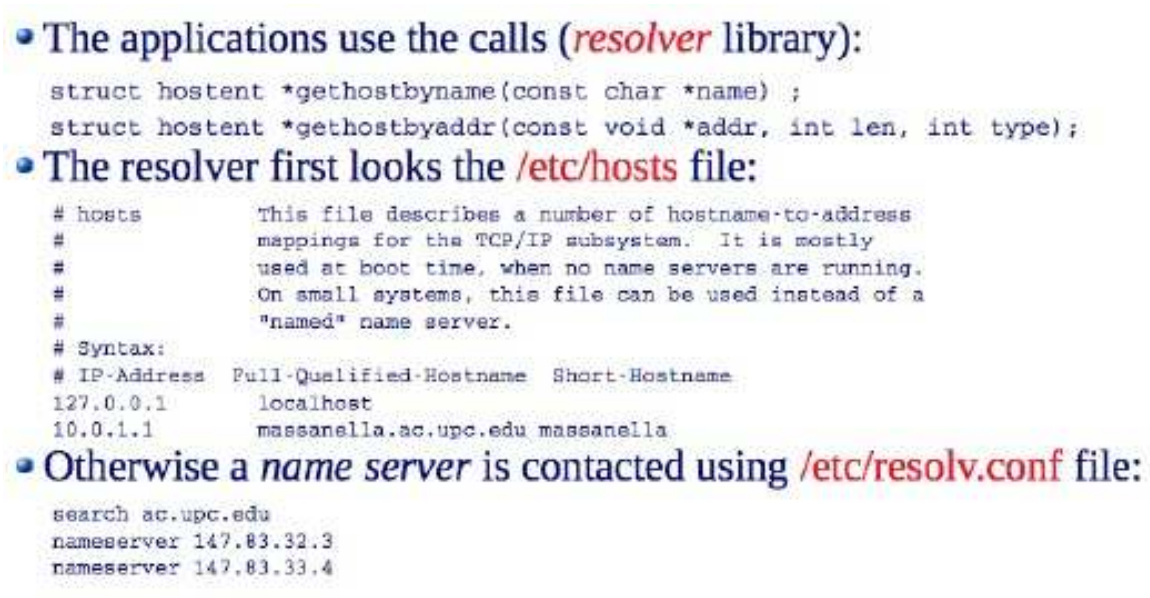
- Están distribuidos por el mundo y tienen las direcciones de los **TLDs**

- Las direcciones de los Root Servers son necesarias en la configuración de los **Name Servers**

5.2.3 Ejemplo en UNIX: El Resolver

Utilizan el sistema linken con las funciones de la librería de resolución de nombres llamada **resolver**. Las funciones son:

- gethostbyname(): devuelve la IP de un nombre
- gethostbyaddr(): resolución inversa



Cuando se llama a resolver:

- Se mira el fichero /etc/hosts donde hay nombres y direcciones IP.
- Se contacta con un servidor de nombres (NS) si no se puede resolver con /etc/hosts
  - /etc/resolv.conf contiene el dominio local y la dirección IP del servidor primario y secundario del dominio.

```
exemple# cat /etc/hosts
201.24.31.87 pc1.uu.vi.com pc1
201.24.31.105 pc2.uu.vi.com pc2
201.24.31.106 pc3.uu.vi.com pc3
```

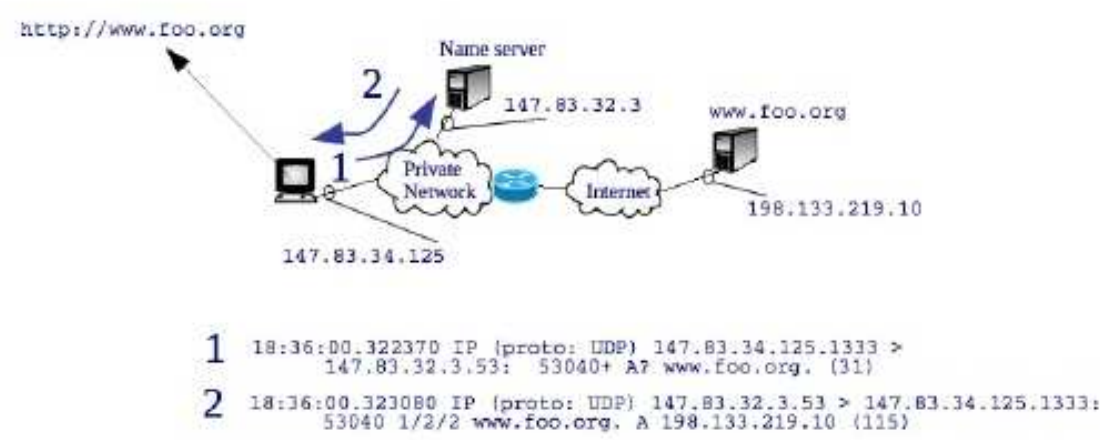
(a) Fitxer /etc/hosts.

```
exemple# cat /etc/resolv.conf
domain uu.vi.com
nameserver 201.24.31.3
nameserver 201.24.31.4
```

(b) Fitxer /etc/resolv.conf.

5.2.4 Protocolo

- Paradigma cliente servidor
- UDP/TCP. Para mensajes cortos usa UDP
- Puerto well-known: 53



5.2.5 Configuración básica del NS en unix

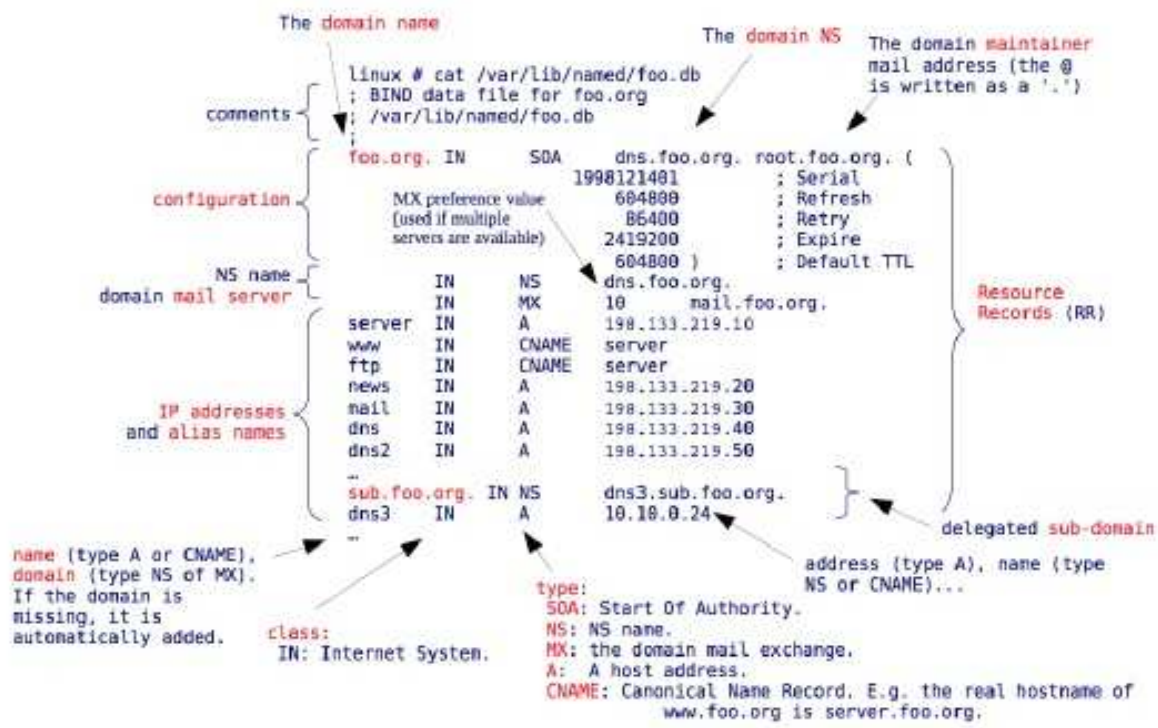
La implementación del NS en unix es **BIND** (Berkeley Internet Name Domain)

named: el daemon de BIND NS

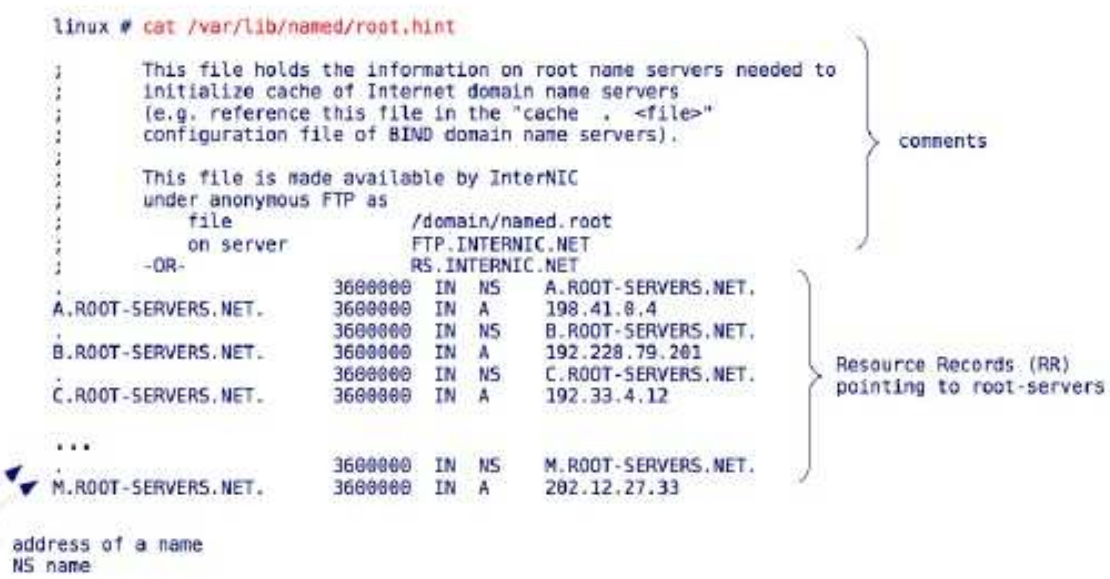
Los **archivos básicos** de **configuración** son:

- /etc/named.conf → configuración global
- /var/lib/named/root.hint → direcciones de root servers
- /var/lib/named/*.db → archivos de zona

5.2.6 Ejemplo: archivo de zona



5.2.7 Ejemplo: dirección root servers



5.2.8 Resolución DNS

Los **Name Servers** cachean las resoluciones de nombres. Las resoluciones cacheadas se devuelven sin consultar en el NS de autoridad.

Un mismo nombre puede estar asociado con diferentes direcciones IP (load balancing)

Las **direcciones** de un **dominio común** pueden **no estar en una misma red ip.**

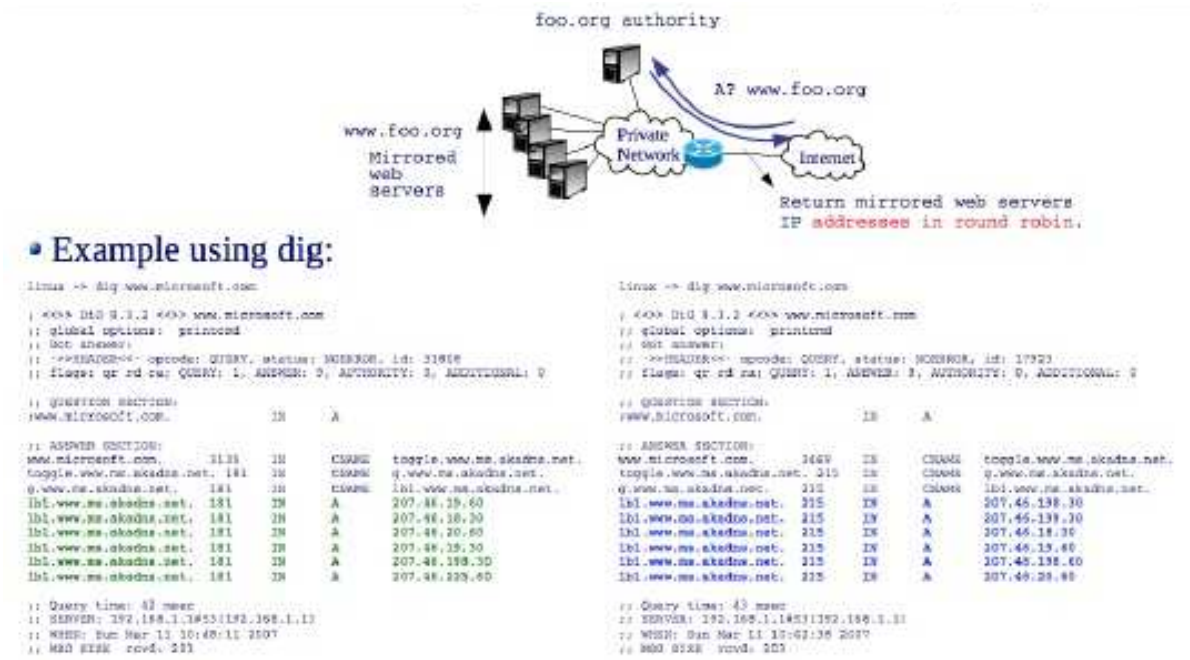
Cuando un host quiere resolver una dirección, (pe. www.foo.com):

- Envía el nombre que se quiere resolver a su servidor DNS.
- El servidor envía la petición a un root-server, que le devuelve la dirección del servidor de nombres (NS) del TLD.
- El servidor se dirige al servidor de dominio *com*
  - Devuelve la dirección del servidor de nombres del **second level domain** → **foo.com**
- El servidor se dirige al servidor de dominio **foo.com**, que le devuelve la dirección buscada.
- El servidor devuelve la dirección buscada al host que lo había solicitado.

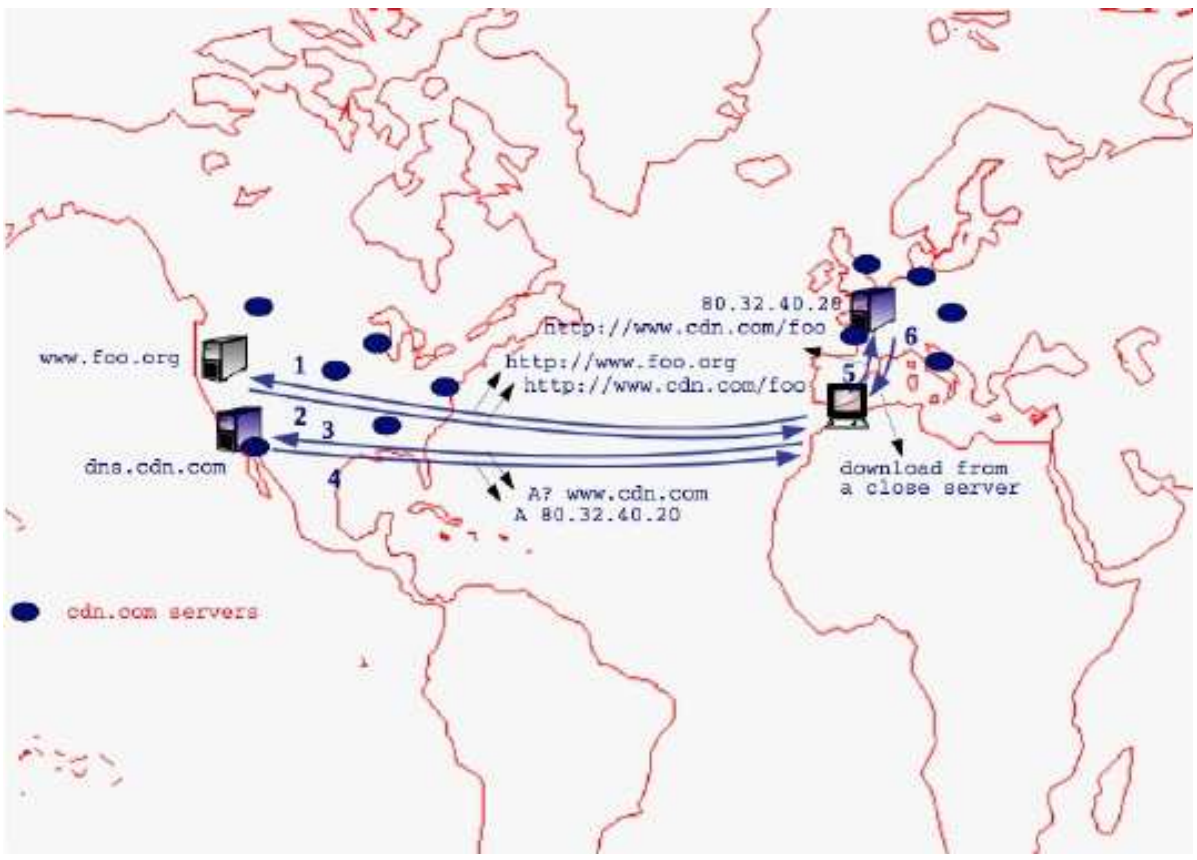
Resolución recursiva: La hace el host y el resultado es la dirección que se busca **Resolución iterativa:** La hace el servidor. Se llama así porque consulta iterativamente los servidores de los dominios hasta que resuelve la dirección buscada.

5.2.9 Ejemplo: load balancing

Ejemplo: Content Distribution Networks Hay empresas que se dedican a la distribución de contenidos. Tienen servidores distribuidos por el mundo, dónde replican los contenidos que sus clientes desean publicar.



Cuando ser recibe una petición se utilizan técnicas sofisticadas para resolver la IP del servidor más conveniente.



5.2.10 Formato de mensaje

Todos los mensajes DNS tienen el mismo formato:

- Header: tipo de mensaje
- Question: lo que ha de resolverse
- Answer: la respuesta a question

- Authority: nombres de dominio de autoridad
- Additional: tipicamente, la dirección de la autoridad de nombre.

Header

Identificación: 16 bits random utilizados para relacionar query/response

Flags

- QR (Query-Response): índica si es mensaje de query o response
  - 0: query
  - 1:response
- AA (Authoritative Answer): indica si ha respondido la autoridad de dominio o si la respuesta estaba cacheada.
- Rd (Recursion Desired): si esta activada, indica que se quiere que la resoluci ón sea recursiva.

Otros campos indican:

- Número de la Question, Answer, Authority
- Campos adicionales del mensaje

Question

QName: indica el nombre que ha de resolverse

QType Indica el tipo de pregunta

- A: Addres
- NS: Name Server
- PTR: Pointer. Para una resolución inversa
- MX: Mail Exchange. Domain Mail Server address.

Qclass: Indica el tipo de dirección que quiere resolverse. Para las direcciones de internet es 1.

Resource Recors (RRs)

Los campos Answer, Authority y Additional están compuestos de uno o más RRs:

- **Name, Type, Class:** Lo mismo que en el campo Question.

- **TTL (Time To Live:** segundos que puede ser cacheado.
- **RDLenth:** Tamaño del RR en bytes
- **Rdata**
  - Direccion IP si Type es A, nombre si Type es NS, MX o CNAME.

5.2.11 Ejemplo: Mensaje DNS

DNS – Messages: Example

```
# tcpdump -s1500 -vvpni eth0 port 53
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 200 bytes
11:17:30.769328 IP (UDP, length: 55) 147.83.30.137.1042 > 147.83.30.70.53: 36388+ A? ns.uu.net. (27)
11:17:30.771324 IP (UDP, length: 145) 147.83.30.70.53 > 147.83.30.137.1042: 36388
                q: A? ns.uu.net. 1/2/2 ns.uu.net. A 137.39.1.3
                ns: ns.uu.net. NS auth00.ns.uu.net., ns.uu.net. NS auth60.ns.uu.net.
                ar: auth00.ns.uu.net. A 198.6.1.65, auth60.ns.uu.net. A 198.6.1.181 (117)
```

Query message:

- 36388: Identifier.
- +: Recursion-Desired is set.
- A?: Qtype = A.
- ns.uu.net.: Name to resolve.

Response message:

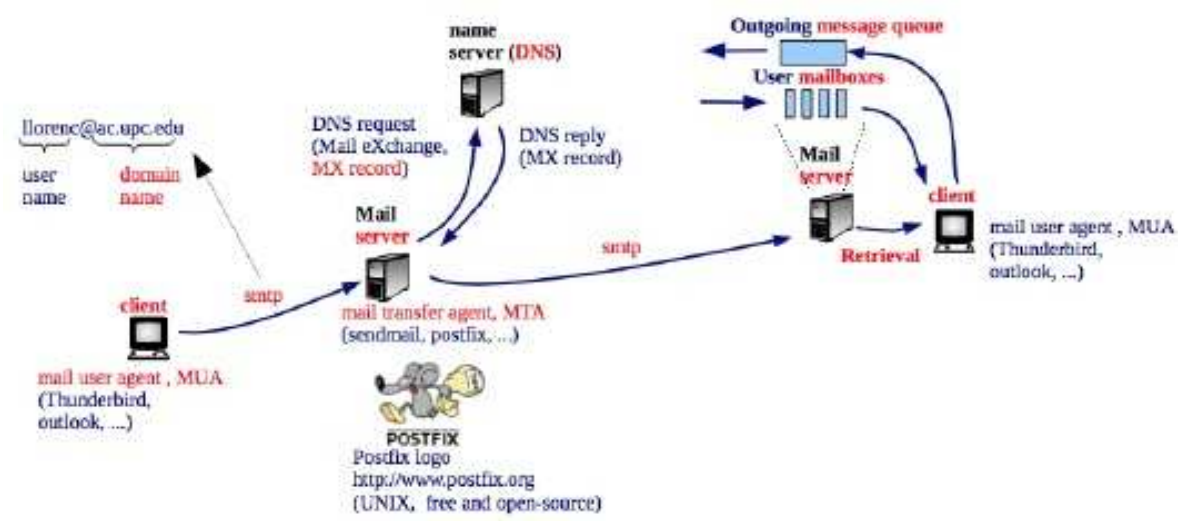
- 36388: Identifier.
- q: A? ns.uu.net.: Repeat the Question field.
- 1/2/2: 1 Answers, 2 Authorities, 2 Additional follows.
- ns.uu.net. A 137.39.1.3: The answer (RR of type A, address: 137.39.1.3).
- ns: ns.uu.net. NS auth00.ns.uu.net., ns.uu.net. NS auth60.ns.uu.net.: 2 Authorities (RRs of type NS: the domain ns.uu.net. authorities are auth00.ns.uu.net. and auth60.ns.uu.net).
- ar: auth00.ns.uu.net. A 198.6.1.65, auth60.ns.uu.net. A 198.6.1.181: 2 Additional (RRs of type A: authorities IP addresses).

5.3 Email

Componentes:

- Capa de transporte: TCP, puerto well-known 25
- Application layer protocol: SMTP (simple mail transfer protocol)
- Retrieval protocols (IMAP, POP, HTTP)
  - POP: Post Office Protocol
    - escucha al puerto 110
    - El usuario elimina el mensaje al recibirlo
  - IMAP
    - servidor IMAP escucha el puerto 143
    - el mensaje se mantiene en el servidor a no ser que el usuario lo elimine
    - Proporciona comandos para crear carpetas, mover mensajes, descargar solo una parte del mensaje...
  - Web base Email (HTTP)
    - Un servidor web gestiona el mailbox del usuario. Se utiliza un navegador web, por tanto se utiliza HTTP para enviar y recibir mensajes.

5.3.1 SMTP



Diseñado como un protocolo simple (pocos comandos) y basado en texto (ASCII).

Comandos básicos:

- HELO: identifica el cliente SMTP
- MAIL FROM: identifica el emisor
- RCPT TO: identifica el receptor
- DATA: mensaje
- QUIT: cierra la transacción

Respuestas del servidor: dígito de 3 numeros, que identifica el estado siguiente al que debe entrar el cliente, y un mensaje entendible por humanos.

5.4 MIME: Multipurpose Internet Mail Extensions

- Utilizado en mail, web...
- Especificación para el transporte de objetos multimedia
  - El receptor puede presentarla automaticamente
  - Codificado para facilitar la transmisión
- El formato interno es invisible a los usuarios
- Incluye uno o más objetos, texto en diferentes alfabetos, archivos grandes...

5.4.1 MIME content type

- Text
  - plain
  - richtext
  - cualquier formato de un procesador de textos
  - html, xml...
- Image
  - png, jpeg, gif...
- Audio
  - basic
- Video

  - mp4

- Application: contenido específico de aplicación. Binario de datos no interpretado
  - json
  -

- Multipart: multiples partes de datos independientes
  - mixed:
  - alternative: representan los mismos datos en formatos diferentes
  - form-data, digest...

- Message
  - mensaje encapsulado que puede contener su propio campo de header Content-Type. Ej: cuando queremos responder a un mail incorporando el mensaje de origen.
  - partial: mensajes parciales para permitir la fragmentación

5.4.2 Multipart

El campo de multipart Content-Type debe aparecer en la cabecera de la entidad.

El body contiene 1 o más partes de body, cada una precedida por un **boundary de encapsulación** y el último cerrado por un boundary de cierre.

Lo siguiente indica que la entidad consiste de múltiples partes, cada una con una infraestructura sintácticamente igual a un mensaje RFC 822, excepto el área de header que puede estar completamente vacía.

Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08jU534c0p

Cada parte iría precedida por:

–gc0p4Jq0M2Yt08jU534c0p

5.4.3 MIME: codificación de la transferencia

Formas de codificar el contenido para que vaya sobre un medio de transporte de 7 bits.

****

- Quoted-printable: La mayoría del texto son 7 bits → tranformar algunos carácteres: €= E4
- Base64

  - 3 bytes /24 bits $\Longleftrightarrow$ 4 ASCII (32 bits)

- Binario: sin codificación. Cualquier carácter y lineas de cualquier longitud.
- 7Bit: sin codificación de caracteres (todo 7 bits) y lineas de la longitud apropiada.
- 8Bit: sin codificación de caracter (8 bits) y lineas de la longitud apropiada

5.5 Web

5.5.1 Web links

URI: Uniform Resource Identifier. Sintaxis genérica para identificar un recurso

URL: Uniform Resource Locator. Subset de URIs que identifican un recurso en internet.

La sintaxis de URL es:

scheme://username:password@domain:port/path?query string#fragment id

5.6 HTTP messages

Metodos:

- Get: comando típico. Pide un objeto
- POST: pide un objeto según los datos en el cuerpo del mensaje de la petición. Estos datos son el contenido de formularios html, dados por el cliente.
- Header: permite al cliente dar información adicional sobre la petición y sobre él mismo.

POST utiliza MIME types: application/octet-stream, para enviar datos planos y application/x-www-form-urlencoded, para enviar name-value pairs.

5.6.1 Mensaje HTTP

5.6.2 Conexiones persistentes y no persistentes

No persistente:

- Por defecto en HTTP/1.0
- El servidor cierra la conexión después de cada objeto
  - Ej: para una web html con 10 imágenes jpeg, se abren secuencialmente 11 conexiones HTTP (1 html + 10 imagenes).

Persistente:

- Por defecto en HTTP/1.1
- El servidor mantiene abierta la conexión TCP hasta que pasa un tiempo de inactividad.
  - Ej: los 11 objetos de la no persistente anterior se envían en 1 sola conexión TCP.

Persistente con pipelining:

- Soportada solo por HTTP/1.1
- El cliente va haciendo peticiones a medida que encuentra una nueva referencia, aunque aún no se haya completado la descarga de los elementos.

Pipelining vs. no pipelining:

- El **pipelining no suele venir activado** por defecto en los navegadores
- El hecho de que con pipelining se puedan enviar varias peticiones sin haber terminado de recibir el objeto de las anteriores crea una **secuencia de transferencias**, de forma que un **objeto grande puede posponer otros más pequeños.**

- Sin pipelining, utilizando varias conexiones TCP simultáneas, se pueden enviar varias peticiones y recibir objetos a la vez, lo que da un **mejor rendimiento en la presentación de la página.**

5.6.3 Cachés y proxys

Caching: el cliente almacena las páginas descargadas en una caché local.

- Petición **Conditional GET:** se utilizan para descargar las páginas si es necesario
  - Puede utilizar **Date** y/o **Etag**

Servidor proxy: actúa como **intermediario** para las peticiones de los clientes. Ventajas:

- **Seguridad**: el proxy puede rechazar el acceso a servidores no autorizados
- **Logs**
- **Caché**
- Protege la dirección IP pública (sólo la conoce el proxy).

5.6.4 Web based applications

Componentes:

- **Presentación**: un navegador (lado del cliente)
- **Engine/motor**: genera las páginas HTML en el momento (lado del servidor)
  - **Lenguajes**:
    - Java
    - PHP (Hypertext Preprocessor): código html y lenguaje de programación.
    - Otros: ASP, CGI, Python...
- **Almacenamiento**: base de datos

Ventajas de las aplicaciones web:

- Fácil de desplegar(deploy) ya que solo es del lado del servidor
- Únicamente se requiere un navegador compatible en la parte del cliente
- Multi-plataforma

5.7 HTML y XML

5.7.1 HTML - Hyper-Text Markup Language

Definido en 1989 inspirado en otro lenguaje llamado SGML.

El objetivo principal de su diseño era mostrar en los navegadores documentos de texto formateados con hyperlinks (incluyendo enlaces a otros documentos).

Esta basado en etiquetas: ¡head¿ data ¡/head¿

Terminología:

- element: definido por la etiqueta
- attribute: atributo de un elemento
- text

Funciones de HTML:

- Forms: el documento acepta la entrada de datos por parte del usuario que son enviados al servidor.
- Scripting: permite añadir programas. El programa se ejecuta en la máquina del cliente cuando se carga el documento o cuando el enlace es activado.
  - Ej: javascript
    - ¡script tupe="text/javascript"¿...¡/script¿
- CSS: permite describir la plantilla física en un documento separado. Multiples páginas HTML pueden utilizar el mismo CSS y si el diseño se ha de cambiar sólo es necesario modificar el CSS.
  - Sintaxis: h1 color:blue, font-size:12px;

5.7.2 XML - Extensible Markup Language

Se desarrolló por el W3C para el **transporte y almacenamiento de información estructurada**.

No es un remplazo de HTML, sino un framework para definir lenguajes de markup.

XML por si solo no hace nada, es necesaria una aplicación para enviar, recibir o mostrar.

5.7.3 Limitaciones de HTML

Tenemos el ejemplo siguiente para una web de recetas:

Surgen los siguientes problemas:

- Como podemos comprobar si una receta se ha introducido correctamente? (ingredientes, canti-

```
<h1>Rhubarb Cobbler</h1>
<h2>Maggie.Herrick@bbs.mhv.net</h2>
<h3>Wed, 14 Jun 95</h3>
Rhubarb Cobbler made with bananas as the main sweetener.
It was delicious.  Basicly it was
  <table>
  <tr><td> 2 1/2 cups <td> diced rhubarb
  <tr><td> 2 tablespoons <td> sugar
  <tr><td> 2 <td> fairly ripe bananas
  <tr><td> 1/4 teaspoon <td> cinnamon
  <tr><td> dash of <td> nutmeg
  </table>
Combine all and use as cobbler, pie, or crisp.
Related recipes: <a href="#GardenQuiche">Garden Quiche</a>
```

dades...)

- Como identificamos los campos de la receta? (autor, ingredientes...)
- Que pasa si queremos mostrar los elementos en un orden diferente?
- ...

Es necesario definir las semánticas de las etiquetas, y la sintaxis (que etiquetas, y como deben ser utilizadas).

Solución: XMLization

```
<recipe id="117" category="dessert">
  <title>Rhubarb Cobbler</title>
  <author><email>Maggie.Herrick@bbs.mhv.net</email></author>
  <date>Wed, 14 Jun 95</date>
  <description>
    Rhubarb Cobbler made with bananas as the main sweetener.
    It was delicious.
  </description>
  <ingredients>
    <item><amount>2 1/2 cups</amount><type>diced rhubarb</type></item>
    <item><amount>2 tablespoons</amount><type>sugar</type></item>
    <item><amount>2</amount><type>fairly ripe bananas</type></item>
    <item><amount>1/4 teaspoon</amount><type>cinnamon</type></item>
    <item><amount>dash of</amount><type>nutmeg</type></item>
  </ingredients>
  <preparation>
    Combine all and use as cobbler, pie, or crisp.
  </preparation>
  <related url="#GardenQuiche">Garden Quiche</related>
</recipe>
```

Un documento XML **bien formateado** satisface un conjunto de reglas de sintaxis aportadas por la especificación. Es mucho más rígido que HTML.

XML namespaces: Permite diferenciar nombres de elementos definidos por desarrolladores distintos.

- Definido por el atributo **xmlns** que hay en el inicio del tag de un elemento
- Las URLs acostumbran a ser usadas para como una manera fácil de definir namespaces "únicos".

Los documentos XML tienen **estructura de árbol.**

XPath: utilizado para navegar por documentos XML.

- Sintaxis para seleccionar partes de un documento XML.
- Utilizado en XSLT (XML transformation language)

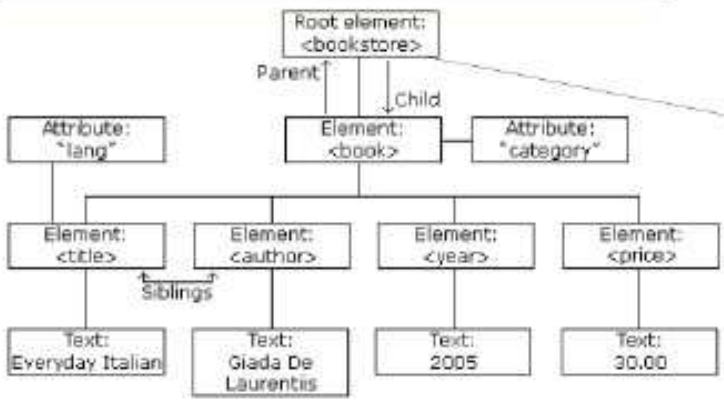
```
<widget xmlns="http://www.widget.org"
        xmlns:xhtml="http://www.w3.org/TR/xhtml1"
        type="gadget">
  <head size="medium"/>
  <big><subwidget ref="gizmo"/></big>
  <info>
    <xhtml:head>
      <xhtml:title>Description of gadget</xhtml:title>
    </xhtml:head>
    <xhtml:body>
      <xhtml:h1>Gadget</xhtml:h1>
      A gadget contains a big gizmo
    </xhtml:body>
  </info>
</widget>
```

default namespace.

namespace with prefix xhtml. The prefix acts as a shortname for the namespace.

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
...
</bookstore>
```

Terminology:
- element
- attribute
- text



… other books

- Ejemplo: /bookstore/book[1]/title

Validación de documentos XML

- Es válido si satisface la sintaxis del XML schema.
  - Define que etiquetas pueden usarse.
- Lenguajes de esquema más utilizados:
  - **DTD:** Document Type Definition
    - El primer lenguaje de esquema.
    - No sigue la sintaxis de XML.
  - **XSD:** XML Schema Definition
    - Sigue la sintaxis de XML
    - Puede expresar reglas más complejas que DTD.

- Document Type Definition, DTD
  - Content of the file "note.dtd":

```
<!ELEMENT note (to,from,heading,body)>
<!ATTLIST note date CDATA #IMPLIED>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Source: http://www.w3schools.com/xml/

element "note" contains the elements to,from,heading,body.

element "note" has attribute "date" (#IMPLIED means "not required").

element "to" contains character data (#PCDATA).

  - Reference to the DTD defined in "note.dtd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note date="10/01/2008">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Declaration starting an XML document.

DTD schema defined in location "note.dtd".

  - Validation example with xmllint (http://xmlsoft.org/):

```
linux ~/> xmllint --dtdvalid note.dtd exemple-dtd.xml
exemple-dtd.xml validates
```

- XML Schema Definition, XSD
  - Content of the file "note.xsd":

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
            <xs:element name="to" type="xs:string"/>
            <xs:element name="from" type="xs:string"/>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

http://www.w3schools.com/xml/

namespace where the schema is defined, the namespace should be prefixed xs.

root element

complexType: contains other elements

sequence: child elements must appear in the same order

  - Reference to the XSD defined in "note.xsd":

```
<?xml version="1.0"?>
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XSD schema defined in location "note.xsd"

- An XML file using XSD can also be validated with xmllint.

5.7.4 XLS: Extensible Stylesheet Language

Extiende la idea de CSS en HTML.

El principal componente es **XSLT**(XSL Transformations):

- Lenguaje de programación para especificar transformaciones entre documentos XML y un determinado lenguaje objetivo (p.ej: HTML).

- Todos los navegadores conocidos lo soportan.

Una hoja de estilo XLS consiste en una o más normas llamadas **templates**, que son aplicadas cuando un nodo especifico esta asocidado.